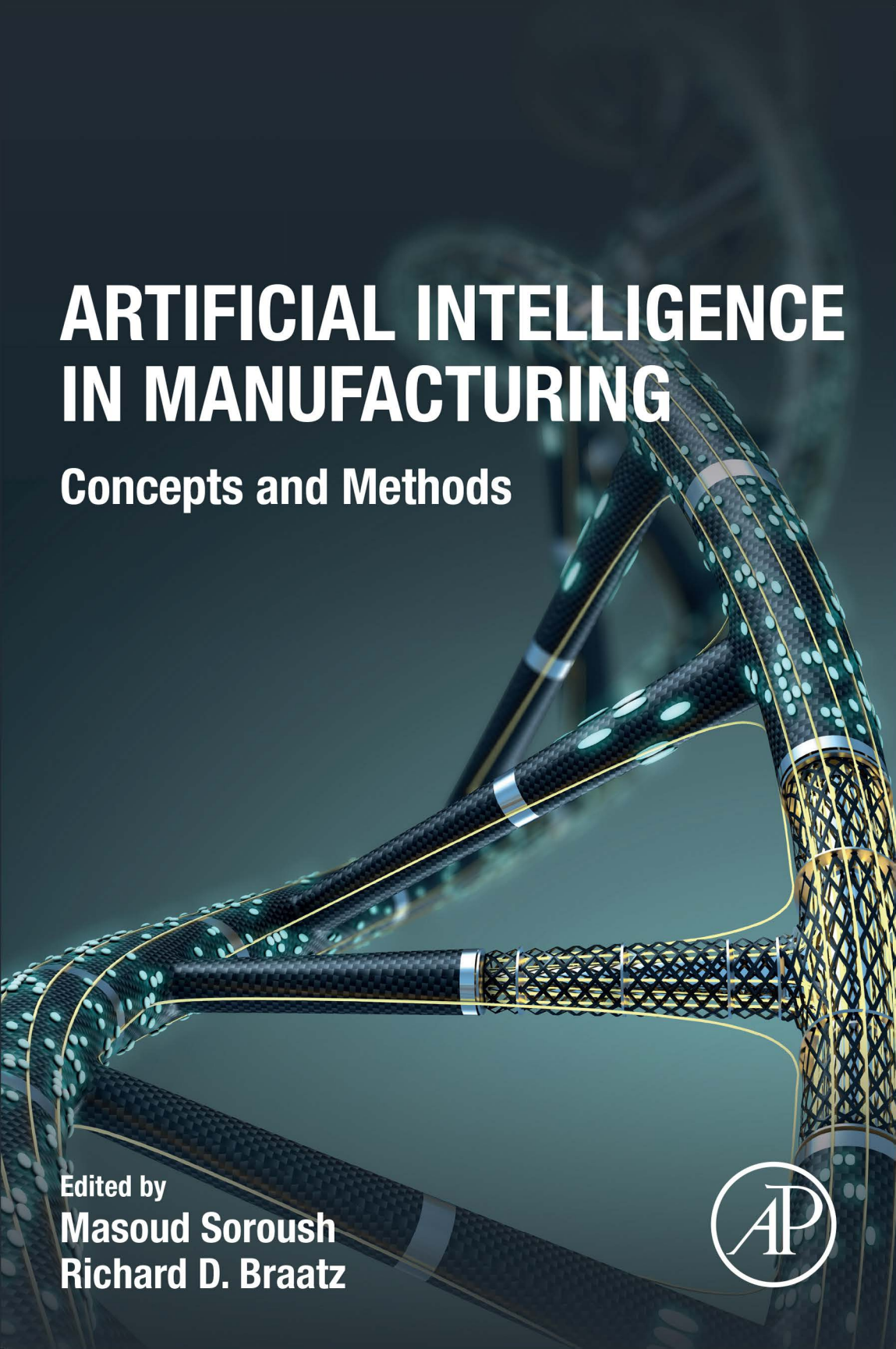
ARTIFICIAL INTELLIGENCE IN MANUFACTURING

Concepts and Methods



Edited by
Masoud Soroush
Richard D. Braatz

ARTIFICIAL INTELLIGENCE IN MANUFACTURING

Concepts and Methods

This book belongs to Central Library -
BADJI Mokhtar-ANNABA UNIVERSITY

Copyright Elsevier 2024

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY

Copyright Elsevier 2024

ARTIFICIAL INTELLIGENCE IN MANUFACTURING

Concepts and Methods

Edited by

MASOUD SOROUSH

Department of Chemical and Biological Engineering, Drexel University, Philadelphia, PA, United States

RICHARD D. BRAATZ

Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA, United States

ELSEVIER

ACADEMIC PRESS

An imprint of Elsevier

elsevier.com/books-and-journals

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY

Copyright Elsevier 2024

Academic Press is an imprint of Elsevier

125 London Wall, London EC2Y 5AS, United Kingdom
525 B Street, Suite 1650, San Diego, CA 92101, United States
50 Hampshire Street, 5th Floor, Cambridge, MA 02139, United States

Copyright © 2024 Elsevier Inc. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or any information storage and retrieval system, without permission in writing from the publisher. Details on how to seek permission, further information about the Publisher's permissions policies and our arrangements with organizations such as the Copyright Clearance Center and the Copyright Licensing Agency, can be found at our website: www.elsevier.com/permissions.

This book and the individual contributions contained in it are protected under copyright by the Publisher (other than as may be noted herein).

Notices

Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods, professional practices, or medical treatment may become necessary.

Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information, methods, compounds, or experiments described herein. In using such information or methods they should be mindful of their own safety and the safety of others, including parties for whom they have a professional responsibility.

To the fullest extent of the law, neither the Publisher nor the authors, contributors, or editors, assume any liability for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

ISBN: 978-0-323-99134-6

For Information on all Academic Press publications visit our website at https://www.elsevier.com/books-and-journals

Publisher: Matthew Deans
Acquisitions Editor: Sophie Harrison
Editorial Project Manager: Howi De Ramos
Production Project Manager: Fizza Fathima
Cover Designer: Miles Hitchen

Typeset by Aptara, New Delhi, India

Working together to grow libraries in developing countries
www.elsevier.com • www.bookaid.org

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY

Copyright Elsevier 2024

Contents

Contributors *ix*
Preface *xiii*

1. Machine learning methods **1**
Daniel Schwartz, Ali Shokoufandeh, Michael Grady and Masoud Soroush

1.1. Introduction 1
1.2. Holistic view of learning models 2
1.3. Classification of learning techniques 11
1.4. Machine learning methods 18
1.5. Conclusion 33
Acknowledgment 34
References 34

2. Learning first-principles knowledge from data **39**
Jaewook Lee, Weike Sun, Jay H. Lee and Richard D. Braatz

2.1. Background 39
2.2. Approaches to analyze manufacturing data 41
2.3. Automation of model selection and hyperparameter search 55
2.4. Conclusion 59
References 60

3. Convolutional neural networks: Basic concepts and applications in manufacturing **63**
Shengli Jiang, Shiyi Qin, Joshua L. Pulsipher and Victor M. Zavala

3.1. Introduction 63
3.2. Data objects and mathematical representations 65
3.3. Convolutional neural network architectures 71
3.4. Case studies 77
3.5. Conclusion 96
Acknowledgment 97
References 97

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY
Copyright Elsevier 2024

4. Sparse mathematical programming for fundamental learning of governing equations 103

Fernando Lejarza and Michael Baldea

4.1. Introduction 103
4.2. Problem definitions 105
4.3. Physics-informed machine learning 108
4.4. Regression-based approaches 112
4.5. Techniques based on mathematical programming 118
4.6. Demonstration of moving horizon discovery for a batch chemical process 124
4.7. Conclusion 131
References 132

5. Data-driven optimization algorithms 135

Burcu Beykal and Efstratios N. Pistikopoulos

5.1. Introduction 135
5.2. Algorithmic approaches for data-driven optimization 137
5.3. Applications to large-scale manufacturing systems 151
5.4. Extensions to other classes of problems 152
5.5. Remarks 165
5.6. Conclusion 165
References 166

6. Machine learning for control of (bio)chemical manufacturing systems 181

Andreas Himmel, Janine Matschek, Rudolph Kok (Louis), Bruno Morabito, Hoang Hai Nguyen and Rolf Findeisen

6.1. Introduction 181
6.2. (Bio)chemical processes 185
6.3. The machine learning Oracle and machine learning approaches in a nutshell 193
6.4. Machine learning-supported modeling for monitoring and control 208
6.5. Control via machine learning 219
6.6. Conclusion 228
References 230

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

7. Learning first principles systems knowledge from data: Stability and safety with applications to learning from demonstration 241

Ashwin Dani and Iman Salehi

7.1. Introduction 241
7.2. Learning robot motions using dynamical systems primitive 249
7.3. Conclusion 258
Acknowledgment 261
References 261

8. Artificial intelligence for materials damage diagnostics and prognostics 265

Sarah Malik and Antonios Kontsos

8.1. Introduction 265
8.2. AI methods for materials diagnostics and prognostics 269
8.3. Challenges and opportunities for AI methods for damage diagnostics and prognostics 298
8.4. Conclusion 299
References 300

9. Artificial intelligence for machining process monitoring 307

Hakkı Özgür Ünver, Ahmet Murat Özbayoğlu, Cem Söyleyici and Berk Barış Çelik

9.1. Introduction 307
9.2. Data acquisition systems 311
9.3. Feature engineering and machine learning 316
9.4. Signal decomposition methods 331
9.5. Deep learning 334
9.6. Transfer learning 338
9.7. Conclusion 342
Acknowledgment 343
References 343

Index 351

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY

Copyright Elsevier 2024

Contributors

Michael Baldea
McKetta Department of Chemical Engineering, The University of Texas at Austin, TX, United States; Oden Institute for Computational Engineering and Sciences, The University of Texas at Austin, TX, United States

Burcu Beykal
Department of Chemical and Biomolecular Engineering, University of Connecticut, Storrs, CT, United States; Center for Clean Energy Engineering, University of Connecticut, Storrs, CT, United States

Richard D. Braatz
Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA, United States

Berk Barış Çelik
Department of Mechanical Engineering, TOBB University of Economics and Technology, Ankara, Turkey

Ashwin Dani
Electrical and Computer Engineering, University of Connecticut, Storrs, CT, United States

Rolf Findeisen
Technical University of Darmstadt, Control and Cyber-Physical Systems, Darmstadt, Germany

Michael Grady
Innovation Center, Axalta Coating Systems, Philadelphia, PA, United States

Andreas Himmel
Technical University of Darmstadt, Control and Cyber-Physical Systems, Darmstadt, Germany

Shengli Jiang
Department of Chemical and Biological Engineering, University of Wisconsin-Madison, Madison, WI, United States

Rudolph Kok (Louis)
Otto von Guericke University Magdeburg, Systems Theory and Automatic Control, Magdeburg, Germany

Antonios Kontsos
Theoretical & Applied Mechanics Group, Department of Mechanical Engineering and Mechanics, Drexel University, Philadelphia, PA, United States

Jaewook Lee
Korea Advanced Institute of Science and Technology, Daejeon, South Korea

Jay H. Lee
Korea Advanced Institute of Science and Technology, Daejeon, South Korea

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY
Copyright Elsevier 2024

Fernando Lejarza
McKetta Department of Chemical Engineering, The University of Texas at Austin, TX, United States

Sarah Malik
Theoretical & Applied Mechanics Group, Department of Mechanical Engineering and Mechanics, Drexel University, Philadelphia, PA, United States

Janine Matschek
Technical University of Darmstadt, Control and Cyber-Physical Systems, Darmstadt, Germany

Bruno Morabito
Otto von Guericke University Magdeburg, Systems Theory and Automatic Control, Magdeburg, Germany

Hoang Hai Nguyen
Technical University of Darmstadt, Control and Cyber-Physical Systems, Darmstadt, Germany

Ahmet Murat Özbayoğlu
Department of Artificial Intelligence Engineering, TOBB University of Economics and Technology, Ankara, Turkey

Efstratios N. Pistikopoulos
Texas A&M Energy Institute, Texas A&M University, College Station, TX, United States; Artie McFerrin Department of Chemical Engineering, Texas A&M University, College Station, TX, United States

Joshua L. Pulsipher
Department of Chemical Engineering, University of Waterloo, Waterloo, ON, Canada

Shiyi Qin
Department of Chemical and Biological Engineering, University of Wisconsin-Madison, Madison, WI, United States

Iman Salehi
Electrical and Computer Engineering, University of Connecticut, Storrs, CT, United States

Daniel Schwartz
College of Computing and Informatics, Drexel University, Philadelphia, PA, United States

Ali Shokoufandeh
College of Computing and Informatics, Drexel University, Philadelphia, PA, United States

Masoud Soroush
Department of Chemical and Biological Engineering, Drexel University, Philadelphia, PA, United States

Weike Sun
Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA, United States

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Cem Söyleyici
Department of Mechanical Engineering, TOBB University of Economics and Technology, Ankara, Turkey

Hakkı Özgür Ünver
Department of Mechanical Engineering, TOBB University of Economics and Technology, Ankara, Turkey

Victor M. Zavala
Department of Chemical and Biological Engineering, University of Wisconsin-Madison, Madison, WI, United States

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY

Copyright Elsevier 2024

Preface

Research and development efforts in recent years have led to considerable advances in the use of artificial intelligence (AI) in manufacturing. This monograph puts these advances into perspective and shows how manufacturing industries can benefit from these advances. Results developed by leading academic and industrial groups in this area are consolidated, along with the AI tools and methods, which are presented in a systematic way. A comprehensive coverage is provided of AI concepts and methodologies. This book is complemented by the companion book titled "Artificial Intelligence in Manufacturing: Applications and Case Studies," which focuses on the state-of-the-art applications of AI concepts and methods in manufacturing industries.

Chapter 1, "Machine learning methods," provides an introduction to machine learning (ML) methods. Existing ML methods are reviewed from different viewpoints and placed in perspective. The strengths and weaknesses of each method and future prospects are discussed.

Chapter 2, "Learning first-principles knowledge from data," describes methods for learning first-principles knowledge from process data. Presented from a machine learning point of view, the chapter describes concepts and methods from linear and nonlinear regression needed to understand methods for learning first-principles knowledge from process data. Methods for learning first-principles knowledge are described, placed into perspective, and compared with popular alternative methods in the literature that do not attempt to learn first-principles knowledge. The chapter discusses the relationship between methods for learning first-principles knowledge to automated machine learning methods.

Chapter 3, "Convolutional neural networks: Basic concepts and applications in manufacturing," discusses the basic concepts of convolutional neural networks (CNNs) and outlines their uses in manufacturing. The representation of different types of data objects commonly encountered in manufacturing in a flexible manner using tensors and graphs is described. The chapter discusses how CNNs use convolution operations to extract informative features (e.g., geometric patterns and textures) from such representations to predict emergent properties and phenomena and/or to identify anomalies. The concepts are illustrated in case studies.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Chapter 4, "Sparse mathematical programming for fundamental learning of governing equations," discusses how physicochemical governing equations are fundamental mathematical expressions that describe the behavior of dynamical systems. The chapter covers recent advances in machine learning for discovering governing equations and provides comparisons against well-established surrogate modeling, systems identification, and parameter estimation problems. Important ongoing and future research directions in this field are discussed.

Chapter 5, "Data-driven optimization algorithms," highlights and explains the necessity of developing fast and efficient data-driven algorithms for mathematical optimization. Data-driven optimization algorithms are reviewed, starting from the earlier developments with direct search algorithms, and proceeding with model-based approaches and the most recent algorithmic developments to handle various classes of problems including mixed-integer nonlinear optimization. The use of the techniques is extensively demonstrated.

Chapter 6, "Machine learning for control of (bio)chemical manufacturing systems," discusses ways to fuse machine learning and control for the safe and improved operation of chemical and biochemical processes. Both learning models for control and learning the control components are outlined and summarized. The chapter provides a structured, general overview, including a literature review and a guideline for utilizing machine learning techniques in control structures.

Chapter 7, "Learning first principles systems knowledge from data: Stability and safety with applications to learning from demonstration," presents two methods of constrained learning of dynamical system models from data with applications to imitation learning in the context of manufacturing robotics. The chapter describes the field of imitation learning and discusses important problems in this area. Recent advances in the field are placed into perspective and promising future directions are discussed.

Chapter 8, "Artificial intelligence for materials damage diagnostics and prognostics," reviews AI methods used to investigate materials level damage. A perspective is provided on how the manufacturing industries can benefit from advances in using AI in different stages of materials qualification. Applications span the preprocessing, manufacturing operation, as-manufactured states, as well as postprocessing phases to succeed in producing manufactured parts with desired or at least acceptable final properties, attributes, and behavior.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Chapter 9, "Artificial intelligence for machining process monitoring," covers advances in intelligent machining process monitoring. The chapter discusses how AI models and machining technologies can be used for the balancing act between productivity, quality, cost, and sustainability measures of machine tool shops, and the broader manufacturing industry. A broad range of AI models and techniques used in matching process monitoring are described in terms of algorithmic fundamentals and recent case studies.

This book is useful for engineers, managers, and consultants in the manufacturing industries, and postdoctoral researchers, graduate students, and other researchers in academia and national laboratories engaged in the area of manufacturing. The book is a reference for finding the most recent advances in AI concepts and methods in manufacturing. This book is complemented by the companion volume titled "Artificial Intelligence in Manufacturing: Applications and Case Studies," which focuses on the state-of-the-art applications of AI concepts and methods in the manufacturing industries.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY

Copyright Elsevier 2024

CHAPTER ONE

Machine learning methods

Daniel Schwartz[a], Ali Shokoufandeh[a], Michael Grady[b] and Masoud Soroush[c]
[a]College of Computing and Informatics, Drexel University, Philadelphia, PA, United States
[b]Innovation Center, Axalta Coating Systems, Philadelphia, PA, United States
[c]Department of Chemical and Biological Engineering, Drexel University, Philadelphia, PA, United States

1.1 Introduction

Machine learning (ML) tools are increasingly capable of capturing and representing complex relationships with human or even superhuman accuracy in an increasing number of fields. For example, they are capable of accurately identifying patterns in massive noisy datasets.

Machine learning is a broad framework for data-driven mathematical modeling/representation of systems. The quality of the predictions made by this type of mathematical model improves as more data/experience becomes available [1]. In general, a mathematical model typically relates a set of effects (outputs [dependent variables]) to a set of causes (inputs [independent variables]). Such a model has a structure and parameter values:

$$Y = f(X; P) \tag{1.1}$$

where Y is the vector of the output variables, X is the vector of the input variables, P is the vector of the model parameters, and $f(.; .)$ is a vector operator describing the model structure. In the case of physicochemical (phenomenological) modeling, the model structure comes from chemical and physical principles, and parameter values from the same principles and/or from estimating the parameters from available input/output data. However, in the case of entirely-data-driven machine learning, both the model structure and parameter values come from the available historical data. Mathematically stated, in entirely data-driven machine learning, the model structure and parameter values are obtained by solving an optimization problem such as:

$$\min_{f \in F, P \in \Omega} \sum_{i=1}^{N} \left\| Z_i - f(W_i; P) \right\| \tag{1.2}$$

Artificial Intelligence in Manufacturing: Concepts and Methods.
DOI: https://doi.org/10.1016/B978-0-323-99134-6.00008-6

Copyright © 2024 Elsevier Inc.
All rights reserved, including those for text and data mining, AI training, and similar technologies.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY
Copyright Elsevier 2024

where $Z_1,\ldots, Z_N$ are the vectors of measurements of the output variables, $W_1,\ldots, W_N$ are the vectors of measurements of the input variables, $\|.\|$ is a vector norm, F is the set of operators that $f(.;\ .)$ belongs to, and $\mathbf{\Omega}$ is the set of values that the parameters P can take. In machine learning, the process of learning is that of finding the optimal operator f and the optimal values of the parameters P that are a solution to the optimization problem of Eq. (1.2).

The optimization problem in Eq. (1.2) can also be formulated with multiple objective functions. The multiple objective functions can be converted to a single objective function by considering a weighted sum of the individual objective functions. For example, the objective function can be a weighted sum of the squared residuals and the zero norm of the vector P (i.e., the number of nonzero components of P). Including the zero norms in the objective function allows for finding models with a smaller number of parameters and preventing "overfitting."

Once the optimal $f(.;.)$ and P are found (the model is trained with the historical data), the model can then be used for forward and backward inference. In the former case, it is desired to predict the response of the system to given values of the input variables, X; the values of the input variables are known, and the corresponding output-variables values need to be calculated. However, in the latter case, it is desired to know the input (cause) that the system should be subjected to, to obtain a desired response; the values of the output variables, Y, are known, and the corresponding input-variables values need to be calculated.

1.2 Holistic view of learning models

There are a variety of learning models. From a holistic point of view, these models are generally classified into four categories: supervised learning, unsupervised learning, hybrid learning, and reinforcement learning. The next four sections briefly review these categories.

1.2.1 Supervised learning

Supervised learning is the most popular category of machine learning models. These models learn from data that consists of example inputs and their corresponding outputs (or target labels). This is formally defined as input examples W_i and target labels Z_i for each example $i \in \{1, \ldots, N\}$. The target labels provide feedback guiding the algorithm through

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

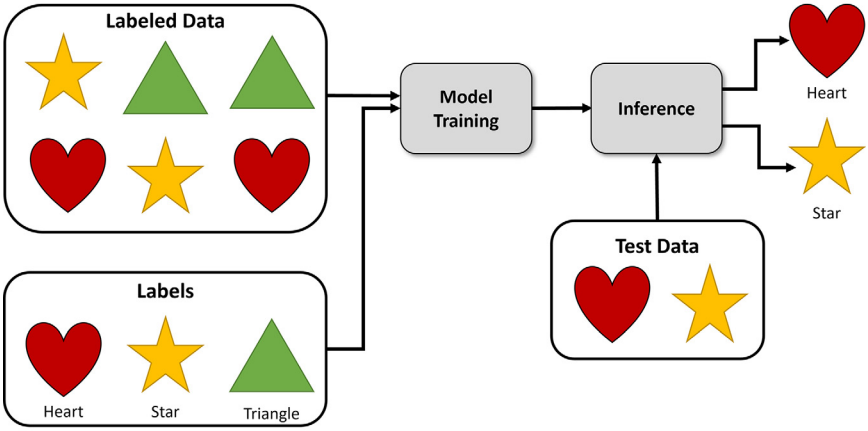


Figure 1.1 An example of a supervised learning model separating classifying shapes by their labels.

the process, "supervising" the algorithm into accurately capturing the relationships between the inputs and outputs. Major supervised learning models include *classification, regression*, and *adversarial learning*. The complete process of supervised learning is outlined in Fig. 1.1.

1.2.1.1 Classification

Classification algorithms aim to find the mapping function, f, where the input variables are continuous but the output variables are discrete. In this case, since the output variables are discrete, they are often referred to as labels or categories. The mapping function, f, predicts the label or category of a given observation from X. Some common algorithms used in classification are: linear classifiers [2], support vector machines [3], decision trees [4], and random forests [5]. Although the output variables are discrete, many classification algorithms often calculate a real number that represents the probability of a given input variable belonging to each category. In these algorithms, the class with the highest predicted probability is then chosen as the predicted class. A common evaluation (accuracy) metric for classification algorithms is the percentage of input examples for which the model predicts the output labels correctly. In simpler terms, classification algorithms seek to find the function f that optimally places the "fence" that separates the data.

1.2.1.2 Regression

In regression, the input and output variables are both continuous. Regression algorithms learn the mapping function f that relates these variables. Popular

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

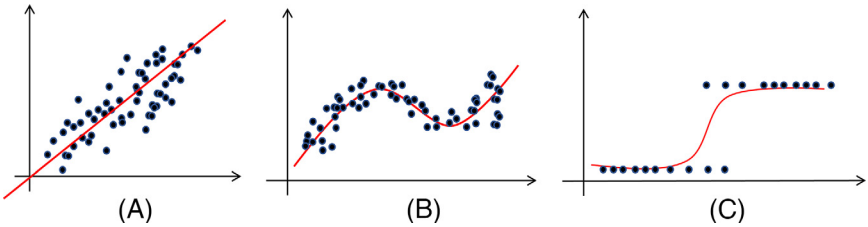


Figure 1.2 A visualization of regression: (A) linear regression, (B) polynomial regression, and (C) logistic regression.

regression algorithms are: linear regression [6], logistic regression [7], and polynomial regression [8]. Regression algorithms seek to learn relationships between the input (independent) and output (dependent) variables to predict output responses to input variable data. Unlike in classification algorithms where the output labels are discrete and the model can easily be evaluated by the metric of accuracy, regression algorithms predict a quantity, and the model can be evaluated in terms of the error in those predictions. One of the most widely-used measures of error in regression algorithms is the root mean squared error (RMSE) [9]. While classification separates data into classes, regression seeks to optimally fit a model to the data. An example of different classes of regression is shown in Fig. 1.2.

1.2.1.3 Adversarial learning

At a high level, a supervised learning model acts as a membership verification mechanism of high-dimensional data for different classes. Since the boundary of separation is not perfect, there is the ability to push one data point to cross the boundary and move from one class to another. This can be performed by perturbing the data in the direction of another class. Neural networks use some form of gradient descent algorithm to solve the optimization problem of the form presented in Eq. (1.2) (to train models). This also implies minor modification of the gradient process may impact the outcome of the original classifier construction. For instance, the fast gradient step method [10] adds little noise to the input at every step of the optimization to modify the prediction away from the ground truth. The noise can either direct the input to predict a specific class (targeted attack) or simply predict a class that is incorrect (nontargeted attack). Although it is very easy to fool the human eye with one of these adversarial attacks, neural networks can become more robust through the use of regularization and sparsity [11]. An example adversarial network, a generative adversarial

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

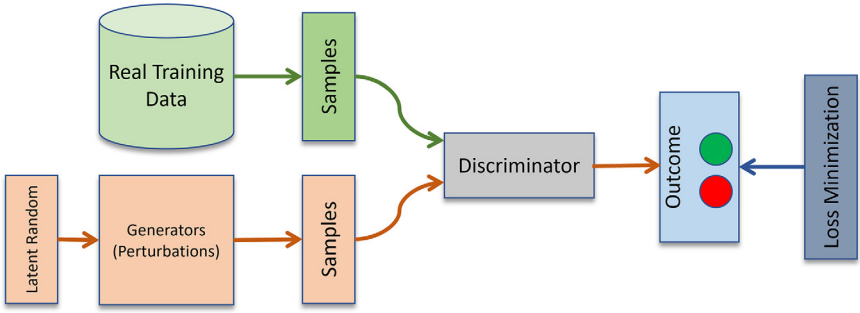


Figure 1.3 Architecture overview of a generative adversarial network (GAN).

network is shown in Fig. 1.3, which depicts the composition of two neural networks that compete with each other to become more accurate in their predictions [12]. The network consists of two components: the generator, which learns to map from a latent space to a data distribution of interest, and the discriminator, which evaluates the candidates generated by the generator and distinguishes between the true candidates and the generated ones.

1.2.2 Unsupervised learning

Unsupervised learning is similar to that supervised learning except that only the input examples are given without any measurements of the output variables. As a result, these algorithms all follow the same intuition to learn the inherent structure of the data without the use of target labels (output variable measurements). Unsupervised learning is often the first step in tackling a machine learning problem and can provide more intuition for explanatory analysis, attempting to better understand the data and any patterns that may exist within the data's latent structure. Thus, these algorithms aim to uncover hidden patterns in the input data without human intervention and are therefore "unsupervised" as shown in Fig. 1.4. This learning technique

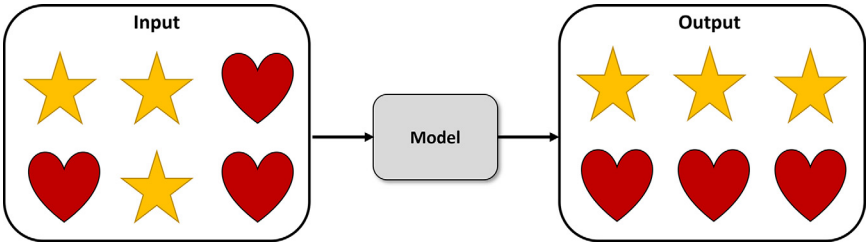


Figure 1.4 An unsupervised learning model separating objects with similar patterns into groups.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

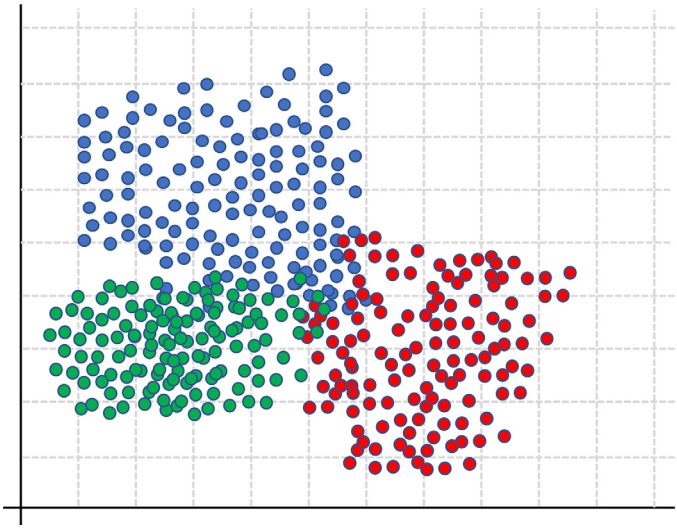


Figure 1.5 An example dataset that can be modeled using clustering.

is also used for dimensionality reduction and the determination of features (or dimensions) that are not as important and can be discarded for other learning tasks. Unsupervised learning methods are generally classified into three categories: *clustering, anomaly detection*, and *encoders*.

1.2.2.1 Clustering

In contrast to unsupervised learning, clustering makes use of what data is available, and the input features to generate or identify coherent groups within input data. As a result, clustering groups this unlabeled data by their similarities and differences. The k-means clustering algorithm [13] does just this and groups similar data points together where the hyperparameter of k defines the size of the grouping and granularity. An example of a clustering dataset is shown in Fig. 1.5.

1.2.2.2 Anomaly detection

The use of clustering can help detect outliers that exist within the input data. These "anomalous" data points are based on rare events or observations that are statistically different from the rest of the observations. Time series anomaly detection is a field in statistical inference that is a popular field in modern machine learning. Problems in this domain consist of data that has values associated with timesteps and attempts to predict anomalies in that sequential data. Since time series data often includes patterns such as

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

cyclic growth, periodicity, and poten tial decay, it is useful to predict changes that occur unpredictably. An example problem is predicting anomalies in stock prices such as a company unexpectedly drops in price and results in a huge loss for shareholders. Some common methods to perform anomaly detection are clustering, wavelet transforms, autoregressive models, and other digital signal processing techniques. These unsupervised techniques attempt to predict whether a data point is anomalous based on how similar it is to previous observations. Two assumptions in unsupervised anomaly detection are that only a small fraction of the data is anomalous and that an anomaly is statistically different from the rest of the samples. A simple approach based on these two assumptions clusters the data and then uses a similarity measure to determine which data points are different from the cluster.

1.2.2.3 Encoders

An encoder maps a high-dimension input into a latent low-dimensional code. An autoencoder is a type of artificial neural network (ANN) that learns how to encode inputs from a high-dimensional space to a low-dimensional latent space through unsupervised learning. The first step of this process is using an encoder as described before to compress data from the original input dimensions to a lower-dimensional latent space directly to a latent code. The second part of this architecture is the decoder, the reverse of the encoder, which reconstructs the data from the latent code in the encoded representation to the original input space. Finally, at the output of the decoder, the reconstruction is measured by a loss of how well it matches the original input. Together, this architecture is shaped as an hourglass where the first and last layers are the original high-dimensional space, and the middle "bottleneck" is much smaller and represents the encoded space. Autoencoders can be used to filter out the noise components of input data corrupted by noise. The objective of the autoencoder is to learn the original input without noise [14]. A measure of the performance of the low-pass filter (denoiser) is how well it can reconstruct the original input free of noise. An example autoencoder that learns the compressed representation of a handwritten digit is shown in Fig. 1.6 [15].

Variational autoencoders (VAEs) are another extension of autoencoders that provide the additional capability of describing a probability distribution for each latent attribute as opposed to just a single value. A VAE is similar to Bayesian models, where the input is mapped to a distribution [16]. While an autoencoder lacks a method to generate new contents, a VAE is trained such that it is regularized not to overfit and learns a latent representational space

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

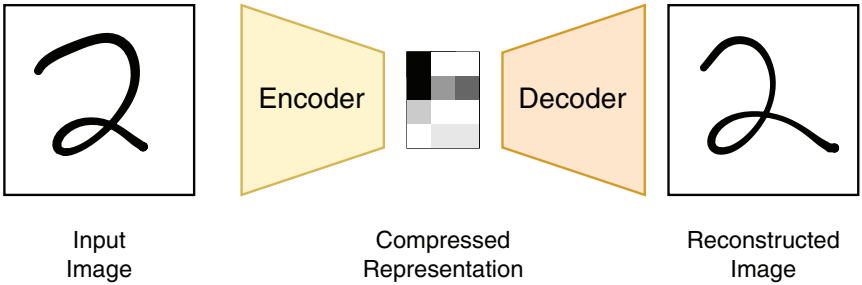


Figure 1.6 Autoencoder to learn compressed representations.

that enables a generative process. Hence, it encodes an input as a distribution over the latent space but not over a single point. Overfitting refers to a model that models the training data too well at the expense of excessive model complexity and size. Overfitting also occurs when a model attends too much to the specific attributes of the training samples, negatively impacting the performance of the model on data outside of the training distribution. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. As a result, the model learns too much about the irregularities of the training samples and fails to generalize. The regularization is measured/quantified using the Kulback-Leibler divergence [17] of the generated distribution and a standard Gaussian. This is to ensure that the distribution is complete such that any point sampled from the latent space has a meaningful context once decoded. VAEs are can be used in statistical methods such as variational inference and realistic human-face generation [18].

1.2.3 Hybrid learning

As discussed previously, there are two methods of learning: supervised which requires target labels, and unsupervised which does not require target labels. The generation of target labels may be a laborious/costly task, as it may require experts in the domain to annotate the data by hand. However, unlabeled data (that includes only input variable measurements) is easier to acquire and much less costly. As a result, there has been research on how to leverage unlabeled data to achieve satisfactory performance in supervised learning tasks with minimal labeled data.

1.2.3.1 Semi-supervised learning

As supervised learning can be very costly, is only helpful in exploratory analysis, and cannot help in classification, semi-supervised learning was introduced. In particular, semi-supervised learning is useful when data has

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

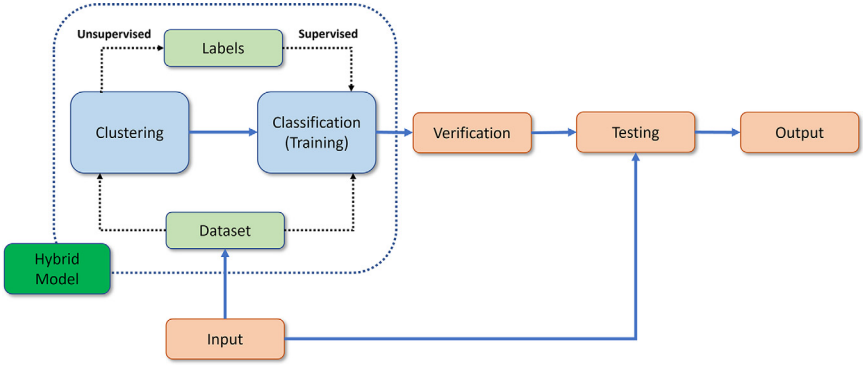


Figure 1.7 An example of hybrid learning architecture that combines clustering and classification models.

a few labeled instances and many instances of unlabeled data. Since a bulk of the data is unlabeled, standard unsupervised learning techniques such as clustering are performed and then the learned clusters are matched with the labeled data to perform a version of classification. This becomes a very useful process as the cost of labeling data is high and the acquisition of unlabeled data is relatively cheap. In short, the addition of unsupervised data to a supervised learning problem can increase generalization and performance while reducing the cost and time to develop the dataset [19]. An example architecture that combines an unsupervised clustering technique and a supervised classification technique used for hydrological assessment [20] is shown in Fig. 1.7.

1.2.3.2 Self-supervised learning

Self-supervised learning is similar to semi-supervised learning as it uses mostly unsupervised data and a few supervised data examples, except their procedure is quite different. The process starts with learning useful representations from the set of unlabeled data where the "self-supervision" occurs followed by a tuning of the representations with supervised data later. This technique has exploded in the field of natural language processing (NLP) in transformer models such as BERT [21] and T5 [22]. These models start by training with large amounts of unlabeled data and then are fine-tuned with several labeled data examples for specific downstream tasks.

1.2.3.3 Multi-instance learning

Multi-instance learning is an abstraction of supervised learning such that groups of samples are labeled instead of individual examples. This is considered a "weakly supervised learning" problem since groups (or bags) of

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

samples are labeled as opposed to individual samples [23]. However, this is a less expensive task since the number of labels required drops significantly. A common application of this technique is with medical imaging such that an image of a patient's cells needs to be classified as cancerous or not. Instead of labeling each minuscule cell with cancer, the whole image itself can be labeled and the patient can be classified as having malignant cells. This task becomes more efficient since the scale of data reduces and in the case of medical imaging it is an "all or nothing" task such that if any of the cells are cancerous than the patient has cancer [24]. This process avoids costly local annotations that may be required by an expert and can properly diagnosis a patient at a fraction of the cost.

1.2.4 Reinforcement learning

Reinforcement learning follows the intuition from the psychological concept [25], where an agent learns by trial and error using the feedback from its own actions and experiences. Unlike supervised learning which maps between inputs and outputs, the outputs of supervised learning take the form of a correct set of actions for performing a task, while the outputs for reinforcement learning use rewards and punishments as signals for positive and negative behavior. Reinforcement learning aims to maximize the total cumulative reward for the given agent and follows a standard action-reward feedback loop as seen in Fig. 1.8. The key components of a reinforcement learning problem are: (1) environment (process/system), the physical world the agent operates in; (2) state (input variable measurements), the current situation of the agent; (3) reward (output variable measurements), the feedback observed by the agent from the environment; and (4) policy (mathematical model), the map that relates the state of the agent (persistent memory) to the desired action (input variables). The value is a common term often used in the domain that defines the potential reward an action would grant an agent in the current state. The ultimate consideration in the policy of a reinforcement learning problem revolves around the idea of exploration *vs.* exploitation to explore all possible actions while also exploiting the best possible action at the current state.

Markov decision processes are a method of representing a reinforcement learning environment as a mathematical framework where the states and actions are represented as a set, the value of the reward is represented as a function that takes the next state as an input, and a transition model which takes the previous state as an input and the next state is given an action. A

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

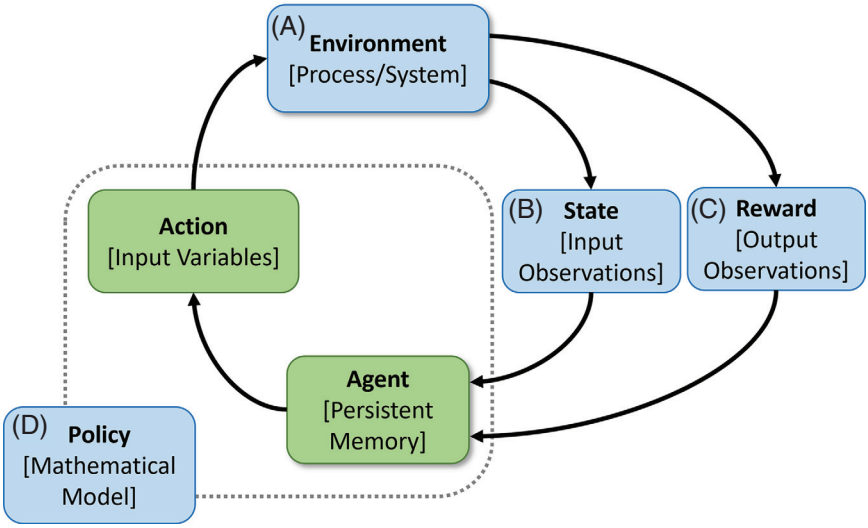


Figure 1.8 Generic action-reward feedback loop.

common technique to simulate a reinforcement learning problem is a Q-learning algorithm [26] that maintains a table of all learned values for each given state and action possibility. This algorithm attempts to learn the best values for Q, the "quality" of a given action gaining some reward, and the objective is to estimate the highest value of Q returned by an action for a given state. However, Q-learning is too simple and fails to generalize to estimate values for unseen states. Deep Q-networks (DQN) [27] use neural networks to estimate Q-values such as Deep Deterministic Policy Gradient [28]. Common applications of reinforcement learning are in gameplay (such as the game AI for Go [29]) and robotics (such as autonomous robotics [30]).

1.3 Classification of learning techniques

1.3.1 Data perspective

Statistical inference and machine learning are commonly used interchangeably. However, there exists a subtle difference between the two. Statistical inference is the process of using data analysis to infer properties of an underlying distribution of probability [31]. The term "inference" is used quite loosely in the field of machine learning as to learn or train a model and then predict based on evaluating the then-trained model. In short, the context of inference is to reach an outcome or decision and the process

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

of achieving that outcome is done via fitting a model and making a prediction from that model. Given the notion of inference, there are different approaches to learning: *inductive, deductive*, and *transductive*.

1.3.1.1 Inductive learning

Inductive learning is the idea of drawing general conclusions about future decisions from history. The intuition of inductive learning is to learn concepts that help develop a generalized rule to process new observations with the key distinction that this process requires past data as evidence. The goal is to estimate a general function derived from specific inputs and outputs; thus, generalizing from the samples. Estimating a mapping function is quite difficult and the result instead is a good approximation of the function. Inductive learning is especially useful when the function is not static and can change from new observations; such as predicting the stock market. This technique is also useful when the task has no formal solution or if it can be performed by a human [32], but there is no clear description of how to perform it directly. Examples of this can be learning how to drive a car or ride a bike.

1.3.1.2 Deductive inference

Deductive inference is the opposite of induction such that induction transitions from specific cases to general outcomes, while deductive inference uses general rules to determine specific outcomes. Furthermore, this type of inference is top–down and follows the process of reducing facts from a general theory to specific, factual conclusions. The first step is to collect many statements, premises, and reduce these statements down to a single logical conclusion. While induction and deduction are opposite in their definitions, they are closely related to the use of machine learning to learn from observations and make predictions. Induction is used to train a model on a dataset and fit the model to then make predictions. Making predictions from a trained model is the process of deductive inference and uses the general concepts learned in the training process to determine the specific outcome. Simply deductive inference derives values of a function given specific values. A comparison of the different types of learning is shown in Fig. 1.9.

1.3.1.3 Transductive learning

Transductive learning aims to predict specific examples given examples from a domain, but does so without any generalization. Since this form

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

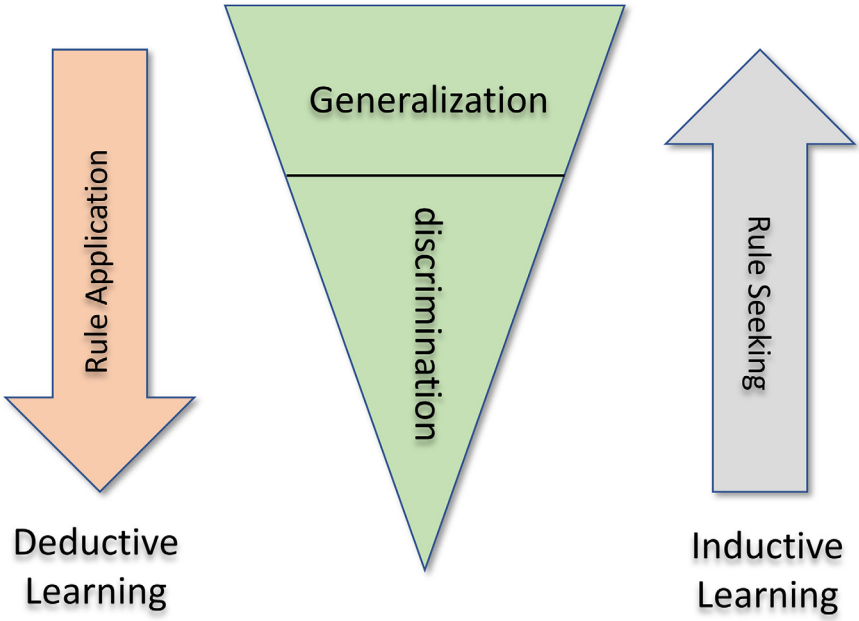


Figure 1.9 Infographic describing the differences in learning.

of learning does not generalize, an output function is not required, but the modeling directly classifies the data. However, with this process, if a new data point is added to the set, the whole model may need to be retrained since there was no generalization done to define an output function. As a result, transductive learning observes both the training and the testing datasets and attempts to learn patterns from the observed training data and labels to predict the labels of the testing data. Applications of transductive learning in the machine learning space are NLP tasks, trying to convert one string into another. Some examples are spelling correction, generate correct word spelling given incorrect word spelling, and machine translation, generating sequences of words in a target language given examples in a source language. Transductive learning can become computationally costly since it does not build a predictive model and can only predict data points it has already observed. A comparison of the different types of learning is shown in Fig. 1.10.

1.3.2 Algorithmic perspective

The boundary defining the differences between supervised and unsupervised learning is often not set in stone, there exist many hybrid approaches.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

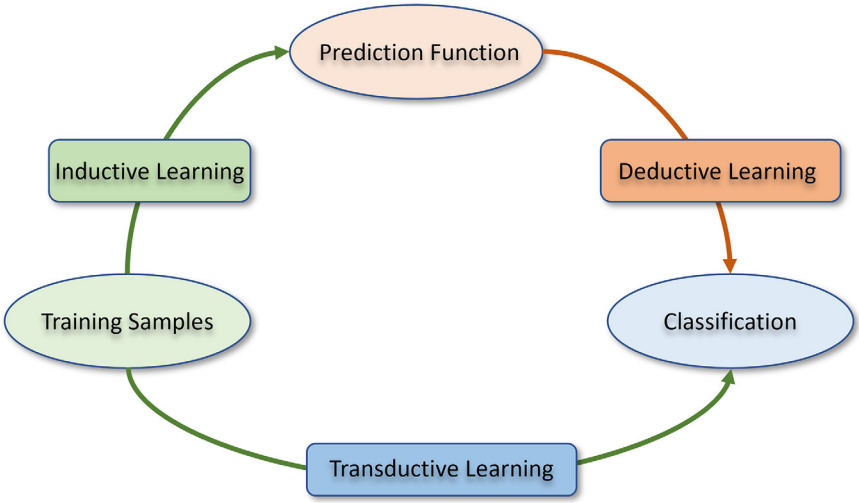


Figure 1.10 Flowchart describing the different types of learning.

In addition, the process of both training a machine learning model as well as the inference of a trained model can vary greatly. In the following section, different approaches to modeling and training a model are defined and explored.

1.3.2.1 Multitask learning

Multitask learning is a subset of machine learning algorithms that are designed to solve the optimization of more than one problem by considering a specific set of functions, *F*. Multitask learning as the name suggests is a machine learning algorithm that aims to solve multiple tasks at the same time; thus, solving a multiobjective function optimization problem (optimizing more than one objective function simultaneously). This is more efficient than training multiple models individually since the patterns learned across all the data will be able to define abstract generalized representations that can be applied to multiple contexts. However, multitask learning is only beneficial if the tasks that are to be learned have some level of relevance to each other and may result in worse performance, if they are unrelated. A common technique in multitask learning is to enforce parameter sharing so that all the tasks share the same learned representation of the data that is being learned. As the objective is to solve multiple tasks at once, the model must optimize multiple objectives with the same input data, increasing the generalization of the model. The implementation of multiple learning tasks at once is

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

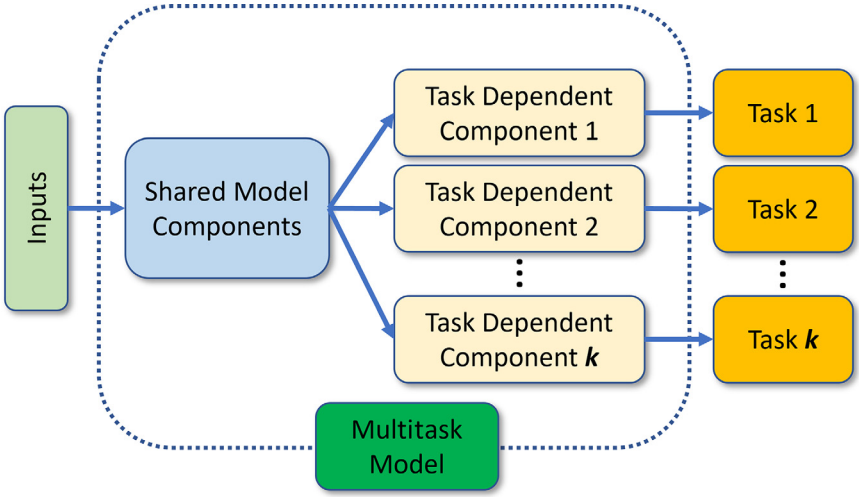


Figure 1.11 Architecture overview of multitask learning.

easy since the objective functions for each task can be combined into one, allowing for optimizing one objective function, which is a weighted sum of the multiple original objective functions. A visualization of the architecture of multitask learning is shown in Fig. 1.11.

1.3.2.2 Active learning

As detailed previously, labeling data is a costly and time-consuming task. Active learning aims to expedite the task of labeling data by improving both the efficiency and expressivity of the labeled dataset [33]. The objective of this process is to prioritize unlabeled data to be labeled by selecting the most significantly expressive data. This technique improves efficiency, as it enables the model to select a subset of the data to be trained on instead of the complete set of data, reducing the time to label all data. Additionally, it enables the expressivity of the dataset, since the data to be labeled is only the most expressive. The measure of expressivity can be chosen by a variety of methods. For example, uncertainty sampling chooses data that the model is most uncertain about when making a prediction. The uncertainty can be verified by the value of least probability when predicted, or by the measure of entropy (average level of information). Furthermore, if the predicted probabilities of the entropy are high, then the model is confused about that example and it cannot accurately classify that example, making it a prime candidate for labeling. Query by committee provides a different

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

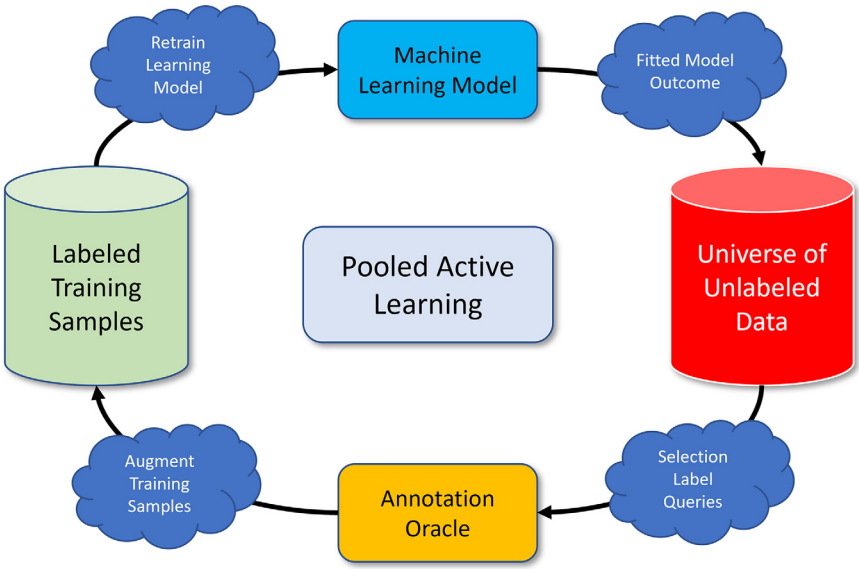


Figure 1.12 Architecture overview of active learning.

method of choosing uncertain examples to label, this process uses a variety of models and chooses the examples that are different across all the models [34]. All in all, active learning forces only the most significant and expressive examples to be labeled enabling the most demonstrative and efficient model. A visualization of the architecture of active learning is shown in Fig. 1.12.

1.3.2.3 Online learning

Since machine learning is often applied to large datasets, it is likely that the entire data cannot fit into the computer memory when training. In this case, learning needs to be performed incrementally over observations. The traditional approach to machine learning is offline such that all data is present and available at all times. However, online learning is the opposite, where data is given as a stream and estimates must be made "online," instead of waiting until all data is available. Instead of the traditional metric of minimizing error, online learning aims to minimize regret; that is, how well the model performed in comparison to how well it performs if all available data was given [35]. Online learning poses a complex problem since the model evolves over time and the data is time-dependent. Thus, it requires a high level of computational power to continuously ingest the data and refine the model. Online learning is commonly applied in the domain of finance and economics because new data patterns are constantly emerging [36].

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

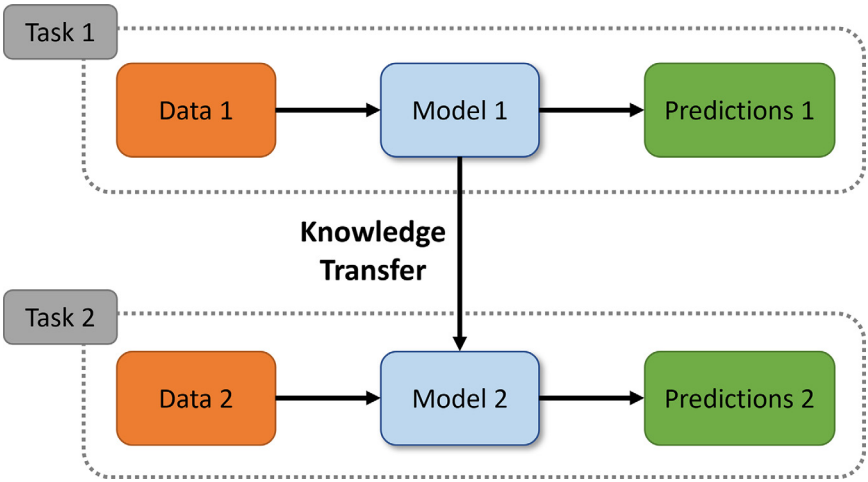


Figure 1.13 An example of the use of transfer learning that applies knowledge learned from a previous task/domain to a new one.

1.3.2.4 Transfer learning

There are cases where a task is related to many other similar tasks. In such cases, based on the task one model can be trained, and the model can be used as a starting point for the additional tasks. In short, transfer learning is a learning technique that a model is developed for a general first task and then repurposed on a second related task to improve performance and reduce training time on that second task. Transfer learning is a common technique in the computer vision domain, specifically with a deep neural network (DNN) like Oxford's Visual Geometry Group (VGG) model [37] or Microsoft's ResNet model [38] trained on the ImageNet dataset [39]. ImageNet consists of over 14 million images spanning 1000 classes, which makes it a perfect general dataset to transfer its learning to smaller more specific computer vision classification problems. Furthermore, a DNN trained on ImageNet has learned to extract features from pictures to make classifications of many different classes on a large corpus of images and should be able to properly extract features from photographs in photographs it has never seen before. An example of the transfer learning process is shown in Fig. 1.13. The second set of layers in this model are designed to learn the classification of specific vehicles. However, the second set of layers can be swapped and modified to transfer the learning of the features to another downstream task.

A note in using transfer learning in DNNs is that the earlier layers in the model have features learned that are more generic than the layers

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

toward the end since those later layers learn specific attributes related to the dataset. Similarly in the language domain, it is common to perform transfer learning with NLP. The use of word embeddings, a representation of a word that has a similar representation to words with similar meaning [40], is a common technique to employ such transfer learning. Furthermore, the learning of these word embeddings acts as the source model, which learns the contextual meaning of the words, can then be applied to other downstream tasks such as text classification [41], named entity recognition [42], and event extraction [43].

1.3.2.5 Ensemble learning

The process of developing a machine learning model is solving the optimization problem in the form of Eq. 1.2. However, a single model may not be the best solution. Ensemble learning is a technique where multiple models are fit on the same data, and the predictions from the models are combined. For instance, decision trees are a useful solution in solving quantitative problems, but they are impacted by the correct set of features chosen as well as deciding upon a threshold to segregate features. Instead of hoping that the decision tree created is the best model, ensemble learning can be applied and multiple decision trees can be generated and a final prediction can be computed based on the aggregated results of the decision trees generated (often referred to as Random Forest). The Random Forest is an example of Bootstrapping and Aggregation where multiple bootstrapped models are generated for a given sample and the prediction is aggregated over those models. Another approach to ensemble learning is stacked learning where multiple models learn in parallel and are then fed as inputs to a higher-level model as the main predictor for a better prediction as shown in Fig. 1.14. Lastly, the ensemble method of boosting has a model learned from previous models in an attempt to improve future predictions. In boosting such as XGBoost [44] and AdaBoost [45], the models learn from each other in a sequential manner where each model can learn from the previous model sequentially to rectify the errors they had.

1.4 Machine learning methods

1.4.1 Dimensionality reduction

Machine learning works directly on data, specifically operating on many samples of data where the size of the data is measured by the number of

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

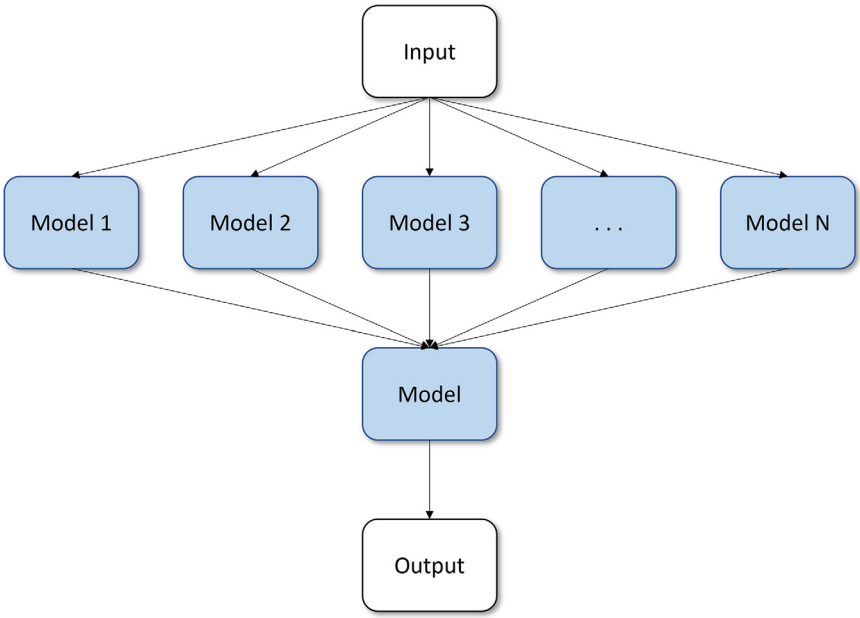


Figure 1.14 Ensemble learning as represented through a stacking ensemble.

samples and the number of features per sample. The number of features or input variables for a sample is often referred to as its dimensionality. For many tasks, there are a myriad of features and in most cases, only a few of the features are important whereas the rest are redundant or dependent upon others. While more data can be perceived as good, there is the curse of dimensionality that more input features make a machine learning task more difficult. As a result, the technique of dimensionality reduction attempts to reduce the number of features in a dataset. Dimensionality reduction has a few benefits by reducing the input feature set. The benefits include: (1) fewer redundant features and noise, (2) less storage space, (3) a lower chance of overfitting, (4) useful for data visualization, (5) less computing, and (6) faster training [46]. The survey by Fodor [46] details a study that showcases how effective dimensionality reductions can be for clustering data that becomes difficult in the original space.

Of many existing methods for computing dimensionality reduction, one of the most common are linear dimensionality reduction methods: principal component analysis (PCA) and linear discriminant analysis. PCA [47] is a form of matrix factorization that reduces a dataset matrix into parts. The parts can then be ranked and a subset of those parts represent the

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

most important features. The principal components are equivalent to the eigenvectors of the dataset matrix's covariance matrix and can easily be computed by an eigendecomposition of the covariance matrix or through singular value decomposition [48] of the original dataset matrix.

Another technique for reducing dimensionality is feature selection, a process that identifies and selects which features to keep and which to remove through statistical methods. Random forests are one such way to keep the most relevant features. Random forests work by generating hundreds of trees such that each tree is built from a series of random observations in the dataset. A feature is represented in a tree at a node branching off in two possible directions, yes if the feature is true, and no if the feature is false. As a result, when generating a tree, the impurity of a feature can be computed and the more a feature decreases the impurity, the more important the feature. Since random forests contain many trees, each feature can be averaged across all trees to compute the importance of a feature variable.

1.4.2 Neural networks and deep learning

1.4.2.1 Multilayer perceptrons

Artificial neural networks follow from the inspiration of biological neural networks composed of biological neurons. A biological neuron is composed of (1) dendrites that receive incoming signals, (2) a cell body that processes the input and passes the information, (3) an axon that passes the transmission of signals, and (4) a synapse that is at the end of the axon and connects to other neurons. An ANN takes these structures and symbolically applies them in perception, a single layer of an ANN. A perceptron takes as input signals, multiplies them by weights, adds a bias, applies an activation function to the inputs, and then propagates the output forward. The steps just described are referred to as forward propagation, but one of the most important steps is backward propagation (backpropagation) which optimizes to find the best parameters that fit the model by propagating the expected output backward through the network. Perceptrons can be stacked on top of one another into multilayer perceptrons (MLPs) to learn complex functions as represented in Fig. 1.15. Often, the first layer of "neurons" in an MLP is known as the input layer, the last layer is known as the output layer, and the layers in between are referred to as hidden layers since they are not directly exposed to the input. Since the backpropagation step learns the best weights that map the input to the output, a form of optimization is required, often in the

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

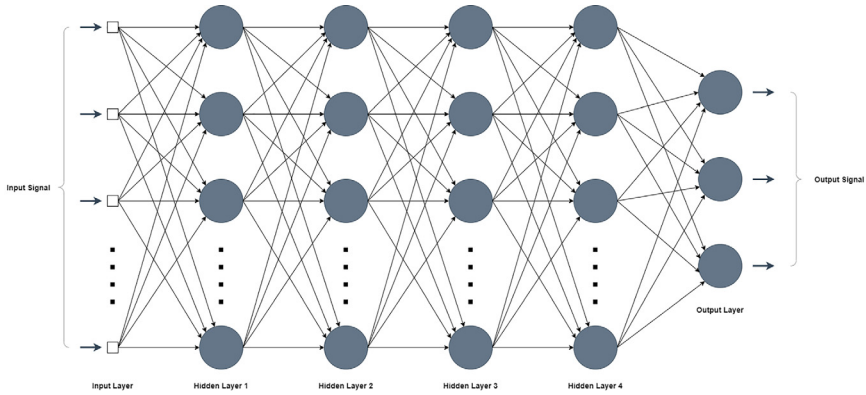


Figure 1.15 Example representation of a multilayer perceptron.

form of a gradient descent that attempts to minimize the predicted output from the expected output. In an MLP, every neuron in the previous layer is connected to every neuron of the next layer, such that the complete network is a fully connected network (FCN). A key improvement of an MLP from a traditional perceptron is that a traditional perceptron cannot distinguish data that is not linearly separable whereas an MLP can. MLPs can be composed of many layers and become deep resulting in a DNN that can approximate very complicated functions.

1.4.2.2 Recurrent neural networks

Multilayer perceptrons suffer from an inability to model sequential data. Recurrent neural networks (RNNs) are a category of neural networks which stores temporal information and gives it temporally dynamic behavior. There are several ways of implementing RNNs depending on the use case, including long–short-term memory (LSTM) networks, bidirectional RNNs, and hierarchical RNNs. An RNN feeds its output to itself at the next time step forming a loop and passing down information as needed. A simple RNN can be unrolled and be a chain-like structure similar to sequences as depicted by Dong et al. [49] and represented in Fig. 1.16. At each time step, the input from the previous time step is encoded and used in conjunction with the current input being passed [50]. However, the RNN does suffer from some drawbacks, especially that of the vanishing gradient problem. This results when RNNs have a very small gradients due to activation functions squashing a large input space into a smaller space between 0 and 1. A large change in the input to the sigmoid function will cause a small change in

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

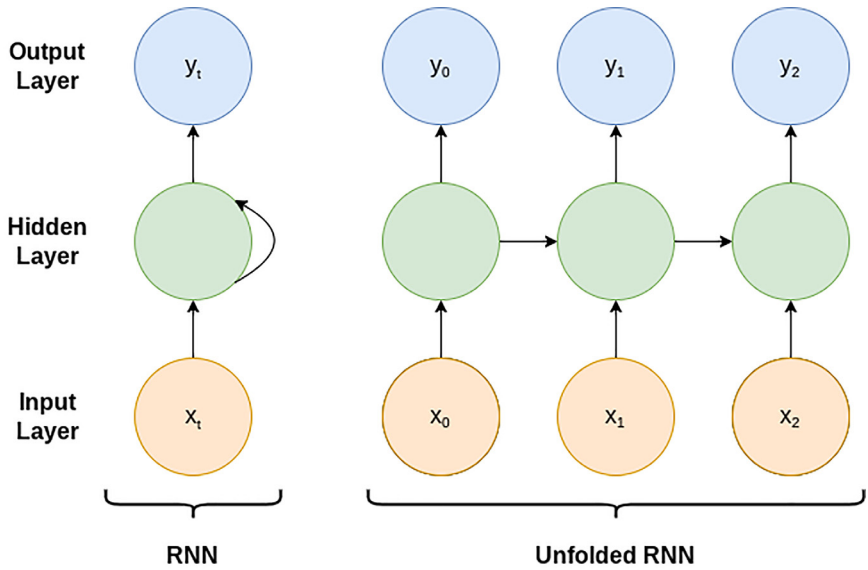


Figure 1.16 A recurrent neural network unfolded to a chain-like structure.

the output and due to the looping nature of its nodes constantly feeding into one another, the derivative becomes small and the gradients approach 0. In some cases, this may completely halt the learning of the network as the gradient is vanishingly small and prevents the weight from being updated.

1.4.2.3 Convolutional neural networks

Within the deep learning domain, convolutional neural networks (CNNs) have been the most effective architectures in computer vision tasks [51]. CNNs extract abstract features from images by convolving the image with filters. These filters are then subsampled via pooling layers, commonly using max-pooling that partitions the image into regions and returns the maximum value within each region. Finally, the last element of a CNN is the fully-connected layer where each node is directly connected to every node in the previous and next layer to pass forward the results from the previous layer. One of the most notable achievements in CNNs was displayed in the AlexNet architecture [52] that used a faster convolution operation implementation on graphical processing units (GPUs) that achieved state-of-the-art results when it was introduced. One of the first tasks CNNs solved was classifying handwritten digits using the MNIST dataset [53]. Yann LeCun is known as the founding father of CNNs and his initial architecture,

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

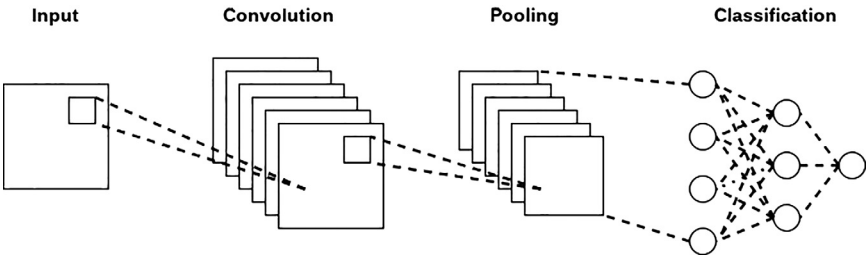


Figure 1.17 Example convolutional neural network architecture pipeline.

LeNet [54], consisted only of two sets of convolutional layers followed by pooling layers completed with two fully-connected layers seen in Fig. 1.17.

Since the introduction of LeCun's CNN and the use of the modified National Institute of Standards and Technology (MNIST) database, ImageNet [55] has been one of the most popular datasets used to evaluate new computer vision architectures. This dataset is derived from WordNet [56], a lexical database consisting of English words grouped into a hierarchical structure that relates similar semantic words. ImageNet consists of millions of images organized by the hierarchy of WordNet consisting of over 1000 distinct categories. Since ImageNet has over 100 times more categories than MNIST, architectures applied to classify ImageNet pictures use deeper layers. The exploration of very deep convolutional networks to classify images from many classes resulted in VGG [57], a deep neural network that consisted of over 16 convolutional layers and three fully-connected layers. The intuition of the additional layers of convolutions and pooling is that these layers progressively learn more complex features. Furthermore, the first set of layers could learn edges, the second set of layers could learn shapes, the following could learn mouths, and the last could recognize faces. Yet, He et al. [38] showed that adding more layers can make a more complex function and be prone to overfitting. Some research has shown that the use of regularization and sparsity can help to prevent overfitting [11,58].

1.4.2.4 Residual networks

The addition of the residual block by He et al. [38] has shown significant improvement in the failure of very deep neural networks improving accuracy from shallower, smaller networks. The key innovation of a residual block (depicted in Fig. 1.18) is that there is a "skip connection" identity mapping that has no parameters but simply outputs the previous layer to the next layer. The block helps maintain the context of previous information that would have been lost otherwise since very deep networks fail to propagate

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

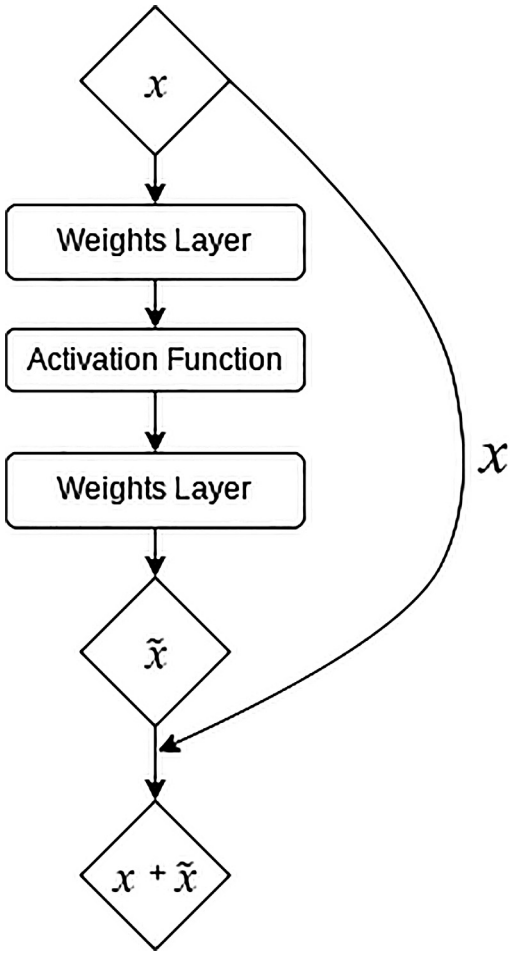


Figure 1.18 Residual block introduced [38].

information from earlier layers to later ones, resulting in the degradation problem. Furthermore, as deep networks start to converge with the depth of the network increasing, the accuracy gets saturated and degrades quickly. These "skip connections" enable earlier layers to easily propagate information to later layers without modifying it and passing the identity mapping instead. The intermediate layers passing the original value from the earlier layers merely need to learn the residual or change in value, from the original input. Learning the residual is much easier to facilitate than learning the identity function from scratch. As a result, residual networks can help alleviate the degradation problem in very deep networks by short-circuiting shallow layers to deeper layers.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

1.4.2.5 Graph neural networks

As deep learning is used more and more, a distinction arises between associating features and their respective target outcomes and associating features with one another. A preeminent computer science combinatorial representation comes to knowledge, graphs, and sets of objects with connections between them. Graphs are a useful structure to represent both entities and the connections between them. Newer research in the field has improved their capabilities and expressive powers. Often graphs are associated with social networks, navigation systems, or chemical structures; yet, graphs can express even more. Images can be represented as graphs such that every pixel represents a node and is connected via an edge to adjacent pixels. Text can even be represented by a simple directed graph where each character is a node and is connected to subsequent characters (or nodes) by edges.

A graph neural network (GNN) can be as simple as an MLP applied to each component of the graph such that for each node vector the MLP is applied and a learned node embedding vector is returned. The same process follows for each edge outputting an edge embedding and finally, a single embedding to represent the global context vector for the complete graph. For each embedding of the different graph components, each item is pooled together by concatenating the embeddings together to a single matrix and then aggregated via a summation. Given the data is a graph, there are a variety of tasks we can learn from them. The objective of a graph-level task is to learn a property of an entire graph such as in the case of an image graph, the prediction of the object within the image [59]. GNNs are often applied to node-level tasks as well, such as predicting the label of a node in a given graph. The classic example is discussed by Zachary [60], where the objective is to classify every node into one of two karate clubs. GNNs can be applied to incomplete data as discussed by Kipf and Welling [61] where using an efficient variant of a CNN to encode local structures and features. These networks can even be applied to higher-level graph structures like mesh that represent 3D geometric objects. One such network, HyNet [62] is a novel GNN that segments different components of an unstructured mesh of 3D objects in high resolution.

1.4.3 Natural language processing

Natural language processing is a field in machine learning that gives machines the ability to understand and learn text and language. Simply, NLP researchers aim to teach machines to understand what is said and written to make sense of the human language. Some common NLP tasks include

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

question answering, speech recognition, predictive typing, document summarization, machine translation, sentiment analysis, etc. NLP can be further subdivided into natural language understanding (NLU) and natural language generation (NLG). NLU tasks are focused on interpreting the meaning of natural language while the inverse is NLG, the learning aims to map data into written narrative. Solving an NLP problem is a hard task because it is a very hard goal to fully understand and represent the meaning of a language. In addition, machine learning models often learn from structured data, and one of the significant challenges in NLP applications is that human speech is often unstructured and ambiguous. As a result, it becomes increasingly difficult for computers to completely understand the intent and emotion behind words and sense the tone when translating and understanding.

The current state-of-the-art performs NLP tasks via syntactic and semantic analysis techniques. The semantics of a sentence denote the meaning behind the words, whereas the syntax often refers to the grammatical structure of the text. Moreover, semantic analysis is the technique of understanding and interpreting the words, symbols, tone, and structure of the sentence by analyzing the meaning or logic. Named entity recognition (NER) is a problem that involves semantic analysis, the goal of this task is to determine the terms of a text that can be categorized into groups such as the names of people and locations. On the other hand, syntactic analysis refers to analyzing the text for formal grammar such as parsing. Other tasks that fall under syntactic analysis are part-of-speech tagging or stemming, slicing words to their root form.

Due to the unstructured nature of the input text, significant preprocessing is required for NLP tasks. Often times the text is converted to lowercase and punctuation is removed since the additional information acts as noise and increases the size of the data resulting in an increase in computational complexity. Other nominal data is removed such as numbers and nonsignificant words, deemed "stop words," such as "in," "what," "the," "and," etc. Finally, the text is tokenized and broken down into minimal meaningful units and then stemmed and lemmatized. As detailed previously, stemming removes any suffices from a word and outputs the root word. Lemmatization, on the other hand, is the process of extracting a root word by searching through a dictionary to find the root word.

1.4.3.1 Word embeddings

As detailed in the NLP section aims to understand and learn the text. However, neural networks cannot directly learn from strings of text, but

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

they learn from numerical data. Word embeddings are models that learn to map a set of words in a vocabulary to vectors of numerical values. Word embeddings follow from transfer learning and aim to improve the ability of networks to learn from text data. Often word embeddings are representations of the text data in low-dimensional vectors to improve computational efficiency. Traditionally, words are mapped to one-hot encodings, essentially binary vectors with the length of the vector matching the length of the vocabulary and one representing the particular word to represent in that vocabulary. This one-hot encoding is very inefficient since most values in the vector are set to zero and when used in a neural network, the output will have mostly zero values. In addition, by using the logic of cosine similarity, this would suggest that all these words are orthogonal to one another and that words cannot be compared for similarity in context.

Low-dimensional word embeddings are used instead and are computed by using a fully-connected layer of a network. This layer is often referred to as an embedding layer and the weights of this layer are referred to as embedding weights. Since the vectors are one-hot encoded, the multiplication of the one hot vector and the embedding weight matrix simply returns a dense vector representing the embedding and can be thought of simply as a lookup table. As a result, this process is called "embedding lookup," and the dimension of this hidden layer is the "embedding dimension." Word2Vec is one of the most popular word embedding models used today. It was developed by Google and trained on Google News data [40]. This model was trained on over 3 million words and has an embedding dimension of 300 (Fig. 1.19). Global Vectors (GloVe) is the word embeddings developed by Stanford ranging from 25, 50, 100, 200, to 300 dimensions implemented by 2, 6, 42, to 840 billion tokens [63]. GloVe is implemented with word-to-word co-occurrence such that if two words occur next to one another many times then they have a semantic similarity. Finally, fastText as developed by Facebook with three separate models, each at 300 dimensions, does not just use word representations, but at the character level as well [64].

1.4.3.2 Gated recurrent unit

The gated recurrent unit (GRU) as introduced by Cho et al. [65] aims to solve the vanishing gradient problem. This module uses update and reset gates that decide what information should be passed to the output. The GRU is trained to retain information from a long time ago without

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

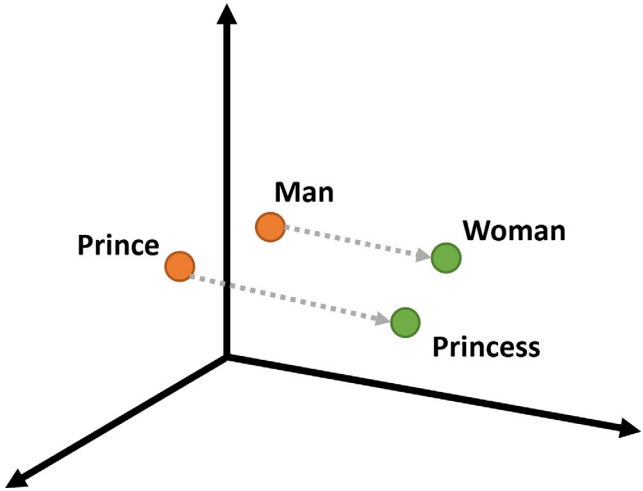


Figure 1.19 A visualization of the latent space learned from word embedding algorithms (i.e., Word2Vec).

forgetting about it over time or discarding information. The update gate is composed of the sum of the input multiplied by its weight matrix, concatenated with the previous time step multiplied by its weight matrix, and all of this is sent through a sigmoid activation function to squash the result between 0 and 1. The intuition of this gate learns how much of the past information should be passed into the future. This is the most significant part of the unit since it can learn to copy up to all of the past information and reduce the chance of the vanishing gradient occurring. On the other hand, the reset gate is used to decide how much of the past information to forget. Additionally, the GRU has fewer tensor operations and removed the use of a cell state to speed up the processing compared to its counterpart, LSTM [66].

1.4.3.3 Sequence-to-sequence

Seq2seq models are used to convert sequences of one type to sequences of another type. For example, the translation of English sentences to German sentences is a seq2seq task. As detailed above in the encoding/embedding section, the typical setup for this type of translation utilizes an encoder that takes the source sequence as input and outputs an encoding, then a decoder that takes the encoding and outputs a sequence in the target language. The encoder and decoder are typically built using a similar architecture to one another and are typically based on either an RNN or LSTM.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

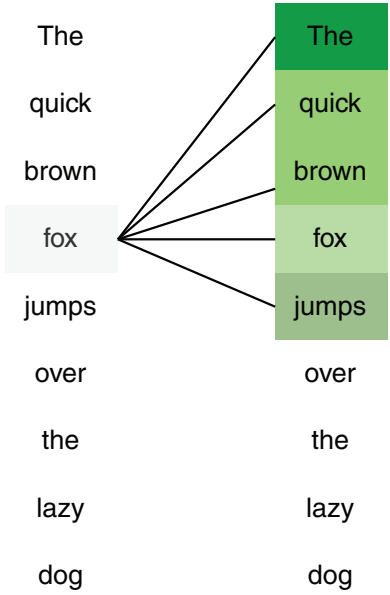


Figure 1.20 Visualization of an example application of an attention module relating the input word to the rest of the sentence. Attention architectures provide a model for capturing the context and inner dependencies of complex systems.

1.4.3.4 Attention

The notion of attention was first introduced in conjunction with seq2seq models for solving NMT [67]. Seq2seq models are traditionally composed of an encoder-decoder architecture where the encoder maps the input sequence to a context vector of fixed length and the decoder generates the transformed output given this vector. As a result of the fixed-length vector, this model suffers from the inability of the system to remember longer sequences. Many times, the beginning part of the sequence has been forgotten once the entire sequence is processed. Seq2seq models discard all the intermediate states of the encoder and only use its final states to initialize the decoder. This tends to work well for short sequences, but as the length of the sequence increases, this compressed vector becomes a bottleneck and struggles to encode all the data of a long sequence into the vector. Attention is introduced to utilize all the states in order to construct the context vector and not discard those intermediate encoder states. Xie et al. [68] highlight the long-distance dependency modeled by an attention module, which is represented in Fig. 1.20.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Attention uses a sub-network to determine which hidden encoder states to "attend" to and by how much. This sub-network consists of a fully-connected dense layer followed by a softmax and the probabilities generated are the attention weights that aid the decoder in choosing how much attention to focus on a specific set of tokens in the input sequence. With attention, it is clear to see which tokens in the output sequence depended on specific tokens in the input sequence and can normally be represented in terms of a heat map, deemed an attention map.

1.4.4 Machine learning as an interpolation function

Machine learning at a high level uses algorithms that gradually adjust weights to best fit a given dataset. There are a variety of approaches to improve the performance of a machine learning model, but most only have empirical guarantees and cannot provide a formal theoretical convergence. As a result, we explore the notion of posing machine learning as an interpolation function and examine the properties related. We explain the hardness of machine learning as a problem as well as aim to solve this via convexification and probably approximately correct learning.

1.4.4.1 Hardness of problem

The mathematical optimization approach to training a machine learning model by minimizing the error of mapping examples to their correct label, Eq. 1.2, is NP-hard as defined by Hazan [69]. While NP-hard problems are regarded as a class of problems that are very difficult in computational complexity, the crucial concept to take away is that mathematical optimization in machine learning is hard [70–72]. This form of optimization is hard, since the global optimal solution cannot be verified by the local solutions observed. Machine learning optimization algorithms are iterative via local improvement steps and halt when local improvements are no longer possible. In other words, they find a local optimum. As a result, the local solution (optimum) generated by these optimization methods can be quite different from the global optimal solution. Mathematical optimization for machine learning greatly benefits from the notion of convexity, where global optimality can be verified locally.

1.4.4.2 Convexification

As stated before, machine learning algorithms across the board utilize optimization in one form or another. Most learning algorithms tend to minimize loss, or error, or maximize some kind of scoring function, as in (2). Gradient descent tends to be the "de facto" optimization algorithm

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

that searches for an optimal solution. Many times if not all, the functions that are being optimized are nonconvex, wavy such that some local minima are not as deep as the overall deepest global minima. In short, convex optimization implies that an objective function's local minima is equivalent to its global minima. On the other hand, many optimization algorithms for nonconvex problems can get stuck in the local minimum, and it can be hard to tell when this happens and if the process will ever converge. In recent years, there have been advances in global optimization methods. Convex optimization methods, however, are faster, simpler, and less computationally intensive. Thus, learning algorithms could observe all these attributes, if they were formulated in a convex-friendly setting, or "convexified." Recall that a function $g(x) : \mathbb{R} \rightarrow \mathbb{R}$ is said to be convex, if it satisfies the condition:

$$g\big(tq_1 + (1-t)q_2\big) \leq t\,g(q_1) + (1-t)g(q_2) \qquad \forall q_1, q_2 \in Q, \forall t \in [0, 1] \tag{1.3}$$

Overall, machine learning is applied to problems where there is no optimal solution guaranteed, and the best one can achieve is a good approximation. Alternatively, given a convex optimization problem, since the problem only needs to find the minima once, it can do so with limited resources and well-defined bounds and provides a stable and exact output to the "convexified" approximation. The resulting convex optimization problem is a low-cost trainable model and provides a benefit over existing ML problems that usually require expensive GPUs and hours to days of training. Convexifying neural networks provide an alternative approach to learning an approximation of neural networks, but the nature of convexified learning algorithms is still in its infancy stage and more exploration must be done. We now look at a case study that convexifies a traditional DNN and how this affects the performance of the model.

1.4.4.3 Convexified deep neural network

As mentioned previously, a CNN uses convolutional filters to extract local features and generalize better than FCN. However, this process requires non-convex optimization and fails to guarantee a globally optimal solution. There have been many papers proposing attempts to improve the performance of CNNs [73–76], but none of them are guaranteed to be as good as the classical CNN.

Zhang et al. [77] described a new architecture, a convexified convolutional neural network (CCNN), that achieves competitive or better performance than CNNs, support vector machines (SVMs), stacked

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

denoised auto-encoders, etc. CNNs have advantages of sparsity such that each nonlinear convolutional filter acts only on a local patch of input and parameter sharing where the same filter is applied to each patch. However, the standard approach to training CNNs is computed by solving a nonconvex optimization problem that is NP-hard. In addition, calculating gradients in backpropagation has a rate of convergence that is slow due to nonconvexity and the related statistical properties are difficult to understand.

Convexified convolutional neural network provide a mechanism to solve a convex optimization problem which is solved efficiently via a projected gradient algorithm. The statistical properties of CCNN can be studied in a precise and rigorous manner and it replaces the nonlinear activation functions of a CNN with a reproducing kernel Hilbert space (RKHS). While the two-layer CCNN guarantees strong optimality, in a deeper CCNN, only the empirical results are improved by convexification. It is clear that nonconvex optimization is hard, but as seen in CCNN, nonconvex optimization is not required to approximate a CNN. CCNN solves the two issues of nonconvexity in traditional CNNs by using RKHS filters instead of CNN filters and relaxing the parameter sharing to a nuclear-norm constraint. As a result, this leaves the open question of "is there a better abstraction for machine learning?"

1.4.4.4 Measuring error through probably approximate correct learning

The ultimate goal in machine learning is to find an approximation that is very close to the process that generated the data and we are trying to learn. Probably approximate correct (PAC) learning provides a metric to explain how far an approximation concept is from the true concept. A naive method of comparing how well the approximate concept learns compared to the ground truth concept is by separating data into training and testing where the model is only trained on the training data and evaluated on the testing data. However, it is unlikely that the testing data segregated completely imitates the complete pool of the ground truth concept; PAC learning guarantees with high probability that the right concept is derived from the training data. PAC learning suggests not so much that a model underperformed when its accuracy on unknown values does not match its training data, but that the model actually overperformed on the training data and that the training data was not an accurate representation of the true concept.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

1.5 Conclusion

Machine learning methods have advanced greatly in the past few decades. Machine learning models are now used in many applications, as they are increasingly capable of capturing and representing complex relationships and are increasingly robust and versatile. However, machine learning methods have more room to grow. As they become more robust and versatile, we will witness more applications of them, especially in process industries. An area that has a lot of room to grow is the area of combined (hybrid) data-driven and phenomenological modeling. Such models will have the robustness and ability-to-extrapolate of phenomenological models and the versatility and computational efficiency of machine learning models.

Current machine learning methods can capture surface-level characteristics of data; they are unable to capture higher-order abstractions that would permit the methods to connect those characteristics to the external domain of knowledge and properly reason about what the observation. Advances in hybrid modeling may be able to address this weakness.

The complexity of training ML models directly impacts their performance. The size of training data and the accuracy of their representation in terms of dimensionality directly impact the model construction. As a result, many state-of-the-art machine learning techniques employ very deep neural networks that cost more both in time and resources to train. As the number of parameters within these very deep neural networks begin to outgrow the number of data points, the models begin to overfit and lose the ability to generalize to unseen data. Overfitting is a significant problem for most ML models.

If the learning is realizable and the family that the learning function belongs to is finite, then we can bound the number of training samples that results in a small empirical error. Yet, these are very strong requirements and finding problems that satisfy these conditions is very hard since most real-world problems cannot satisfy these conditions feasibly. On the other hand, if sample distributions do not have a good representation, the outcome is overfitting. Most models need uniform distribution of samples from all classes in the distribution of sample space to avoid overfitting. However, in practice, the dataset given is often an unknown distribution and the problem of overfitting cannot be solved without using techniques such as sparsity and regularization.

As the popularity of DNNs dominated the research field to make deeper and more complicated architectures, some research looked into imposing

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

sparsity in deep learning to improve the trade-offs of efficiency and accuracy. Deep expander networks [78] leverage the notion of sparsity to improve the connectivity and the flow of information across layers in DNNs while maintaining state-of-the-art performance in a more efficient manner. Another technique applied in deep learning to increase sparsity in the network is the use of dropout [58], which randomly ignores some number of layer outputs during training "dropping them out." This technique simulates a sparse activation in the given layer; thus, learning a sparser representation results in more robust networks. Other methods of imposing sparsity and regularization schemes like dropout have been applied to neural networks to make them more sparse resulting in improved performance [79].

Beyond classification, ML models can act as surrogates for many optimization problems. This is increasingly relevant for many engineering applications for which creating an exact interpolation becomes intractable. ML models can receive structured data as graphs, as graphs are becoming extremely popular. Graphs are already a common input for computer vision, pattern recognition, and soon other computational domains. Yet, this is an area that requires lots of research in learning associations within the structural inputs. An example is when the structure of the input is a graph with nodes and an adjacency structure. ML models that use such type of structured data as an input, may also be required to act as a generative model. Specifically, such networks might be required to predict a graph-like structure as output that satisfies certain properties or requirements. Such generative models are becoming very relevant for both engineering applications and structural pattern recognition [80,81].

Acknowledgment

Daniel Schwartz, Ali Shokoufandeh, and Masoud Soroush acknowledge financial support from the U.S. National Science Foundation (NSF); this material is partially based upon work supported by the U.S. NSF under Grant No. CBET–1953176. Any opinions, findings, and conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] T. Mitchell, Machine Learning, McGraw Hill, Burr Ridge, 1997.
[2] R.O. Duda, P.E. Hart, et al., Pattern Classification and Scene Analysis, 3, Wiley, New York, 1973.
[3] B.E. Boser, I.M. Guyon, V.N. Vapnik, A training algorithm for optimal margin classifiers, in: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, 1992, pp. 144–152.
[4] J. Ross Quinlan, Induction of decision trees, Mach. Learn. 1 (1) (1986) 81–106.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[5] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[6] J.F. Kenney, E.S. Keeping, Linear regression and correlation, Mathematics of Statistics 1 (1962) 252–285.
[7] C.R. Boyd, M.A. Tolson, W.S. Copes, Evaluating trauma care: the TRISS method. Trauma score and the injury severity score, J. Trauma 27 (4) (1987) 370–378.
[8] J.D. Gergonne, The application of the method of least squares to the interpolation of sequences, Historia Mathematica 1 (4) (1974) 439–447.
[9] J. Scott Armstrong, F. Collopy, Error measures for generalizing about forecasting methods: empirical comparisons, Int. J. Forecast 8 (1) (1992) 69–80.
[10] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, arXiv preprint arXiv:1412.6572 (2014).
[11] D. Schwartz, Y. Alparslan, E. Kim, Regularization and sparsity for adversarial robustness and stable attribution, in: International Symposium on Visual Computing, Springer, 2020, pp. 3–14.
[12] I. Goodfellow et al., Generative adversarial nets, Advances in Neural Information Processing Systems 27 (2014).
[13] J. MacQueen, et al., Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1, Oakland, CA, 1967, pp. 281–297.
[14] P. Vincent, et al., Extracting and composing robust features with denoising autoencoders, Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 1096–1103.
[15] D. Bank, N. Koenigstein, R. Giryes, Autoencoders, arXiv preprint arXiv:2003.05991, 2020.
[16] D.P. Kingma and M. Welling. Auto-encoding variational Bayes. In: arXiv preprint arXiv:1312.6114 (2013).
[17] S. Kullback, R.A. Leibler, On information and sufficiency, Ann. Math. Stat. 22 (1) (1951) 79–86.
[18] A. Vahdat and J. Kautz. Nvae: A deep hierarchical variational autoencoder. In: arXiv preprint arXiv:2007.03898 (2020).
[19] X.J. Zhu, Semi-supervised learning literature survey, *University of Wisconsin-Madison Department of Computer Sciences*, 2005.
[20] J. Kim, et al., Hybrid machine learning framework for hydrological assessment, J. Hydrol. 577 (2019) 123913.
[21] J. Devlin et al. Bert: Pre-training of deep bidirectional transformers for language understanding. In: arXiv preprint arXiv:1810.04805 (2018).
[22] C. Raffel et al. Exploring the limits of transfer learning with a unified text-to-text transformer. In: arXiv preprint arXiv:1910.10683 (2019).
[23] M. Ilse, J. Tomczak, M. Welling, Attentionbased deep multiple instance learning, in: International Conference on Machine Learning, PMLR, 2018, pp. 2127–2136.
[24] G. Campanella, et al., Clinical-grade computational pathology using weakly supervised deep learning on whole slide images, Nat. Med. 25 (8) (2019) 1301–1309.
[25] B.F. Skinner, Contingencies of Reinforcement: A Theoretical Analysis, 3, BF Skinner Foundation, 2014.
[26] Christopher John Cornish Hellaby WatkinsLearning from Delayed Rewards PhD thesis, King's College, Cambridge, May 1989.
[27] V. Mnih et al. Playing atari with deep reinforcement learning. In: arXiv preprint arXiv:1312.5602 (2013).
[28] T.P. Lillicrap et al. Continuous control with deep reinforcement learning. In: arXiv preprint arXiv:1509.02971 (2015).
[29] D. Silver, et al., Mastering the game of go without human knowledge, Nature 550 (7676) (2017) 354–359.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[30] S. Gu, et al., Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2017, pp. 3389–3396.
[31] G. Upton, I. Cook, A Dictionary of Statistics 3E, Oxford University Press, 2014.
[32] Why machine learning needs semantics not just statistics, Forbes (2019). https://www.forbes.com/sites/kalevleetaru/2019/01/15/why-machine-learning-needs-semantics-not-just-statistics/?sh=28d4f51277b5.
[33] D.A. Cohn, Z. Ghahramani, M.I. Jordan, Active learning with statistical models, J. Artif. Intell. Res. 4 (1996) 129–145.
[34] P. Melville, R.J. Mooney, Diverse ensembles for active learning, in: Proceedings of the Twenty-First International Conference on Machine Learning, Association for Computing Machinery, 2004, p. 74.
[35] S.C.H. Hoi et al. Online learning: a comprehensive survey. In: arXiv preprint arXiv:1802.02871 (2018).
[36] C.W.J. Granger, P. Newbold, Forecasting Economic Time Series, Academic Press, 2014.
[37] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In: arXiv preprint arXiv:1409.1556 (2014).
[38] K. He, et al., Identity mappings in deep residual networks, in: European Conference on Computer Vision, Springer, 2016, pp. 630–645.
[39] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Adv. Neural. Inf. Process Syst. 25 (2012) 1097–1105.
[40] T. Mikolov et al. Efficient estimation of word representations in vector space. In: arXiv preprint arXiv:1301.3781 (2013).
[41] J. Lilleberg, Y. Zhu, Y. Zhang, Support vector machines and word2vec for text classification with semantic features, in: 2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC), IEEE, 2015, pp. 136–140.
[42] S.K. Sienčnik, Adapting word2vec to named entity recognition, in: Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015), 2015, pp. 239–243.
[43] T.H. Nguyen, K. Cho, R. Grishman, Joint event extraction via recurrent neural networks, in: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016, pp. 300–309.
[44] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd acm Sigkdd International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.
[45] Y. Freund, R. Schapire, N. Abe, A short introduction to boosting, J. Jpn. Soc. Artif. Intell. 14 (771-780) (1999) 1612.
[46] I.K. Fodor. A survey of dimension reduction techniques. Tech. rep. Lawrence Livermore National Lab., CA, 2002.
[47] I.T. Jolliffe, Principal components in regression analysis, Principal Component Analysis, Springer, 1986, pp. 129–155.
[48] G. Golub, W. Kahan, Calculating the singular values and pseudo-inverse of a matrix, SIAM J. Numer. Anal. 2 (2) (1965) 205–224.
[49] Y. Dong, E. Spinei, A. Karpatne, A feasibility study to use machine learning as an inversion algorithm for aerosol profile and property retrieval from multi-axis differential absorption spectroscopy measurements, Atmos. Meas. Tech. 13 (10) (2020) 5537–5550.
[50] N. Kalchbrenner, P. Blunsom, Recurrent continuous translation models, in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 2013, pp. 1700–1709.
[51] S. Albawi, T.A. Mohammed, S. Al-Zawi, Understanding of a convolutional neural network, in: 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1–6, doi:10.1109/ICEngTechnol.2017.8308186.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[52] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: F. Pereira, et al. (Eds.), Advances in Neural Information Processing Systems, 25, Curran Associates, Inc., 2012, pp. 1097–1105. https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.
[53] Y. LeCun. The MNIST Database of Handwritten Digits. In: http://yann.lecun.com/exdb/mnist/ (1998).
[54] Y. LeCun, et al., Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[55] J. Deng, et al., Imagenet: a large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009, pp. 248–255.
[56] G.A. Miller, et al., Introduction to WordNet: an on-line lexical database, Int. J. Lexicogr. 3 (4) (1990) 235–244.
[57] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In: arXiv preprint arXiv:1409.1556 (2014).
[58] G.E. Hinton et al. Improving neural networks by preventing co-adaptation of feature detectors. In: arXiv preprint arXiv:1207.0580 (2012).
[59] M. Edwards and X. Xie. Graph based convolutional neural network. In: arXiv preprint arXiv:1609.08965 (2016).
[60] W.W. Zachary, An information flow model for conflict and fission in small groups, J. Anthropol. Res. 33 (4) (1977) 452–473.
[61] T.N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In: arXiv preprint arXiv:1609.02907 (2016).
[62] B. Shakibajahromi, et al., HyNet: 3D Segmentation Using Hybrid Graph Networks, in: 2021 International Conference on 3D Vision (3DV), IEEE, 2021, pp. 805–814.
[63] J. Pennington, R. Socher, C.D. Manning, Glove: global vectors for word representation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532–1543.
[64] P. Bojanowski, et al., Enriching word vectors with subword information, Trans. Assoc. Comput. Linguist. 5 (2017) 135–146.
[65] K. Cho et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: arXiv preprint arXiv:1406.1078 (2014).
[66] R.C. Staudemeyer and E.R. Morris. Understanding LSTM–a tutorial into Long Short-Term Memory Recurrent Neural Networks. In: arXiv preprint arXiv:1909.09586 (2019).
[67] A. Vaswani et al. Attention is all you need. In: arXiv preprint arXiv:1706.03762 (2017).
[68] H. Xie, et al., Deep learning enabled semantic communication systems, IEEE Trans. Signal Process. 69 (2021) 2663–2675.
[69] E. Hazan. Lecture notes: optimization for machine learning. In: arXiv preprint arXiv:1909.03550 (2019).
[70] N.V. Sahinidis, BARON: a general purpose global optimization software package, J. Global Optim. 8 (2) (1996) 201–205.
[71] S. Arora, S. Singh, Butterfly optimization algorithm: a novel approach for global optimization, Soft Comput. 23 (3) (2019) 715–734.
[72] S. Kaur, et al., Tunicate swarm algorithm: a new bio-inspired based metaheuristic paradigm for global optimization, Eng. Appl. Artif. Intell. 90 (2020) 103541.
[73] J. Bruna, S. Mallat, Invariant scattering convolution networks, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1872–1886.
[74] T.-H. Chan, et al., PCANet: A simple deep learning baseline for image classification? IEEE Trans. Image Process. 24 (12) (2015) 5017–5032.
[75] J. Mairal, et al., Convolutional kernel networks, Advances in Neural Information Processing Systems, 27, Curran Associates, Inc., 2014, pp. 2627–2635.
[76] A. Daniely, R. Frostig, Y. Singer, Toward deeper understanding of neural networks: the power of initialization and a dual view on expressivity, Adv. Neural. Inf. Process Syst. 29 (2016) 2253–2261.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[77] Y. Zhang, P. Liang, M.J. Wainwright, Convexified convolutional neural networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 4044–4053.
[78] A. Prabhu, G. Varma, A. Namboodiri, Deep expander networks: efficient deep networks from graph theory, in: Proceedings of the European Conference on Computer Vision (ECCV), Springer International Publishing, 2018, pp. 20–35.
[79] A.W.E. McDonald, A. Shokoufandeh, Sparse superregular networks, in: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), IEEE, 2019, pp. 1764–1770.
[80] J. You, et al., Graph convolutional policy network for goaldirected molecular graph generation, Adv. Neural. Inf. Process Syst. 31 (2018).
[81] M. Simonovsky, N. Komodakis, Graphvae: towards generation of small graphs using variational autoencoders, in: International Conference on Artificial Neural Networks, Springer, 2018, pp. 412–422.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

CHAPTER TWO

Learning first-principles knowledge from data

Jaewook Lee[a], Weike Sun[b], Jay H. Lee[a] and Richard D. Braatz[b]
[a]Korea Advanced Institute of Science and Technology, Daejeon, South Korea
[b]Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA, United States

Acronyms

ACF	autocorrelation function
ANN	artificial neural network
BIC	Bayesian information criterion
BLUE	best linear unbiased estimator
CASH	combined algorithm selection and hyperparameter optimization
(D)ALVEN	(dynamic) algebraic learning via elastic net
EN	elastic net
GRU	gate recurrent unit
LASSO	least absolute shrinkage and selection operation
LSTM	long short-term memory
NAS	neural architecture search
OLS	ordinary least squares
RNN	recurrent neural network
SPA	smart process data analytics
SVM	support vector machine
SVR	support vector regression
TLS	total least squares
VIF	variation inflation factor

2.1 Background

As data continue to accumulate in modern society at an exponential rate, significant efforts are being directed toward the development of better methods for analyzing that data in various application fields. With the rapid increases in computing power, including storage and memory, even quite large datasets can be stored and handled freely (for further discussion on big data within the context of chemical engineering data, see [46]). Whether the datasets are large or small, of significant interest is the automated search for meaningful relationships within the data, which is one of the primary objectives of *machine learning*. Another primary objective in machine learning

Artificial Intelligence in Manufacturing: Concepts and Methods.
DOI: https://doi.org/10.1016/B978-0-323-99134-6.00005-0
Copyright © 2024 Elsevier Inc.
All rights are reserved, including those for text and data mining, AI training, and similar technologies.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY
Copyright Elsevier 2024

is the development of algorithms to employ these learned models to optimal decision making.

A model is referred to as being *black box* (aka data-driven) when physicochemical principles are not used in the building of the model. Most algorithms for the construction of black-box models are designed for the objective of optimizing the prediction accuracy of the model rather than building a model that can be *interpretable*, that is, provides insight into the mechanisms of the underlying process. This book chapter primary describes the identification of the latter type of data-driven model, in which the goal is to learn first-principles knowledge from data, where *first-principles* refers to chemical, physical, and/or biological phenomena, including (bio)chemical network stoichiometry and kinetics, constitutive relations (e.g., Fick's, Fourier's, and Newton's laws), and mass, energy, and momentum conservation equations. This task is distinct from *white-box* modeling, which is the building of models when the first-principles knowledge is already known and is used to build the model, and *grey-box* modeling, in which a model is constructed from partially known first-principles knowledge and data-driven modeling. The goal of the modeling objective in this chapter is quite different, in that the purpose is to develop data-driven model identification approaches to *learn* the first-principles knowledge from data. For this objective, data are the only inputs to the model identification procedure, and first-principles knowledge is an *output* rather than an *input* to the procedure.

While learning first-principles knowledge is widely studied in the machine-learning community, the focus of this chapter will be on methods applicable to *process* data. Process data have very different characteristics than data for most other applications, which has motivated the use of the term *process data analytics* which has been increasingly used in the chemical engineering community. First, process data typically have a much higher cost than data for most other applications. Significant cost and time are usually required to carry out the experiments needed to generate process data suitable for modeling purposes compared to, for example, Google searches, chess matches, and images of puppies. As such, in process applications, often the data analytics problem is more about how to extract information from *small* rather than *large* quantities of data. Second, when large process datasets are available, such as years of archived manufacturing data, the data have imperfections such as unknown disturbances and sensor biases, and relatively low signal-to-noise ratio. Process data typically have one or more of three characteristics: nonlinearity, multicollinearity, and dynamics. Depending on

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

the situation in which learning is performed, all three characteristics may not be reflected or only some may be reflected [1].

This chapter describes methods for learning first-principles knowledge that are applicable to process data with these characteristics, including ways to automatically transform and select among transformed input variables (Section 2.2) and automatically select appropriate regression methods and hyperparameters (Section 2.3). Section 2.2 includes a background in linear and nonlinear regression needed to understand methods for learning first-principles knowledge from process data, as well as to place these methods in perspective to popular alternative methods in the literature that do not attempt to learn first-principles knowledge. Section 2.3 includes a discussion of the relationship between methods for learning first-principles knowledge to automated machine learning methods, which are also known in the literature as *AutoML*. Section 2.4 summarizes the content and main points made in the chapter.

2.2 Approaches to analyze manufacturing data

This section is a compact self-contained description of data-driven modeling methods that starts from basic linear regression and then successively generalizes to address input–output relationships that are nonlinear, are multicollinear, and/or possess dynamics. The data analytics method should be selected based on the characteristics of the data. For static data, relatively simple models can be used; for dynamic data, a more sophisticated model must be constructed considering additional factors such as the size of the time lag between variables and how far in the past to include inputs and outputs when predicting future outputs.

2.2.1 Static data approaches

Models for static relationships are categorized as being linear or nonlinear regression, depending on the dependency of the model on the fitted parameters.

2.2.1.1 Linear regression

Linear regression describes a linear correlation between the dependent variable y and the independent variable X. Constructing models with one independent variable is *simple linear regression*, and models with two or more

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

independent variables are referred to as *multiple linear regression*. The linear model structure is

$$y = X\beta + \varepsilon$$

where

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix},$$

$$X = \begin{pmatrix} x_1^{\mathrm{T}} \\ x_2^{\mathrm{T}} \\ \vdots \\ x_n^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix},$$

$$\beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix},$$

$$\varepsilon = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix},$$

x_i is a vector of input variables associated with scalar output y_i, y is the vector of outputs from n experiments, X is the data matrix (aka input data matrix), β is the vector of p scalar regression coefficients, ε_i is the difference between the linear model and output for the ith experiment, and ε is typically referred to as the *error vector*. The objective of linear regression is to minimize some norm of the error vector, either directly or indirectly.

Multiple techniques are available for finding the best value for β from data, which can largely be categorized as being least-squares or maximum-likelihood estimation. The two types of methods are closely related and give the same results for some choices of assumptions on the nature of the errors. Least-squares methods, which form the basis for most data analytics algorithms, minimize the sum-of-squared errors $\sum_i w_i\big(y_i - f(x_i, \beta)\big)^2$ where the *weights* w_i are positive scalars. This equation applies to both linear and nonlinear regression, that is, whether the function f is linear or nonlinear in

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

the model parameters. Readers interested in maximum-likelihood estimation are referred to DeSarbo and Cron [2].

Ordinary least squares (OLS) refer to the case where f is linear in β and the weights are equal, e.g., $w_i = 1$. The analytical solution for the best-fit β is

$$\hat{\beta} = \left(X^{\mathrm{T}}X\right)^{-1}X^{\mathrm{T}}y = \left(\sum x_i x_i^{\mathrm{T}}\right)^{-1}\sum x_i y_i$$

provided that the inverse of the matrix $X^{\mathrm{T}}X$ is well-defined. Three classical assumptions are that

- Each independent variable x_i is deterministic
- $\mathrm{E}(\varepsilon_i) = 0$ for all i
- $\mathrm{E}(\varepsilon_i\varepsilon_j) = 0$ for all $i \neq j$ and $\mathrm{E}\left(\varepsilon_i^2\right) = \sigma^2$

where $\mathrm{E}(\cdot)$ refers to the expected value. Under these assumptions, the Gauss-Markov Theorem states that the parameter set selected by OLS is the optimal parameter set for the linear regression model. More precisely, the OLS is the best linear unbiased estimator (BLUE) [3].

Alternative methods such as generalized least squares (GLS), percentage least squares (PLS), and total least squares (TLS) have been developed that have optimality properties for process data that do not satisfy the above assumptions. These methods can take longer to calculate due to having optimization objectives that are more complicated than OLS but perform better in certain situations, such as for input data that are stochastic (have associated noise) which is addressed by TLS. In the latter situation for the case in which the input and output are vectors, a suitable parameter estimation objective is

$$\arg\min_{B} \; ||[E \;\; F]||_F$$

where $\|\cdot\|_F$ is the Frobenius norm and $(X+E)B = Y+F$, that is, E is the error on the independent (aka input) variable, and F is the error on the dependent (aka output) variable. This optimization can be solved by calculating the singular value decomposition (SVD):

$$[X \;\; Y] = [U_X \;\; U_Y]\begin{bmatrix} \Sigma_X & \\ & \Sigma_Y \end{bmatrix}\begin{bmatrix} V_{XX} & V_{XY} \\ V_{YX} & V_{YY} \end{bmatrix}^{\mathrm{T}}$$

where the input and output orthogonal matrices U and V have been partitioned to be compatible with the dimensions of X and Y. After some matrix manipulation [4], the best-fit parameter vector is given by

$$B = -V_{XY}V_{YY}^{-1}.$$

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

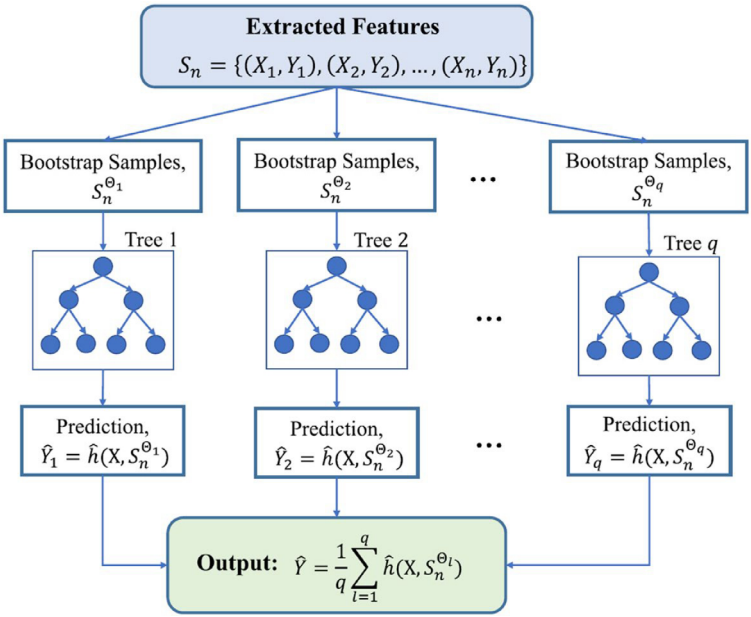


Figure 2.1 Random Forest regression [45].

2.2.1.2 Nonlinear regression without interpretability: black-box models

This section summarizes two nonlinear regression methods used to construct black-box models—random forest regression [5] and support vector regression [6]—that are popular in the machine learning community and have been applied to process data. The most popular nonlinear black-box modeling method—artificial neural networks—are discussed in a later section within the context of dynamic models.

Random forest (RF) is an ensemble algorithm that uses data to train a large number of individual decision trees whose predicted results are averaged to reach a prediction (Fig. 2.1). Each decision tree is constructed from randomly selected data by a procedure known as *bagging* [7]. Bagging repeatedly selects a random sample with the replacement of the training set and fits trees to these samples. After training of all of the decision trees, the prediction is the average taken over all of the decision trees. A desirable feature of a random forest is the ability to handle textual datasets. Some disadvantages are (1) RF models are not interpretable, (2) poor prediction accuracy for sparse data of high dimensionality, and (3) the input–output behavior produces flat regions over most of the data range, which is physically unrealistic for most process applications and breaks numerical optimization

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

algorithms that incorporate the models (e.g., see discussion by Sun and Braatz [1]).

Another popular nonlinear regression method is support vector regression (SVR) which builds on the data classification method known as a support vector machine (SVM). The SVM and SVR are based on

- Hyperplanes, which are planes that determine boundaries for the prediction of the output. The hyperplanes determine the range of output when new input is provided.
- Support vectors, which are points that define the positions of the hyperplanes. These vectors are closest to the hyperplanes, compared to other vectors in the dataset.
- A kernel, which is a set of functions to transform input data to the required form. The kernel is when the hyperplane needs to be constructed in a higher dimensional space. In SVM, the kernel function is needed to make nonlinear boundaries between classes. In SVR, a kernel is needed to make the nonlinear regression model. Linear, nonlinear, polynomial, radial basis function (RBF), and sigmoid functions are widely used kernels. The type of kernel is selected when considering the characteristics of the dataset.

Basically, the SVM is a classifier to make a hyperplane with the maximum margin between distributed classes, which is considered to be the optimal decision boundary. To make a maximal margin for data with n properties, at least $n+1$ support vectors exist. The SVM has the advantage that the optimal decision boundary is only a function of a small number of datapoints, and so is not biased by the distribution of that data, e.g., whether that data satisfy a multivariable normal distribution. A disadvantage is that the SVM is not able to extract or exploit any information from the manner in which the data are distributed, e.g., does not place a stronger weight when a larger number of data points are located at a particular position.

In SVR, in contrast to the SVM, the hyperplane that possesses the largest number of points nearby is the best hyperplane. This objective is also different from most regression models, which aim to minimize least-squares errors between real and predicted points. The SVR only aims to find an optimal hyperplane that includes the given data within the specified threshold and does not exploit all of the information on the distribution of the data.

In summary, black-box nonlinear regression methods such as RF and SVR can model nonlinear relationships between input and output data but have some drawbacks for process applications. Most importantly within

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

the context of this chapter, the models generated by these methods lack interpretability. The next section describes the construction of an interpretable, continuous nonlinear input–output model suitable for process data that combines linear regression with prespecified nonlinear transformations (aka features) and optimization. Before introducing the regression method, methods are described for selecting the best from among a list of features when constructing models.

2.2.1.3 Feature selection for constructing a nonlinear, interpretable regression model

It is desirable in process data analytics for a nonlinear regression model to be *interpretable*. Instead of using the input of process data *as is*, interpretability is achieved by selecting a concise set of features from a set of predefined features. This section describes three categories of methods that have been proposed for selecting concise and meaningful features.

The first category of the feature selection method is based on symbolic regression. Symbolic regression generates a mathematical model constructed from symbolic relationships between the independent and dependent variables in the dataset. The symbolic relationships can be predefined or constructed on the fly such as by using genetic programming [8] which uses mutation, crossover, and evolution to search for a mathematical model that minimizes the regression error. The number of symbolic relationships can be down-selected by cross-validation. Considering too many potential symbolic relationships in such methods can result in overfitting and quite delicate care should be taken [9].

The second category of feature selection methods is based on mixed-integer optimization. Optimization-based methods select a subset of features via a minimization formulated to restrict the number of features that appears in the final model. One such optimization-based method is the automated learning of algebraic models for optimization (ALAMO) for machine learning [10] which solves

$$\min \|y - Xw\|_2^2 + \lambda \|w\|_q$$

where $q = 0$ and λ is a positive scalar penalty that specifies the sparsity, that is, the number of nonzero elements of the regression vector w. The magnitude of λ balances the accuracy and complexity of the model.

ALAMO was originally invented to derive simple models to match raw data. A detailed flowchart of the model-building algorithm via ALAMO is represented in Fig. 2.2. The model-building step requires finding the

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

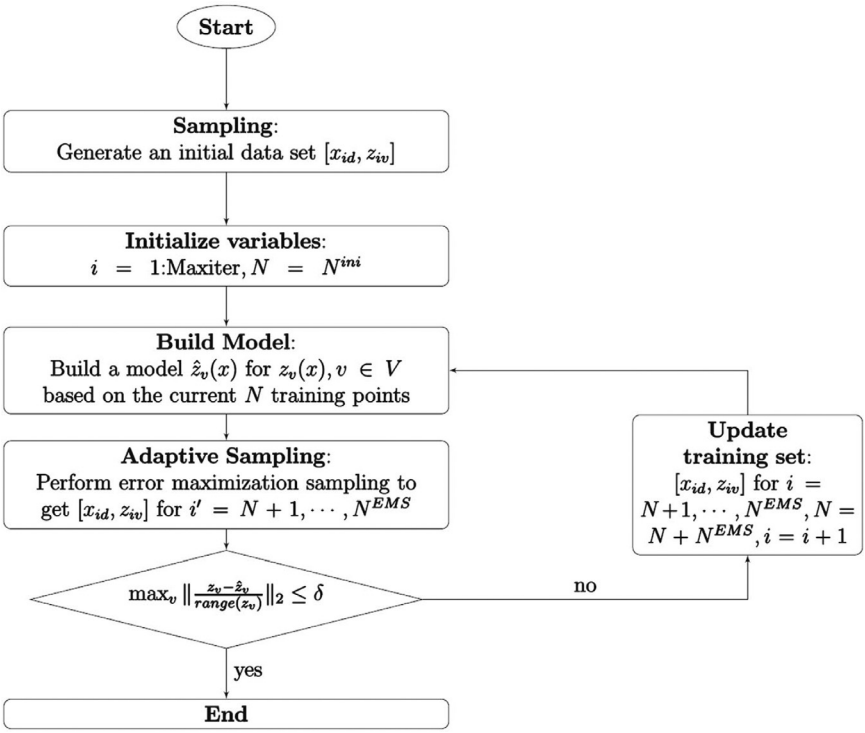


Figure 2.2 Flowchart of model building with automated learning of algebraic models for optimization (ALAMO) [10].

numerical solution of an NP-hard optimization that is solved by branch-and-bound methods.

Wilson and Sahinidis [10] used the Bayesian information criterion (BIC) as the fitness metric to balance the bias-variance tradeoff in the process of finding appropriate features and terms of the inputs for building an appropriate linear surrogate model. Constraints on the response variables can be enforced to incorporate first-principles knowledge. The optimization for finding appropriate features is much more computationally expensive than the category of methods described next.

The third category of feature selection methods is based on a convex optimization that enforces sparsity, that is, selects features among a nonlinearly transformed basis for constructing a nonlinear interpretable regression model. One such mathematical formulation is

$$\min \|y - Xw\|_2^2 + \lambda \|w\|_q$$

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

where $q = 1$ and λ is a positive scalar penalty that specifies the sparsity, that is, the number of nonzero elements of the regression vector w. The magnitude of λ balances the accuracy and complexity of the model. This optimization formulation is known as the least absolute shrinkage and selection operator (lasso [11]). The optimization can be reformulated as a convex quadratic program. Lasso has been very widely used for the last 15+ years, in large part due to its computational efficiency which is especially important when the optimization is solved a large number of times for different subsets of the original dataset, to provide some confidence in the performance of the model when applied to data not used in the regression. The lasso optimization with $q = 1$ can be shown to be a convex relaxation of the optimization with $q = 0$ [11]. The value $q = 0$ results in the l_0 pseudonorm, in which case the regression is called *the best subset selection problem*. The l_0 pseudonorm, which is used by ALAMO, results in a mixed-integer program that is nonconvex and NP-hard.

Both $q = 0$ and $q = 1$ have the characteristic that the selection of features can be arbitrarily sensitive to arbitrarily small perturbations in the data, which limits interpretability and motivated the development of the elastic net (EN) [12]. Elastic net solves the optimization.

$$\min \|y - Xw\|_2^2 + \lambda\left(\alpha\|w\|_1^1 + \frac{1-\alpha}{2}\|w\|_2^2\right)$$

where the scalar α, which ranges from 0 to 1, is introduced to balance the l_1 and l_2 penalties. At the maximum value of α of 1, EN becomes equal to lasso. EN selects the features with higher robustness than lasso [12]. EN is able to select appropriate features even when data are scarce and is able to select correlated variables. The l_2 penalty is the term that provides the ability to deal with multicollinearity in the dataset while increasing robustness. EN-based feature selection forms the basis for the algebra learning via elastic net (ALVEN) method for the construction of nonlinear models from data which is described below.

2.2.1.4 Example of interpretable nonlinear regression: ALVEN

This section describes ALVEN, which is an interpretable nonlinear sparse regression method [9]. ALVEN employs features that are prespecified to be able to capture a large proportion of relationships that have been observed in chemical, physical, and biological processes. Then ALVEN employs EN to automatically select nonlinear features that show stable and good prediction performance. ALVEN's model-fitting algorithm consists of the three steps

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

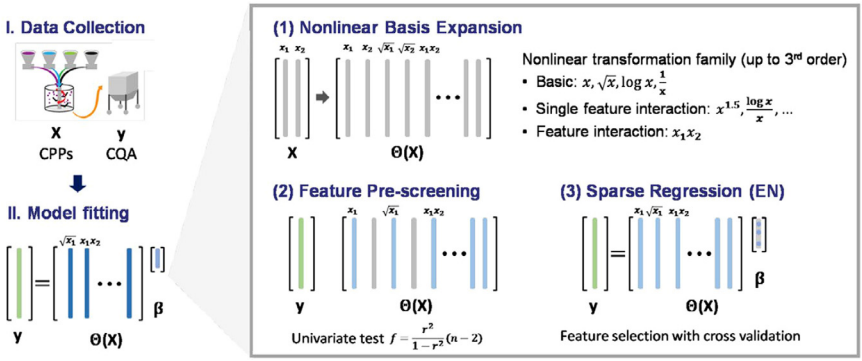


Figure 2.3 Schematic of the algebra learning via elastic net (ALVEN) algorithm [9].

shown in Fig. 2.3: (1) nonlinear transformations of the raw data, (2) feature preselection, and (3) EN with cross-validation. These steps are described in more detail below.

Step 1: With the training data matrix X, nonlinear transformations are applied and the results are stacked to form a candidate features matrix $\Theta(X)$. The nonlinear transformations are constructed from four basic relationships: linear, logarithm [13], square root, and inverse. A large proportion of first-principles relationships can be constructed by a combination of repeated application of these nonlinear relationships and multiplication (aka feature interaction). The degree of complexity d controls the complexity of the candidate features. Most first-principles relationships derived for chemical or biological process data can be represented by $d \leq 3$, with power-law relationships [14] handled by applying ALVEN to the logarithm of the output.

The family of nonlinear transformations for $d = 1$ are x_i, $\sqrt{x_i}$, $\log x_i$, $\frac{1}{x_i}$, $i = 1, \ldots, m_x$ where m_x is the number of input variables.

For $d = 2$, the nonlinear transformation family includes the family for $d = 1$ and (1) the second-order interactions between input variables $x_i x_j$ where $\forall i \neq j$, $i = 1, \ldots, m_x$ and (2) the second-order interactions for each observation, which x_i^2, $x_i^{3/2}$, $(\log x_i)^2$, $\frac{1}{x_i^2}$, $x_i^{-1/2}$, $i = 1, \ldots, m_x$.

For $d = 3$, the nonlinear transformation family includes the transformations for $d = 2$, and (1) the third-order interactions between input variables $x_i x_j x_k$ where $\forall i \neq j \neq k$, $i = 1, \ldots, m_x$; (2) the third-order interactions for each input variable, which include x_i^3, $(\log x_i)^3$, $\frac{1}{x_i^3}$, $x_i^{5/2}$, $\frac{(\log x_i)^2}{x_i}$, $\frac{\log x_i}{\sqrt{x_i}}$, $\frac{\log x_i}{x_i^2}$, $x_i^{-3/2}$, $i = 1, \ldots, m_x$.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

As a result, the matrix $\Theta(X)$ consists of candidate nonlinear transformations of the columns of X.

Step 2: To remove terms of low potential value while reducing the dimensionality among the candidate features from Step 1, a prescreening univariate test is conducted [9]. Several options are available for prescreening. One option is based on the linear correlation coefficient between the response and each transformed feature, $r_i = \text{corr}(y, x_i), \forall x_i \in \Theta(x)$. Then a univariate statistical test is applied for the test statistic calculated from

$$f_i = \frac{r_i^2}{1 - r_i^2}(N - 2).$$

If the P-value is lower than the specified signal level, it is selected as a potentially useful feature.

Another option is to select the features with the highest scoring function values f_i and remove the rest. In this approach, the percentage to remain should be selected. A third option is the elbow method [15]. Briefly explaining the elbow, by calculating and arranging the scoring functions from higher to lower values, the arranged functions are plotted. The approach selects which features to remove based on their position based on the elbow of the plot. This method is applicable to data that show an obvious elbow in the plot.

Step 3: EN is applied using the preselected features from Step 2. Elastic net is chosen for its ability to handle multicollinearity between the candidate features; the l_2-norm is key to producing models that are interpretable, accurate, and robust [12]. The hyperparameters λ and degree of complexity d are chosen by cross-validation. To prevent overfitting, for datasets of limited quantity and quality, typically cross-validation selects d as 1 or 2.

The next section describes approaches for building dynamic models from data.

2.2.2 Dynamic data approaches

2.2.2.1 Recurrent neural networks

Recurrent neural networks (RNN) is a class of artificial neural network (ANN), which uses the internal state of the previous time step as input into the next time step [16]. The internal state of the RNN, called the *hidden state*, possesses a memory of the past. Based on hidden states and new input, the RNN predicts the next output. The RNN is trained by receiving a sequence as input and a sequence as output. The length of input and output sequence vectors could be different.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

In RNN, the hidden and output layers are calculated from

$$h_t = \tanh\left(W_x x_t + W_h h_{t-1} + b_h\right)$$

$$y_t = f\left(W_y h_t + b_y\right)$$

where the matrices W_x, W_y, and W_h are weights and the vectors b_h and b_y are biases that are determined during the regression. The hidden layer computes the current value of the hidden state h_t from the hidden state at the previous time step, h_{t-1}, and the input at the current time step x_t. The output layer computes the current value of the output y_t from the current value of the hidden state. The hyperbolic tangent is called a *neuron* or an *activation function*, with alternative neurons including the sigmoid and the rectified linear unit. An alternative network structure employs a small number of hidden layers, which has been demonstrated to be highly useful in process applications [17,18]. In any case, the weights and biases are computed by gradient descent, which is called *backpropagation* in the neural network literature. Although backpropagation is not guaranteed to find the global minimum of the error function, and rarely does, the method tends to converge to weights and biases that fit the training data when a small number of hidden layers are used.

A substantial literature considers a large number of hidden layers, which are called *deep learning networks*. In such networks, the gradients used in backpropagation approach zero which causes the training procedure to become impractically slow. The problem can be reduced by using the rectified linear unit as the activation function. An alternative RNN structure known as long–short-term memory (LSTM) can model time series data without running into the vanishing gradient problem by incorporating forget gates [16]. LSTM can model long-time delays while retaining the ability of standard RNNs to handle any mix of slow and fast time scales.

In order to calculate long-term memory, the LSTM uses the concept of a *gate* [19]. Each gate performs different roles for processing important data or deleting unimportant data and reflecting those memories to prediction in the future. The network structure is much more complicated than standard RNNs, and requires the definition of more types of weights and biases—namely W_{xi}, W_{xg}, W_{xf}, W_{xo}, W_{hi}, W_{hg}, W_{hf}, W_{ho} and b_i, b_g, b_f, b_o—which are defined below within the context of each type of gate.

1. Input gate

In the input gate, important information in the time step t is being remembered. The input gate's activation vector i_t and cell input activation

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

vector g_t are given by

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$

$$g_t = \tanh\left(W_{xg}x_t + W_{hg}h_{t-1} + b_g\right)$$

2. Forget gate

In the forget gate, unimportant data are being forgotten. Its inputs are the same as the input gate, however, at this time, if f_t becomes near 0, past information is almost being forgotten, otherwise, past information is hardly forgotten. The forget gate's activation vector is given by

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + b_f\right)$$

3. Cell state

The cell state c_t combines information from the input and forget gates,

$$c_t = f_t \circ c_{t-1} + i_t \circ g_t$$

The input to the cell state from the input gate, which represents how much to reflect current information, is calculated by element-by-element multiplication of i_t and g_t. Element-by-element multiplication is known as the *Hadamard product*, which is represented by $\circ$. Meanwhile, the immediate past value of the cell state, c_{t-1}, is modulated by the value from the forget gate.

4. Output gate and hidden state

Lastly, the output gate calculates the hidden state vector h_t from the immediate past hidden state vector h_{t-1}, the input vector x_t, and the cell state c_t,

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

$$h_t = o_t \circ \tanh(c_t)$$

where o_t is the output gate's activation vector.

An RNN consisting of LSTM units satisfying the above equations is trained by backpropagation.

The gated recurrent unit (GRU) is similar to the LSTM unit but simpler, with only two gates [20]. Since the number of gates is less, training a network with GRUs is much faster than training a network with LSTM units. In most cases, GRU has shown similar prediction accuracy as LSTM. The

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

backpropagation training of a GRU network is similar to LSTM networks. The GRU is given by

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r)$$

$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z)$$

$$g_t = \tanh\left(W_{hg}(r_t \circ h_{t-1}) + W_{xg}x_t + b_g\right)$$

$$h_t = (1 - z_t) \circ g_t + z_t \circ h_{t-1}$$

RNNs and other dynamic neural network models have been applied in the process industries for decades [21]. More sophisticated RNNs that provide probabilistic output distributions has also been applied in the industry [22], and dense compact networks more amenable to theoretical analysis have been developed [23]. Since computing power keeps increasing, RNNs and related networks converge faster each year, enabling such models to be trained for larger training datasets. However, learning first-principles knowledge from such networks is challenging and remains a topic of research. Moreover, since neural network training does not converge globally due to the complex structure inside the hidden layer, RNNs rarely possess interpretability. The next section describes a method for constructing dynamic data-driven models that are interpretable.

2.2.2.2 DALVEN

ALVEN is directly extended to nonlinear dynamical system identification by dynamic modification on the transformation matrix. The resulting model, dynamic ALVEN (DALVEN), has an interpretable input-output structure while maintaining high modeling accuracy.

DALVEN uses the widely used *time lag shift* method [24] to augment the original input space X to form a new expanded space $\hat{X}$. The mathematical form of DALVEN becomes $y_t = \sum_t w_i \emptyset_i(x_t, \ldots, x_{t-l}, y_{t-1}, \ldots, y_{t-l}) + \varepsilon_t$, where l is the magnitude of the lag shift. The three steps of the DALVEN algorithm are described below.

Step 1: For the nonlinear mapping, DALVEN provides two options. The first option, called the DALVEN-full model, is to augment the original input

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

space X with the previous l observations of the state and output [9]:

$$\hat{X} = \begin{bmatrix} x_t & \cdots & x_{t-l} & y_{t-1} & \cdots & y_{t-l} \\ x_{t+1} & \cdots & x_{t+1-l} & y_t & \cdots & y_{t+1-l} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_N & \cdots & x_{N-l} & y_{N-1} & \cdots & y_{N-l} \end{bmatrix}.$$

When ALVEN is applied to this augmented matrix, the resulting model will apply nonlinearities to both inputs and past outputs and include interactions between inputs and outputs at different time points. The second option is to apply the nonlinearities in ALVEN to transform the inputs to form the matrix:

$$\Theta(\hat{X}) = \begin{bmatrix} \phi(x_t) & \cdots & \phi(x_{t-l}) & y_{t-1} & \cdots & y_{t-l} \\ \phi(x_{t+1}) & \cdots & \phi(x_{t+1-l}) & y_t & \cdots & y_{t+1-l} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \phi(x_N) & \cdots & \phi(x_{N-l}) & y_{N-1} & \cdots & y_{N-l} \end{bmatrix}.$$

This second option allows only nonlinearity and interactions between past inputs variables within the same time point, while the past system outputs are not transformed. By reducing the set of potential nonlinearities, this option generates fewer features that can be robust in some applications.

Step 2: Feature prescreening is applied as in ALVEN. Linear correlations between the output variable and variables in the design space from Step 1 are calculated and fed to a univariate statistical test to remove features that lack statistical significance. The two alternatives (percentage or elbow method) in Step 2 in ALVEN can be used instead to preselect features.

Step 3: EN is used to minimize the one-step-ahead prediction error. The optimal hyperparameters are selected by cross-validation or an information criterion (e.g., Akaike) [25]. If autocorrelation is statistically significant after this step, an autoregressive integrated moving average (ARIMA) model [26] is constructed and added to the DALVEN model to improve prediction accuracy.

DALVEN and RNN-based model identification are at opposite ends of the spectrum in terms of constructing discrete-time nonlinear input–output models. DALVEN extends elastic net and feature engineering, which are approaches from the applied machine learning literature. DALVEN includes feature design based on expressions that commonly arise in chemical and biological systems, includes feature selection to generate a sparse/interpretable model, is formulated in terms of a polynomial-time convex optimization,

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

and implements a quadratic optimization that converges globally. In contrast, the optimization for finding an RNN is nonconvex, and a locally optimal value of the model parameters is found by a gradient descent algorithm.

2.3 Automation of model selection and hyperparameter search

For developing a data-driven model, a common practice is to iteratively search for a model that results in a small model error. These procedures require a lot of human expertise and effort, which has motivated the development of procedures to automate these processes. This section describes such automated procedures, potential pitfalls in their application, and ways to avoid those pitfalls.

2.3.1 Automation for general data: AutoML

The intended objective of automated machine learning (AutoML) is to provide software so that nonexperts in machine learning would be able to build high-quality machine learning models from the preprocessing to the hyperparameter tuning without needing to become experts in machine learning. Two AutoML strategies have been developed. The first strategy is to automate a combination of algorithm selection and hyperparameter optimization (CASH [27]). In this strategy, the objective is to find an optimal algorithm for the given data and maximize the selected algorithm's performance. Many methods have employed the CASH strategy. Some methods find optimal hyperparameters using traditional optimization methods such as grid search and random search, whereas other methods use Bayesian optimization methods such as Gaussian processes [28] and Fast Bayesian Optimization for Data Analytic Pipelines [29] or armed bandit methods [30]. Detailed descriptions of these methods can be found in the references.

The second AutoML strategy considers neural networks as the only allowable algorithm and to automate the neural architecture search (NAS [31]). NAS methods can be categorized in terms of:

- Search space: the set of ANN architectures that are searched to find the optimal architecture
- Search strategy: the strategy used to explore the search space
- Performance estimation strategy: the evaluation metric used to select the optimal ANN among the possible ANN architectures

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Neural architecture search methods has been developed that incorporate reinforcement learning [32], convolutional networks [33], zeroing out of networks during evaluation [34,35]. Details of each algorithms can be found in the references.

AutoML packages include AutoWEKA [36], Auto-Sklearn [37], Auto-Keras [38], and TPOT [39]. As an illustrative example, Auto-Sklearn combines meta-learning, Bayesian optimization, and ensemble modeling to select an optimal algorithm and tune hyperparameters. The meta-learning feeds into the machine learning framework consisting of a data preprocessor, a feature preprocessor, and a classifier in series in feedback with a Bayesian optimizer. The output of the machine learning framework goes into building an ensemble model [37].

The AutoML literature has been focused on automation rather than interpretability, and has not been focused on process data. The main focus of this chapter is on learning first-principles knowledge, for which interpretability is a key concern, and on process data. As such, the AutoML methods that dominate the literature do not meet our objective of constructing an interpretable, nonlinear regression model based on process data. The next subsection describes an AutoML method—smart process analytics (SPA)—which is an automated machine learning method developed for process data.

2.3.2 Automation for manufacturing data: smart process data analytics

Smart process analytics [1] and AutoML software share the same objective of providing a tool that enables individuals who are not experts in machine learning to build predictive models correctly and efficiently, and ideally, the software should involve minimum decision-making procedures to automate the data analytics procedure.

Smart process analytics has two major differences from AutoML. First of all, SPA is focused on process data and is not designed to handle many of the datasets that are the focus of AutoML, such as data on images or associated with playing games such as chess, checkers, and Go. Process data have characteristics that scientists and engineers have some familiarity with, and attempts have been made in the past to build models from relatively small datasets by using human knowledge during model construction.

The second difference is that SPA does not consider machine learning methods designed for handling datasets that are infinite or nearly infinite in quantity. As such, SPA needs to take more care when selecting the best

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

model among candidate models, to avoid overfitting. As such, SPA is based on the premise that there is not huge quantities of data, and the model itself is intended to be as light as possible.

The first step in SPA is to quantify the characteristics of the data. First, the SPA determines the nonlinearity between the predictor and input variables. If the nonlinearity is not statistically significant, then a linear model can be constructed and the computational load is much smaller; if the nonlinearity is statistically significant, then a linear model is not sufficient and a no-linear model such as a natural network should be constructed.

Smart process analytics uses Pearson's correlation coefficient,

$$r_{XY} = \frac{\sum_i^n \left(X_i - \bar{X}\right)\left(Y_i - \bar{Y}\right)}{\sqrt{\sum_i^n \left(X_i - \bar{X}\right)^2}\sqrt{\sum_i^n \left(Y_i - \bar{Y}\right)^2}}$$

for an initial test of strong linearity. The Pearson correlation coefficient is measured with two variables and is distributed from –1 to 1. The closer its value is to –1 or 1, the stronger the negative or positive correlation. If the absolute value of Pearson's correlation coefficient is large enough, then only linear models should be considered.

To assess nonlinear correlation, SPA employs two methods, a quadratic test and maximal correlation analysis. The quadratic test compares the null hypothesis H_0: $y = w_1 x + w_0$, that two variables x and y are linear, with the alternative hypothesis $H_1 : y = w_2 x_2^2 + w_1 x + w_0$, that y is related to x by a quadratic function. The null hypothesis is tested by the statistic $F = \frac{SSE(0) - SSE(a)}{df_0 - df_a} \div \frac{SSE(a)}{df_a}$ where df is the number of degrees of freedom. The nonlinearity of the given data can be quantified by the P-value of the hypothesis test. In maximum correlation analysis, nonlinearity is quantified by the maximum correlation [40]:

$$\sup_{\theta,\phi} \ \mathrm{corr}(\theta(x),\ \phi(y))$$

where corr is Pearson's correlation coefficient. Interestingly, a very computationally efficient algorithm is available to determine nonlinearities of the two variables that maximize the correlation.

Each method has a limitation for quantifying nonlinearity from data. (1) The quadratic test considers only the quadratic model, and (2) the maximum correlation analysis has low statistical strength the quantity of data is very low [41]. Accordingly, SPA judges nonlinearity by combining the two methods into a matrix comparing with the linear correlation matrix.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Having both the linear correlation and the maximal correlation close to zero or one means that the variables are uncorrelated or linearly correlated, respectively. Having the linear correlation close to zero and the maximal correlation closer to 1 indicates the variables are not linearly correlated but are nonlinearly correlated.

After the nonlinearity assessment, SPA assesses the multicollinearity of the data. Multicollinearity refers to where the relationship between input variables is not independent and the dependence is high. Multicollinearity occurs for most process datasets. SPA assesses multicollinearity by using the variable inflation factor (VIF). The VIF for the kth input variable is

$$VIF_k = \frac{1}{1 - R_k^2}$$

and is considered significant for VIF greater than 10.

Finally, SPA assesses the dynamics of the data. In a dynamic model, the value at the previous point in time influences the value at the current point in time. SPA uses the autocorrelation function,

$$\mathrm{ACF} = \mathrm{corr}(\varepsilon_t,\ \varepsilon_{t-l})$$

to assess the serial correlation, where l is the lag variable. If the autocorrelation function (ACF) value is statistically significant, then SPA uses a dynamic model. For selecting a dynamic model, if a dataset only has the characteristic of dynamics, then canonical variate analysis (CVA), state-space autoregressive exogenous model (SSARX), and multivariable output error state space (MOESP) are the most suitable methods. The readers are directed to references for detailed explanations of those methods [42–44].

After assessing the extent of nonlinearity, multicollinearity, and dynamics in the data, SPA selects the potential data analytics and machine learning methods to be used for model construction. The decision tree for model selection used by SPA can be represented by the diagram in Fig. 2.4. For example, if the data were assessed as having significant nonlinearity but not significant multicollinearity or dynamics, then SPA would apply ALVEN, SVR, or RF—using cross-validation to select between the methods. If the data were assessed as having significant nonlinearity and dynamics but not significant multicollinearity, then SPA would apply cross-validation to select between the best models constructed by recursive neural networks and DALVEN. DALVEN has the advantage that the model would be interpretable, as the method automatically drops inputs that do not significantly affect the model output.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

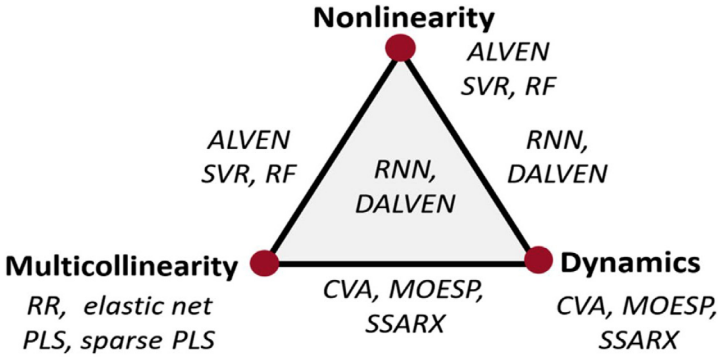


Figure 2.4 Data analytics triangle [1]. The acronyms are specified in the reference.

For data assessed to exhibit multicollinearity but negligible nonlinearity and dynamics, ridge regression (RR), partial least squares (PLS), elastic net, and sparse PLS are suitable methods, where *sparse PLS* is combined lasso and partial least squares to drop inputs that do not significantly affect the model output. If the objective is for the model to be interpretable, then elastic net and sparse PLS would be used in the model construction, and ridge regression and PLS would be dropped since those models include all inputs in the resulting model.

2.4 Conclusion

This chapter described various methods for data-driven modeling and for the determination of first-principles knowledge from data, and how to automatically select the data analytics and machine learning methods based on characteristics of the data. Included in the discussion was how to select among predefined features that define potential nonlinear relationships between the model inputs and outputs.

Three characteristics of process data were discussed: nonlinearity, multicollinearity, and dynamics. Models were also characterized, in terms of being either linear or nonlinear functions of the model parameters. Most of the attention to regression were on nonlinear methods, including RF, SVR, RNN, LSTM, and GRU for static models and RNN, LSTM, and GRU for dynamic data. An algebraic learning method for the construction of interpretable models called ALVEN was described. This method specializes in processing data by limiting the features to forms observed in first-principles laws such as bilinear, quadratic, and power-law relationships. An extension to dynamic

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

data, known as DALVEN, was described, for which two options were described. One option considers nonlinear transformations of both input variables and output variables, and the second option considers nonlinear transformation of only the input variables.

After looking at these data-driven methods and models, we discussed the automatic selection of a method, model, hyperparameter, and architecture by assessing the characteristics of the data. AutoML is an approach of combining automatic selection algorithms developed by many researchers in the machine learning field. Limitations in the general AutoML methods were described and addressed by the SPA algorithm. SPA analyzes three characteristics of the data using various analysis methods, uses this information to limit the number of algorithms that can be applied, and determines the algorithm to be applied. Data-driven modeling algorithms that can be selected include RNN, D(ALVEN), SVR, and RF. Therefore, it is judged that research can be conducted to select a model hyperparameter to suit the characteristics of process data, and further design a new data-based model, as AutoML did.

Currently, SPA is already able to describe many relationships that arise in process data, including Newton's law ($F = ma$), elementary chemical kinetics ($r = k_0 C_A C_B^2$), and convective heat and mass transfer expressions (e.g., $\mathrm{Nu} = 0.6\mathrm{Re}^{1/2}\mathrm{Sc}^{1/3}$). Looking forward, a promising research direction is to extend SPA or alternative algebraic learning frameworks to handle additional first-principles expressions that arise in process systems.

References

[1] W. Sun, R.D. Braatz, Smart process analytics for predictive modeling, Comput. Chem. Eng. 144 (2021) 107134.
[2] W.S. DeSarbo, W.L. Cron, A maximum likelihood methodology for clusterwise linear regression, J. Classification 5 (2) (1988) 249–282.
[3] F. McElroy, A necessary and sufficient condition that ordinary least-squares estimators be best linear unbiased, J. Am. Statist. Assoc. 62 (320) (1967) 1302–1304.
[4] H.P. Gavin, Total Least Squares. http://people.duke.edu/~hpgavin/SystemID/CourseNotes/TotalLeastSquares.pdf (2017).
[5] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[6] V. Vapnik, O. Chapelle, Bounds on error expectation for support vector machines, Neural. Comput. 12 (9) (2000) 2013–2036.
[7] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[8] J.H. Holland, Genetic algorithms, Sci. Am. 267 (1) (1992) 66–73.
[9] W. Sun, R.D. Braatz, ALVEN: algebraic learning via elastic net for static and dynamic nonlinear model identification, Comput. Chem. Eng. 143 (2020) 107103.
[10] Z.T. Wilson, N.V. Sahinidis, The ALAMO approach to machine learning, Comput. Chem. Eng. 106 (2017) 785–795.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[11] R. Tibshirani, Regression shrinkage and selection via the lasso, J. R. Stat. Soc., B: Stat. Methodol. 58 (1) (1996) 267–288.
[12] H. Zou, T. Hastie, Regularization and variable selection via the elastic net, J. R. Stat. Soc., B: Stat. Methodol. 67 (2) (2005) 301–320.
[13] K. Pielichowski, J. Njuguna, Thermal Degradation of Polymeric Materials, iSmithers Rapra Publishing, 2005.
[14] M. Milosevic, Internal Reflection and ATR Spectroscopy, John Wiley & Sons, 2012.
[15] R.L. Thorndike, Who belongs in the family? Psychometrika 18 (1953) 267–276.
[16] F.A. Gers, N.N. Schraudolph, J. Schmidhuber, Learning precise timing with LSTM recurrent networks, J. Mach. Learn. Res. 3 (2002) 115–143.
[17] S.J. Qin, Recursive PLS algorithms for adaptive data modeling, Comput. Chem. Eng. 22 (4-5) (1998) 503–514.
[18] S.J. Qin, T.J. McAvoy, Nonlinear PLS modeling using neural networks, Comput. Chem. Eng. 16 (4) (1992) 379–391.
[19] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural. Comput. 9 (8) (1997) 1735–1780.
[20] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation, In: arXiv preprint arXiv:1406.1078 (2014).
[21] S. Piche, J. Keeler, G. Martin, G. Boe, D. Johnson, M. Gerules, Neural network based model predictive control, Advances in Neural Information Processing Systems, MIT Press, Denver, Coloardo, 1999.
[22] W. Sun, A.R. Paiva, P. Xu, A. Sundaram, R.D. Braatz, Fault detection and identification using Bayesian recurrent neural networks, Comput. Chem. Eng. 141 (2020) 106991.
[23] P.R. Jeon, M.S. Hong, R.D. Braatz, Compact neural network modeling of nonlinear dynamical systems via the standard nonlinear operator form, Comput. Chem. Eng. 159 (2022) 107674.
[24] W. Ku, R.H. Storer, C. Georgakis, Disturbance detection and isolation by dynamic principal component analysis, Chemom. Intell. Lab. Syst. 30 (1) (1995) 179–196.
[25] H. Akaike, A new look at the statistical model identification, IEEE Trans. Autom. Control. 19 (6) (1974) 716–723.
[26] C. Chatfield, The Analysis of Time Series: An Introduction, Chapman and Hall/CRC, 2003.
[27] C. Thornton, F. Hutter, H.H. Hoos, K. Leyton-Brown, Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms, in: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, Association for Computing Machinery, 2013.
[28] J. Snoek, H. Larochelle, R.P. Adams, Practical bayesian optimization of machine learning algorithms, Advances in Neural Information Processing Systems, Curran Associates, Lake Tahoe, Nevada, 2012.
[29] Y. Zhang, M.T. Bahadori, H. Su, J. Sun, FLASH: fast Bayesian optimization for data analytic pipelines, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, Association for Computing Machinery, 2016.
[30] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, A. Talwalkar, Hyperband: a novel bandit-based approach to hyperparameter optimization, J. Mach. Learn. Res. 18 (1) (2017) 6765–6816.
[31] T. Elsken, J.H. Metzen, F. Hutter, Neural architecture search: a survey, J. Mach. Learn. Res. 20 (1) (2019) 1997–2017.
[32] B. Zoph, Q.V. Le. Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578 (2016).

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[33] B. Zoph, V. Vasudevan, J. Shlens, Q.V. Le, Learning transferable architectures for scalable image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, IEEE, 2018.
[34] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, Q. Le, Understanding and simplifying one-shot architecture search, in: International Conference on Machine Learning Proceedings of Machine Learning Research, Stockholm, Sweden, 2018.
[35] H. Cai, L. Zhu, S. Han. Proxylessnas: direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332 (2018).
[36] L. Kotthoff, C. Thornton, H.H. Hoos, F. Hutter, K. Leyton-Brown, Auto-WEKA: automatic model selection and hyperparameter optimization in WEKA, Automated Machine Learning, Springer, Cham, 2019, pp. 81–95.
[37] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, F. Hutter, Efficient and robust automated machine learning, Advances in Neural Information Processing Systems, Curran Associates, Montréal, Canada, 2015.
[38] H. Jin, Q. Song, X. Hu, Auto-keras: An efficient neural architecture search system, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Association for Computing Machinery, New York, 2019.
[39] T.T. Le, W. Fu, J.H. Moore, Scaling tree-based automated machine learning to biomedical big data with a feature set selector, Bioinformatics 36 (1) (2020) 250–256.
[40] A. Rényi, On measures of dependence, Acta. Math. Hung. 10 (3-4) (1959) 441–451.
[41] W.K. Härdle, L. Simar, Canonical correlation analysis, Applied Multivariate Statistical Analysis, Springer, 2015, pp. 443–454.
[42] D. Astuti, B.N. Ruchjana, Generalized space time autoregressive with exogenous variable model and its application, J. Phys. Conf. Ser. 893 (1) (2017) 012038.
[43] R.B. Darlington, S.L. Weinberg, H.J. Walberg, Canonical variate analysis and related techniques, Rev. Educat. Res. 43 (4) (1973) 433–454.
[44] F. Ding, X. Zhang, L. Xu, The innovation algorithms for multivariable state-space models, Int. J. Adapt. Control Signal Process. 33 (11) (2019) 1601–1618.
[45] Y. Li, C. Zou, M. Berecibar, E. Nanini-Maury, J.C.-W. Chan, P. Van den Bossche,..., N. Omar, Random forest regression for online capacity estimation of lithium-ion batteries, Appl. Energy 232 (2018) 197–210.
[46] M.S. Reis, R.D. Braatz, L.H. Chiang, Challenges and future research directions, Chem. Eng. Prog. 112 (3) (2016) 46–50.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

CHAPTER THREE

Convolutional neural networks: Basic concepts and applications in manufacturing

Shengli Jiang[a], Shiyi Qin[a], Joshua L. Pulsipher[b] and Victor M. Zavala[a]
[a]Department of Chemical and Biological Engineering, University of Wisconsin-Madison, Madison, WI, United States
[b]Department of Chemical Engineering, University of Waterloo, Waterloo, ON, Canada

3.1 Introduction

Manufacturing is witnessing increasing use of real-time sensing and instrumentation technologies that generate data in the form of images/video (e.g., infrared and thermal), vibration/audio, and other complex data forms such as chemical spectra and geometrical structures (e.g., 3D printed objects, synthesized molecules, and crystals). Also, increasingly automation systems that aim to exploit such data to make decisions (e.g., optimize production and detect anomalies) are being used in manufacturing. Moreover, modern automation systems are being designed to take instructions/targets in the form of complex data objects (e.g., voice, text, chemical spectra, and molecular structures).

Modern automation systems used in manufacturing embed highly sophisticated computing workflows that use tools from data science and machine learning (ML) to extract and interpret actionable information from complex data streams. Such workflows resemble those used in other advanced technologies such as autonomous vehicles (e.g., aerial, terrestrial, and aquatic) and robotics. Moreover, such workflows begin to resemble human systems in which visual, auditive, tactile, and olfactory signals (data) are routinely used to make decisions. For instance, the human olfactory system generates signals when exposed to specific chemical structures, and such signals are processed and interpreted by the brain to detect anomalies. Similarly, the human visual system generates interpretable signals when exposed to objects with specific geometrical and color features and our auditory system generates interpretable signals when exposed with specific

Artificial Intelligence in Manufacturing: Concepts and Methods.
DOI: https://doi.org/10.1016/B978-0-323-99134-6.00007-4
Copyright © 2024 Elsevier Inc.
All rights are reserved, including those for text and data mining, AI training, and similar technologies.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY
Copyright Elsevier 2024

frequencies. As such, from a conceptual standpoint, we can see that sensing and data science technologies are enabling increasing convergence between industrial (artificial) and human (natural) perception and decision-making. This opens new and exciting opportunities to synergize human and artificial intelligence with the ultimate goal of making manufacturing more efficient, safe, sustainable, and reliable.

In this chapter, we focus on ML technologies that enable information extraction from complex data sources commonly encountered in manufacturing. Specifically, we review basic concepts of convolutional neural networks (CNNs) and outline how these tools can be used to conduct diverse decision-making tasks of interest in manufacturing. At their core, CNNs use a powerful and flexible mathematical operation known as convolution to extract information from data objects that are represented in the form of regular grids (1D vectors, 2D matrices, and high-dimensional tensors) and irregular grids (2D graphs and high-dimensional hypergraphs). These data representations are flexible and can be used to encode a wide range of data objects such as audio signals, chemical spectra, molecules, images, and videos. Moreover, such representations can encode multichannel data, which allows for capturing color and multivariate inputs (e.g., multivariate time series and molecular graphs). CNNs use convolution operations to identify features that can be extracted from the data that best explain an output; such features are identified by identifying optimal convolution filters or operators, which are the parameters that are learned by CNN. The learning process of the operators requires sophisticated optimization algorithms and can be a computationally expensive process. CNNs are a class of models of an emergent ML field known as geometric deep learning, which leverages tools from geometry and topology to represent and process data.

The earliest version of a CNN was proposed in 1980 by Kunihiko Fukushima [1] and was used for pattern recognition. In the late 1980s, the LeNet model proposed by LeCun et al. introduced the concept of *backward propagation*, which streamlined learning computations using optimization techniques [2]. Although the LeNet model had a simple architecture, it was capable of recognizing hand-written digits with high accuracy. In 1998, Rowley et al. [3] proposed a CNN model capable of performing face recognition tasks (this work revolutionized object classification and detection). The complexity of CNN models (and their predictive power) has dramatically expanded with the advent of parallel computing architectures such as graphics processing units [4]. Modern CNN models for image recognition include SuperVision [5], GoogLeNet [6], VGG [7], and ResNet [8]. New models are currently being developed to perform diverse computer

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

vision tasks such as object detection [9], semantic segmentation [10], action recognition [11], and 3D analysis [12]. Nowadays, CNNs are routinely used in smartphones (unlock feature based on face recognition) [13].

While CNNs were originally developed for computer vision, the grid data representation used by CNNs is flexible and can be used to process datasets arising in many different applications. For instance, in the field of chemistry, Hirohara and co-workers [14] proposed a matrix representation of SMILES strings (which encodes molecular topology) by using a technique known as one-hot encoding. The authors used this representation to train a CNN that could predict the toxicity of chemicals; it was shown that the CNN outperformed traditional models based on fingerprints (an alternative molecular representation). In biology, Xie and co-workers [15] applied CNNs to count and detect cells from micrographs. More recently, CNNs have been expanded to process graph data representations, which has greatly expanded their application scope. These types of CNNs (known as graph neural networks) have been widely used in the context of molecular property predictions [16,17].

Manufacturing covers a broad space of important products and processes that is virtually impossible to enumerate; in this chapter, we focus our attention on applications of CNNs to examples of potential relevance to chemical and biological manufacturing (which cover domains such as pharmaceuticals, agricultural products, food products, consumer products, petrochemicals, and materials). We also highlight that these manufacturing sectors are seeing an emergent use of autonomous platforms that enable flexible and high-throughput experimentation and/or on-demand production; as such, the concepts discussed can be applicable in such a context. We provide specific case studies that we believe provide representative examples of how CNNs can be used to facilitate decision-making in manufacturing. Specifically, we show how to use CNNs to (1) decode multivariate time series data; (2) decode complex signals generated from microscopy and flow cytometry to detect contaminants in air and solution; (3) decode real-time ATR-FTIR spectra to characterize plastic waste streams; (4) predict surfactant properties directly from their molecular structures; and (5) map image data into signals for feedback control.

3.2 Data objects and mathematical representations

A wide range of datasets encountered in manufacturing can be represented in the form of a couple of fundamental mathematical objects: *tensors and graphs*. Such representations are so general that, in fact, it is

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

difficult to imagine a dataset that cannot be represented in this way. The key distinction between a tensor and a graph is that a tensor is a regular object (e.g., a regular mesh) while a graph is not (e.g., an irregular mesh). Moreover, tensors implicitly encode positional context, while graphs might or might not encode positional context. Differences between tensors and graph representations play a key role in the way that CNN architectures are designed to extract information from data. Unfortunately, it is often not obvious what data representation is most suitable for a particular application, and sometimes the representation might not naturally emerge from the data. In fact, one can think of CNNs as a tool for *representation learning*, in the sense that it aims to learn the best way to represent the data to make a prediction.

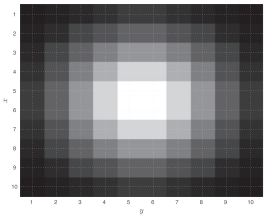
3.2.1 Tensor representations

Data objects are often attached to a grid; the most common example of this is a grayscale image, which can be represented as a 2D grid object. Here, every spatial grid point is a pixel and the data entry in such pixel is the intensity of light. A grayscale video is simply a sequence of grayscale images and can be represented as a 3D grid object (a couple of spatial dimensions plus time); here, every space-time grid point is a voxel and the data entry is the intensity of light. A common misconception of grid data is that it can only be used to represent images and videos but its scope is much broader. For instance, 3D density fields of chemicals or velocities (as those generated using molecular and fluid dynamics simulations) can be represented as a 3D grid; moreover, the thermal field of a surface can be represented as a 2D grid.

Tensors are mathematical objects used to represent grid data. A tensor is a generalization of vectors (1D tensors) and matrices (2D tensors) to high dimensions [18]. A key property of a tensor is that it implicitly encodes positional context and order (it lives in Euclidean space); specifically, every entry of a tensor has an associated set of coordinates/indexes that uniquely specify the location of an entry in the tensor (Fig. 3.1). Due to its positional context, the nature of the tensor can be altered by rotations; for instance, rotating an image (or transposing its associated matrix) distorts its properties.

Tensors are flexible objects that can also be used to represent multi-attribute/multichannel grid data. For example, color images can be represented as a superposition of three grids (red, green, and blue channels). Here, each channel is a 2D tensor (a matrix) and the stacking of these three channels is a 3D tensor. Channels can also be used to represent multivariate data in each entry of a grid. For instance, an audio or sensor

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024



0	0	0	1	1	1	1	0	0	0
0	1	1	2	2	2	2	1	1	0
0	1	2	3	4	4	3	2	1	0
1	2	3	5	6	6	5	3	2	1
1	2	4	6	8	8	6	4	2	1
1	2	4	6	8	8	6	4	2	1
1	2	3	5	6	6	5	3	2	1
0	1	2	3	4	4	3	2	1	0
0	1	1	2	2	2	2	1	1	0
0	0	0	1	1	1	1	0	0	0

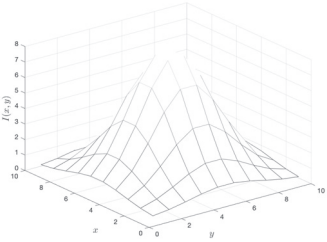


Figure 3.1 Representation of a grayscale image (*left*) as a 2D grid object (*middle*). The grid is a matrix in which each entry represents a pixel and the numerical value in the entry is the intensity of light. Representation of the grayscale image as a manifold (*right*); this reveals geometrical patterns of the image.

signal is a time series that can be represented as a one-channel vector, while a multivariate time series (e.g., as such obtained from a collection of sensors in a manufacturing facility) can be represented as a multichannel vector.

It is important to highlight that there is an inherent duality between images (reality as perceived by human vision or an optical device) and tensors (its mathematical representation). Specifically, images are optical fields that our visual sensing system capture and process to navigate the world and make decisions, while tensors are artificial mathematical representations used for computer processing. Making this distinction explicit is important because humans typically excel at extracting information from visual data (without having any knowledge of mathematics) compared to numerical data (e.g., number sequences). As such, this begs the questions: *Why do automation systems present data to human operators as numbers? What are the best visual representations that humans can use to interpret and analyze data more easily?* These questions are at the core of human-computer interaction and highlight the relevance of data visualization and processing techniques.

It is also important to highlight that the human vision system and the brain have inherent limitations in sensing and interpreting optical fields. For instance, the human vision and auditory system cannot capture all frequencies present in an optical field or an audio signal; as such, we need instrumentation (e.g., microscopes and nocturnal vision systems) that reveal/highlight information that cannot be captured with our limited sense. Moreover, the human brain often gets "confused" by distortions of optical fields and audio signals (e.g., rotations and deformations) and by noise (e.g., fog and white noise). These limitations can be overcome with the use of artificial intelligence tools such as CNNs. Here, sensing signals such as images and audio signals are represented mathematically as grid data to

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

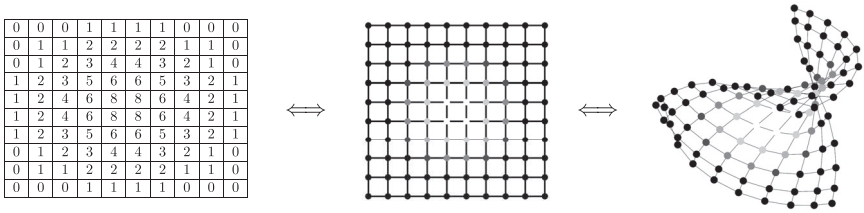


Figure 3.2 Representation of a grayscale image (*left*) as graph (*middle*). Each node in the graph represents a pixel and the weight in the node encodes the intensity of light. The graph is a topologically invariant object that is not affected by deformations (*right*).

extract information. Unfortunately, grid data representations are limited in that they inherently represent regular objects and are susceptible to rotation and deformations.

3.2.2 Graph representations

Graphs provide another flexible and powerful mathematical data representation. A graph is a topological object that consists of a set of nodes and a set of edges; each node is a point in a graph object and each edge connects a pair of nodes. The connectivity (topology) of a graph is represented as an adjacency matrix (Fig. 3.2).

A graph defines an irregular grid and one can attach multichannel data (features) to nodes and edges in such a grid. For instance, when representing a molecule, one can attach multiple features to each atom (e.g., identity and charge). Here, one can think of each feature as a channel of the graph; graphs with multiple features in nodes and edges are also known as multiattribute graphs. An example of how multiattribute graphs are used to represent molecules is presented in Fig. 3.3.

A fundamental property of a graph is that it does not directly encode positional context (it does not live in Euclidean space). As such, the graph can be stretched and rotated or the nodes can be shuffled around in space without affecting the underlying topology of the graph. In other words, the topology of the graph is fully dictated by the node–edge connectivity. This gives graph representations many useful properties (such as rotational invariance), which can be highly beneficial when representing certain systems. For instance, in molecular simulations, molecules might move around randomly in space, but their connectivity (e.g., hydrogen bonds) might not be affected [19]. Another key property of graphs is that, because these are irregular grid objects, one can develop CNNs that learn features of graphs of different sizes (e.g., polymer and peptide sequences).

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

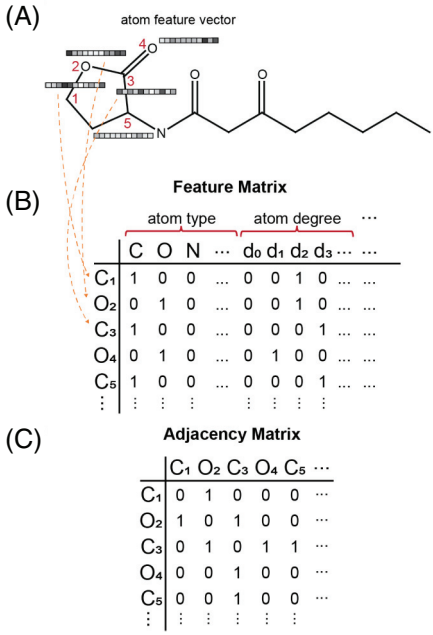


Figure 3.3 Representation of a molecule as a multichannel/multiattribute graph. The molecule topology is encoded in the adjacency matrix, while features of the atoms are encoded as vectors that are stacked in a feature matrix. *(Modified from [60] with permission from the American Chemical Society).*

It is worth highlighting that, while graphs are inherently 2D objects with no positional context, it is possible to encode the 3D (or higher dimensional) position of each node in a graph as features [20]. For instance, in a graph that represents the time evolution of a supply chain network, node features can contain the specific time point and the geographical coordinates. Moreover, it is possible to generalize graph representations to hypergraphs, which allow representing objects in which an edge connects multiple nodes. These more advanced geometric representations give rise to an emerging field of machine learning known as geometric deep learning [21] and to topological data analysis [22,23]. These approaches enable the analysis of complex 3D (and higher-dimensional) objects.

3.2.3 Color representations

The human vision system uses color as a key source of information to make decisions. For computer processing, color is typically represented using a red–green–blue (RGB) color space (see Fig. 3.4). The RGB color space

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

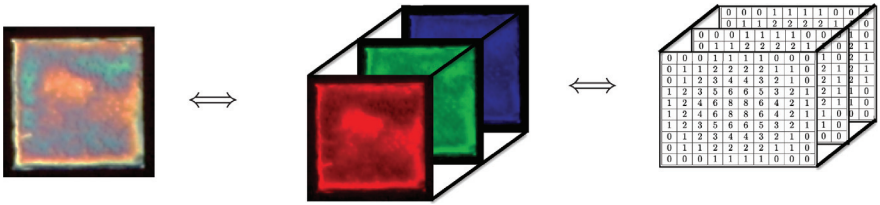


Figure 3.4 Representation of a color image (*left*) as a superposition of *red, green*, and *blue* channels (*middle*). Each channel is a 2D grid object (a matrix) and stacking the channels generates a 3D tensor.

is an additive space in which the red, green, and blue channels are mixed together to produce a wide range of colors [24].

Before the RGB space became the default in computer electronic systems, it already had a solid theory based on human perception of color. The choice of these three primary colors is related to the physiology of the human eye; specifically, the three photoreceptor cells (cone cells) in the human eye have peak sensitivity at three wavelengths associated with red, green, and blue. The RGB color space was designed to maximize the difference in response of cone cells to different wavelengths of light [25]. However, there are many other color spaces that can be useful in different situations. For example, the CIEL*A*B (or LAB) color space is a powerful representation that can be used for quantitative color comparisons.

The LAB space was defined by the International Commission on Illumination (CIE) to be a perceptually uniform color space, meaning that a group of colors separated by the same distance in LAB color space will look equally different. The three coordinates of the LAB color space represent the brightness of the color ($L* = 0$ for black, $L* = 100$ for white), its position between red and green (a, negative values indicate green, positive values indicate red), and its position between yellow and blue (b, negative values indicate blue, positive values indicate yellow). One can convert the RGB color space to a LAB color space using a nontrivial, nonlinear transformation. The LAB space can be used to highlight image features that might not be directly obvious to the human eye; this is a principle that can be exploited by CNNs to extract hidden information from images.

Unlike RGB and LAB imaging, which captures only three wavelength bands in the visible spectrum, spectral imaging can use multiple bands across the electromagnetic spectrum [26]. Spectral imaging can use infrared, visible spectrum, ultraviolet, X-rays, or some combination of the above. Multispectral and hyperspectral cameras can capture hundreds of bands for

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

each pixel in an image, which can be interpreted as a complete spectrum. Due to the wealth of information encoded in spectral images, this technique has been widely used in agriculture [27], healthcare [28], and nondestructive analysis of materials [29].

3.3 Convolutional neural network architectures

Convolutional neural networks denote a broad class of ML models that utilize specialized convolutional blocks to extract feature information from data objects. CNNs were originally designed to extract data from grid objects but they recently have been adapted to extract information from graphs. The key mathematical operation behind CNNs is convolution; loosely speaking, this operation transforms the data in a domain of the data object by applying a weighted sum of the entries in such domain. The weights of such sum are defined by a convolution filter or operator. The goal of the CNN is to identify optimal filters or operators that maximize information extracted from a collection of input data objects; here, "maximum information" means that such information can predict outcomes (labels) associated with the input data objects. For instance, one can train a CNN to predict the toxicity of a molecule directly from its molecular structure; this is done by training a CNN with a collection of molecules and associated toxicity levels.

Convolutional neural network architecture is simply a forward propagation of the input data object through a sequence of convolution, activation, and pooling operations that contain parameters that are learned by minimizing a loss function. A backward propagation (backpropagation) scheme is used to compute the gradient of the loss function with respect to the parameters (Fig. 3.5).

In this section, a basic discussion of the mathematics of convolution and computational procedures behind the training of CNNs is presented. With this, we aim to highlight the inner workings of CNNs and associated bottlenecks and challenges.

3.3.1 Convolution operations

We will first consider CNN networks f_{cnn} that map an input tensor V to output predictions $\hat{y}$. A particular convolutional layer takes an input object V and *convolves* it by applying a convolutional operator U which employs a set of convolutional filters or operators to produce a feature map Ψ. The

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

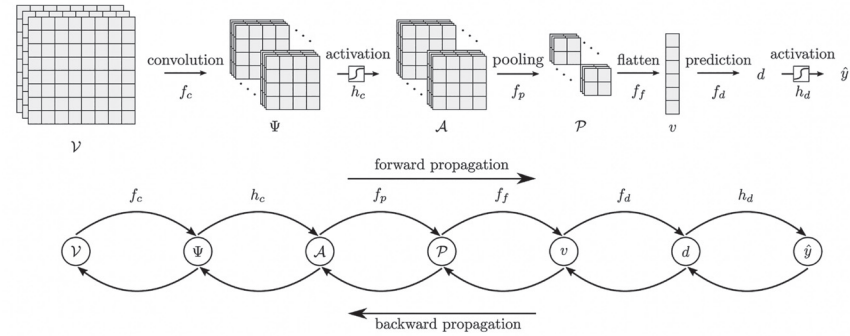


Figure 3.5 High-level view of a convolutional neural network (CNN) architecture. The input data object (e.g., grid or graph) is propagated through a sequence of convolution, activation, and pooling operations to make an output prediction. The trainable parameters of the operations are learned by minimizing a loss function. The derivative/gradient of the loss function is computed using backpropagation. *(Modified from [77] with permission from Wiley publishing).*

convolution operation is denoted as $\Psi = U * V$ and is simply a weighted summation between the entries of the convolution operator and the input object. Here, we also notice that the operator is typically much smaller than the input object and thus the operator is applied over a moving window (neighborhood) that covers the entire input object. This operation is not well-defined by the boundaries of V, as some indexing will violate the input domain, but this is typically resolved by adding padding of zero-valued entries around the image (a technique called zero-padding). We can also express this operation in the compact form $\Psi = f_c(V; U)$. As shown in Fig. 3.6, different operators highlight different features of the input object. Moreover, we note that one can propose a wide range of operators to extract information from the input data. In a CNN architecture, these operators are learned in a way that they extract information that best explains an output of interest.

Following Fig. 3.7, a common and intuitive interpretation of applying convolution operators is in terms of pattern recognition. The filters that comprise U each embed a particular pattern that, when convolved with a particular neighborhood of V (as depicted in Fig. 3.6), gives a score of how well the pattern is matched (where larger values denote greater matching). Hence, the feature map Ψ can be interpreted as a scoresheet of how certain patterns (as encoded by the filters in U) are manifested in V.

In a graph CNN architecture [30], the input graph is propagated based on its topology; here, convolution operations are performed analogously to

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

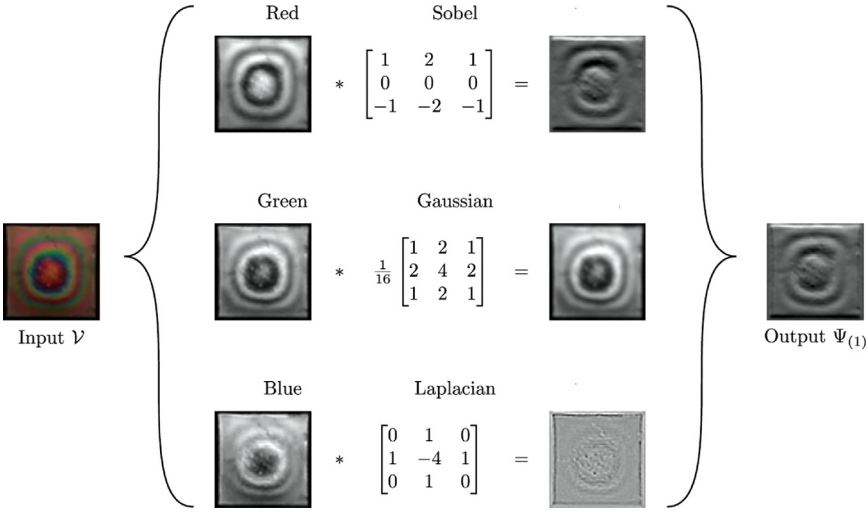


Figure 3.6 Convolution of a 3-channel grid (a color image). Each channel is processed using a different convolution operator. Each operator highlights/extracts different features (e.g., geometrical features) from the image. *(Modified from [77] with permission from Wiley publishing).*

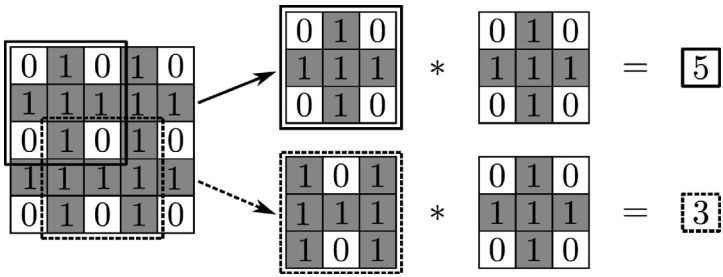


Figure 3.7 Convolution as a pattern matching technique. The convolution operator highlights regions in the grid in which the patterns of the image and of the operator matched. *(Modified from [77] with permission from Wiley publishing).*

grid data objects, by using weighted summations of the features of a node and its neighboring nodes. A message-passing framework [17] is typically used as a generalized approach to build GNN architectures that incorporate both node and edge features.

3.3.2 Activation functions

The feature map output of a convolutional layer is typically mapped element-wise through an activation function α to yield the activated object A. This operation can be written as $A = h_c(\Psi)$ and helps the CNN encode

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

nonlinear behavior. Common choices for the activation function include:

$$\alpha_{\text{sig}}(z) = \frac{1}{1 + e^{-z}}$$

$$\alpha_{\tanh}(z) = \tanh(z)$$

$$\alpha_{\text{ReLU}}(z) = \max(0, z).$$

Here, the rectified linear unit (ReLU) function $\alpha_{\text{ReLU}}(\cdot)$ is highly popular since it generally exhibits greater sensitivity to changes in the input [31]. Moreover, this function can be easily represented using piecewise linear functions and mixed-integer programming formulations.

3.3.3 Pooling

Another key component of CNNs are pooling layers; pooling operations are dimension reduction mappings f_p that seek to summarize/reduce the activation A by collapsing certain sub-regions of smaller dimension (referred to as pooling). Pooling helps to make the learned representation more invariant to small length-scale perturbations [32]. Common choices include max-pooling and average pooling where the maximum or average of a sub-region is used to scalarize, respectively.

3.3.4 Convolution blocks

Convolution blocks in CNNs denote the combination of convolution, activation, and pooling of a given input. In other words, a convolution block employs the mapping f_{cb} such that:

$$P = f_{\text{cb}}(V; U) = f_p\big(h_c\big(f_c(V; U)\big)\big). \tag{3.1}$$

Blocks can employ more complex operation nesting (e.g., multiple convolutional layers), but we consider blocks that follow Eq. (3.1) for simplicity in presentation. Moreover, these can take pooled outputs P as input and thus facilitate the use of multiple convolutional blocks in succession via recursively calling f_{cb}. These blocks essentially act as feature extractors via the convolutional filters they employ. Moreover, this feature extraction becomes more specialized for blocks that are located deeper in the CNN. In the context of computer vision, this means that the first blocks extract simple features (e.g., edges or colors) and the deeper blocks can extract more complex patterns (e.g., particular shapes). Thus, we can think of performing multiple convolution operations as a *multistage distillation process*

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

where the successive feature spaces capture patterns of increased length-scale and complexity (i.e., distill the image data into an increasingly purified feature) [33].

3.3.5 Feedforward neural networks

The output P of the last convolution block is typically flattened (vectorized); this can be represented as the mapping f_f and yields the feature vector v:

$$v = f_f(P). \tag{3.2}$$

The feature vector is then fed into a feedforward neural network model f_d which predicts the desired state space vector $\hat{y}$. Fig. 3.5 illustrates a typical image CNN model that implements the components described above. It employs a couple of convolution blocks and can be described in the functional form:

$$\hat{y} = f_{\text{cnn}}(V), \tag{3.3}$$

where, the mapping f_{cnn} is a nested set of convolution, activation, pooling, and flattening operations. This emphasizes that convolution blocks are feature information extractors and the dense layers act as the predictors whose feature space is the flattened output v of the final block.

3.3.6 Data augmentation

Data augmentation denotes a class of techniques that aim to artificially expand the size of a training dataset by applying varied perturbations/transformations to the input data. In the context of computer vision, augmentation is often used to expand the size of the training image set in an effort to decrease the likelihood of the CNN encountering novel images and in an effort to avoid rotation invariance issues associated with grid data objects. Image augmentation generally denotes perturbing the training images such that the CNN sensor can be robust to those types of visual disturbance. Common perturbations include rotation, translation, cropping, blurring, brightness changing, splattering, and more. There are many software tools available that implement these transformations, which include TensorFlow and ImgAug [34,35]. Fig. 3.8 shows how a training image is augmented via a variety of perturbations. This methodology helps mitigate the risk of CNNs encountering novel images, but it is not typically possible to account for all the disturbances a process might encounter.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

(A) Original (B) Splattered (C) Fogged (D) Shifted

Figure 3.8 Examples of image perturbation methods used in data augmentation techniques. *(Modified from [79] with permission from Elsevier).*

3.3.7 Training and testing procedures

CNN training procedures seek optimal model parameters (i.e., convolution operators and dense network weights) that minimize the error incurred by the output CNN predictions relative to the outputs of the training dataset. Here, we consider a training set $|K|$ that contains input–output pairs. The prediction error minimized in the training procedure is called the loss function L. For example, regression models typically use a sum-of-squared-error (SSE) loss function:

$$L(\hat{y}) = \left\| \hat{y} - y \right\|_2^2. \tag{3.4}$$

Thus, by grouping all the CNN model parameters into θ we can express model training as a standard optimization problem:

$$\begin{aligned} \min_{\theta} \quad & \sum_{k \in \mathcal{K}} L\left(\hat{y}^{(k)}\right) \\ \text{s.t.} \quad & \hat{y}^{(k)} = f_{\text{cnn}}\left(V^{(k)}; \theta\right), k \in \mathcal{K}. \end{aligned} \tag{3.5}$$

This can readily be expressed as an unconstrained optimization problem by inserting the constraint equations directly into the objective. Stochastic gradient descent (SGD) is typically used to solve this problem due to the large amount of training data, the high number of model parameters, and the model complexity. Moreover, forward and backward propagation techniques are used to evaluate the objective and derivative values required by each iteration of the SGD algorithm (see Fig. 3.5). For classification models, one typically uses an entropy loss function.

3.3.8 CNN architecture optimization

Neural architecture search (NAS) has been designed to automatically search for the best CNN architecture for a given dataset. Specifically, it uses a search method (e.g., reinforcement learning, evolutionary algorithms, or stochastic

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

gradient descent) to explore the user-defined search space and chooses the best architecture based on the performance of the generated model (e.g., validation accuracy) on a given task. The search space contains all possible architectures. The architectures discovered by NAS have been shown to outperform manually designed architectures in various tasks such as image classification [36], image segmentation [37], natural language processing [38], and time series prediction [39].

3.3.9 Transfer learning

The training process of a CNN can be highly computationally intensive because this requires performing a large number of convolution operations (e.g., at each SGD iteration and for a large training set). However, it is possible to use the filters/operators of existing, trained CNNs to extract information for a new dataset and such information can be used for different tasks (e.g., develop a different ML model such as a support vector machine). This approach is known as transfer learning and is based on the observation that, while the filters used are not optimal for the new dataset at hand, they can still extract valuable information. For instance, in the context of human learning, one can often detect features for objects that we have not seen before.

3.4 Case studies

In this section, we present a set of case studies that highlight the potential uses of CNNs in a manufacturing context. Our goal with this is not to provide an exhaustive list of applications, but rather to highlight the capabilities of CNNs as well as to show how data representations can be used in creative ways to tackle diverse problems.

3.4.1 CNNs for sensor design

3.4.1.1 Detecting contaminants in solution

This case study is based on the results presented by Jiang et al. [40]. Endotoxins are lipopolysaccharides (LPS) present in the outer membrane of bacteria [41]. Recently, it has been observed that micrometer-sized liquid crystal (LC) droplets dispersed in an aqueous solution can be used as a sensing method to detect and measure the concentration of endotoxins in different bacterial organisms. After exposure to endotoxins, LC droplets undergo

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

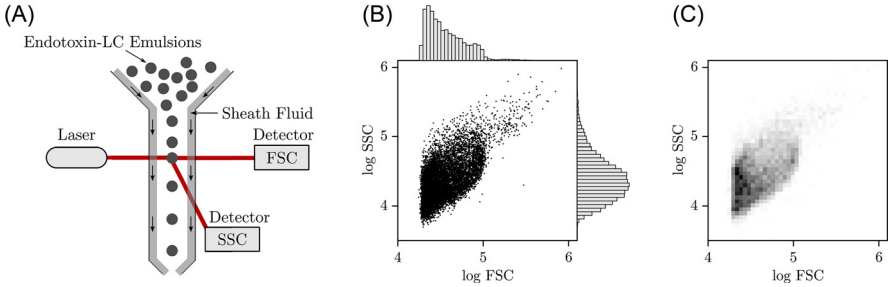


Figure 3.9 Overview of the interaction between an endotoxin and liquid crystal (LC) emulsions. (A) Generation of forward and side scattering (FSC/SSC) scatter fields. The endotoxin-LC emulsions are pumped into the flow cytometer in the direction of the sheath flow. Laser light is scattered from the LC droplets and collected at two angles (FSC and SSC). By combining the FSC and SSC data points for 10,000 LC droplets, we generate an FSC/SSC field. (B) Scatter field generated by LC droplets exposed to 100 pg/mL of endotoxin. Marginal probability densities of FSC (*top*) and SSC (*right*) light in log scale are generated with 50 bins. (C) 2D grid of the scatter fields by binning and counting the number of events in a bin. *(Modified from [40] with permission from the Royal Society of Chemistry).*

transitions that yield distinct optical signatures that can be quantified using flow cytometry (Fig. 3.9).

Flow cytometry produces complex data objects in the form of scatter point clouds of forward and side scattering (FSC/SSC); here, each point represents a scattering event of a given droplet. The key observation is that a point could be converted into a 2D grid data object via binning; this is done by discretizing the FSC/SSC domain and by counting the number of points in a bin. The 2D grid object obtained is a matrix that we can visualize as a grayscale image; here, each pixel is a bin and the intensity is the number of events/droplets in the bin (bins with more droplets appear darker). This visualization is a 2D projection of a 3D histogram (the third dimension corresponds to the number of events, also known as the frequency). In other words, the 2D grid object captures the geometry (shape) of the joint probability density of the FSC/SSC. These geometrical features can be extracted automatically using a CNN.

The effect of endotoxin concentration on the FSC/SSC scatter fields (after binning) is shown in Fig. 3.10; clear differences in the patterns can be observed at concentrations that are far apart but differences are subtle at close concentrations. We trained a 2D CNN that can automatically detect these changes and predict concentrations from the scatter fields. We used the following procedure to obtain the 2D grid data objects (samples) that were

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

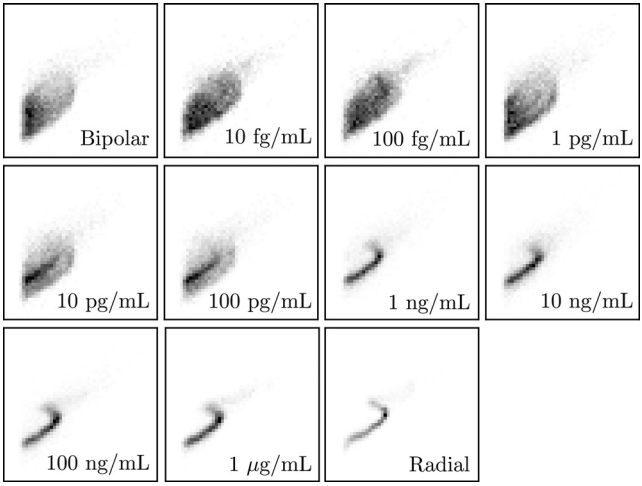


Figure 3.10 Effect of endotoxin concentration on the forward and side scattering (FSC/SSC) scatter field (represented as 2D grid objects). Scatter fields obtained using liquid crystal (LC) droplets exposed to different concentrations of endotoxin. As the endotoxin concentration increases, the LC droplet population shifts from a bipolar to a radial configuration. *(Modified from [40] with permission from the Royal Society of Chemistry).*

fed to the CNN. For each sample, we generated bins for a given scatter field by partitioning the FSC and SSC dimensions into 50 segments (the grid has $50 \times 50 = 2500$ pixels). For each sample, we were also given reference scatter fields that represented limiting behavior: bipolar control (negative) and radial control (positive). Each sample is given by a 3-channel object V where the first channel is the negative reference matrix $V_{(1)}$, the second channel is the target matrix $V_{(2)}$, and the third channel is a positive reference matrix $V_{(3)}$ (each channel contains a 50×50 matrix). This procedure is illustrated in Fig. 3.11. This multichannel data representation approach magnifies differences in the target matrix from the references (we will see that neglecting the negative/positive references does not give accurate predictions).

The 3-channel data object was fed to a CNN, which we call *EndoNet*. EndoNet has an architecture of Conv(64)-MaxPool-Flatten-Dense(32)-Dense(32)-Dense(1). The output block generates a scalar prediction $\hat{\gamma}$ (corresponding to the endotoxin concentration). In other words, the CNN seeks to predict the endotoxin concentration from the input flow cytometry fields. The architecture of EndoNet is shown in Fig. 3.11. The regression results for EndoNet are presented in Fig. 3.12. EndoNet extracts pattern information within and between each channel of the input image. Capturing

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

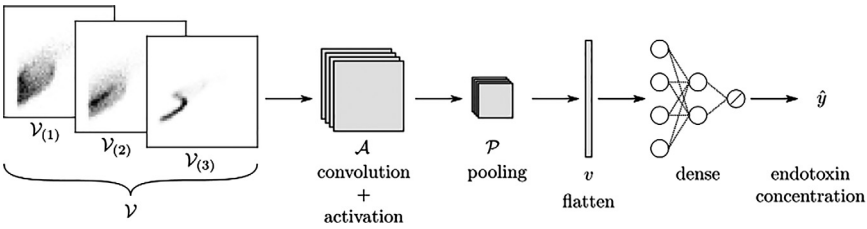


Figure 3.11 EndoNet architecture. The input to EndoNet is a 3-channel object $\mathcal{V} \in \mathbb{R}^{50\times50\times3}$; the channels correspond to the target, the negative reference, and the positive reference (each is a matrix of dimension 50×50). EndoNet includes a convolutional block with 64, 3×3 operators and a maxpooling block with 2×2 operators. The feature map $\mathcal{A}$ generated by the convolution and activation blocks is a tensor $\mathbb{R}^{48\times48\times64}$. The max-pooling block generates a feature map $\mathcal{P} \in \mathbb{R}^{24\times24\times64}$ that is flattened into a long vector $v \in \mathbb{R}^{36863}$. This vector is passed to two dense layers (each having 32 hidden units). The predicted endotoxin concentration $\hat{y}$ is the output from the dense layer activated by a linear function. *(Modified from [40] with permission from the Royal Society of Chemistry).*

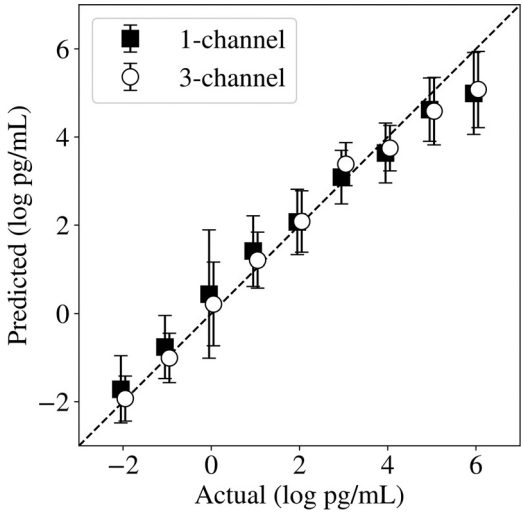


Figure 3.12 Predicted and actual concentrations at different concentrations using 1-channel and 3-channel representations. *(Modified from [40] with permission from the Royal Society of Chemistry).*

differences between channels provides context for CNN and has the effect of highlighting the domains in the scatter field that contain the most information. To validate this hypothesis, we conducted predictions for EndoNet using the 1-channel representation as input (we ignored the positive and negative channels). For the 1-channel representation, we obtained an RMSE

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

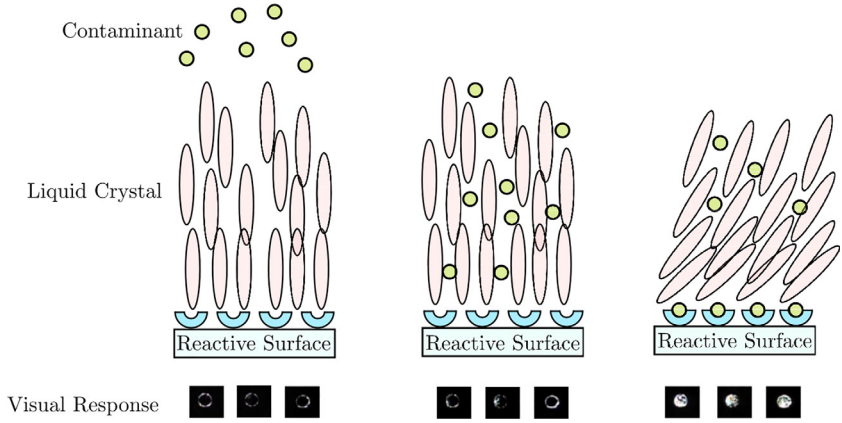


Figure 3.13 Working design principles of a liquid crystal chemical sensor. A functionalized liquid crystal (LC) film selectively responds to the presence of an air contaminant and triggers an optical response. *(Modified from [42] with permission from the American Chemical Society).*

of 0.97; for the 3-channel representation, we obtained an RMSE of 0.78. It is particularly remarkable that EndoNet can accurately predict concentrations that span eight orders of magnitude.

3.4.1.2 Detecting contaminants in air

This case study is based on work presented by Smith et al. [42]. LCs also provide a versatile platform for sensing air contaminants [43,44] and for sensing heat transfer and shear stress (mechanical sensing) [45]. In the context of air chemical sensing, an LC sensor can be prepared by supporting a thin LC film on a chemically functionalized surface. The molecules within the LC film (the mesogen) bind to the surface and assume a homeotropic orientation that provides an initial optical signal. Subsequent exposure of the LC film to an analyte leads to diffusive transport of the analyte through the LC phase and displacement of the mesogen at the surface, triggering rich space-time optical responses (see Fig. 3.13).

A challenge in the development of LC sensors is their sensitivity to *interfering* species. For instance, LC sensors designed for the detection of dimethyl methylphosphonate (DMMP), $CH_3PO(OCH_3)_2$, might exhibit similar optical responses when exposed to humid nitrogen [46]. These issues are illustrated in the experimental responses shown in Fig. 3.14. Our goal here is to determine whether or not one can unravel *hidden patterns* in the optical responses that can help discern between chemical species.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

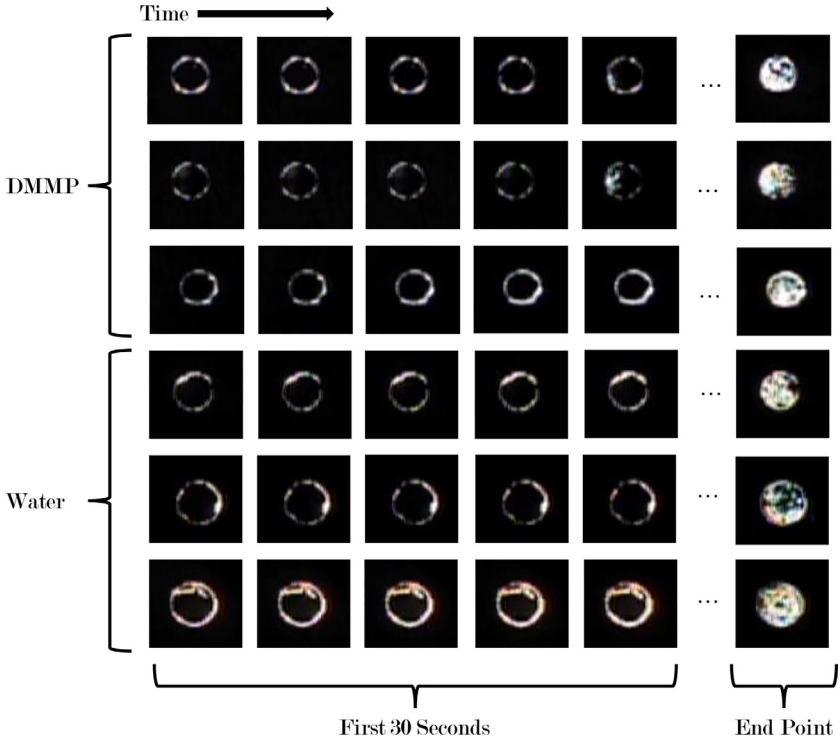


Figure 3.14 Optical responses of liquid crystals under gaseous N_2-water (30% relative humidity) and N_2-DMMP (10 PPM) environments. Liquid crystals (LCs) were deposited into microwells with a diameter of 3mm to enable high-throughput data collection. *(Modified from [42] with permission from the American Chemical Society).*

We designed a CNN to automatically detect contaminant presence from optical LC images (micrographs). In summary, the approach is based on transfer learning; specifically, we used VGG16, which is a pretrained CNN for extracting features from the optical responses, and these features were fed into a simple support vector machine (SVM) that predicts if the contaminant is present. The results indicate that features extracted from the first and second convolutional layers of `VGG16` allow for *perfect* classification accuracy. The results also indicate that complex *spatial color patterns* are developed within seconds in the LC response, which leads us to hypothesize that fluctuations in LC tilt orientation (angle) play a key role in sensor selectivity. Moreover, the analysis reveals that color is a key source of information that CNN is looking for.

The dataset analyzed in this study was obtained from six videos that show the response of LCs to a gaseous stream of N_2 containing 10 ppm

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

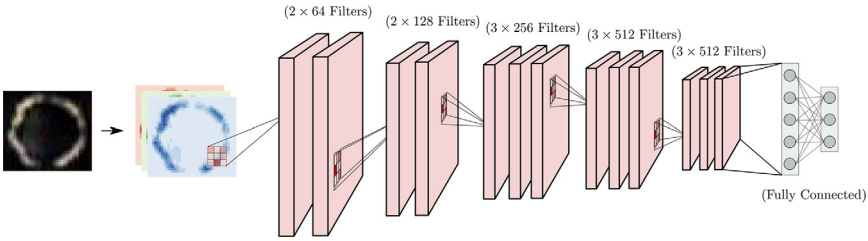


Figure 3.15 Schematic of a pretrained `VGG16` architecture used for feature extraction of liquid crystal (LC) micrographs. *(Modified from [42] with permission from the American Chemical Society).*

DMMP and six videos that show the response of LCs to a gaseous stream of N_2 containing 30% relative humidity (both at room temperature). Each video tracks the dynamic evolution of multiple independent microwells (the total number of microwells recorded was 391). We split each frame into several images, each containing a single microwell at a specific time. The total number of microwell snapshots generated was 75,081.

The trained `VGG16` network is freely available through the Keras software and is what is used during feature extraction [34]. This highlights the fact that highly sophisticated CNNs are openly available nowadays for conducting feature extraction. A simplified representation of the `VGG16` architecture is shown in Fig. 3.15. The `VGG16` network has been pretrained to classify highly complex images from the internet (not related to our particular application, such as cats and dogs) and the deepest layers have been carefully tuned to differentiate such images. The early layers of the network are the most general and are easier to interpret; accordingly, we use the outputs of the first and second convolutional blocks to inform features for LSVM classification. A visual representation of this process for the first and second convolutional blocks is provided in Fig. 3.16.

The framework using `VGG16` features and LSVM was able to classify water and DMMP micrographs with *100% accuracy*. This result was achieved when using all of the 128 features of the second convolutional layer. An accuracy of 98% is obtained when we use the 64 features of the first convolutional layer. These results indicate that LC features that develop *early in the sensor response* are highly informative and sufficient to discriminate among chemical environments.

3.4.2 Molecule design

Machine learning techniques have been recently used for property predictions of molecules such as water solubility [47–49], toxicity [50–52], and

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

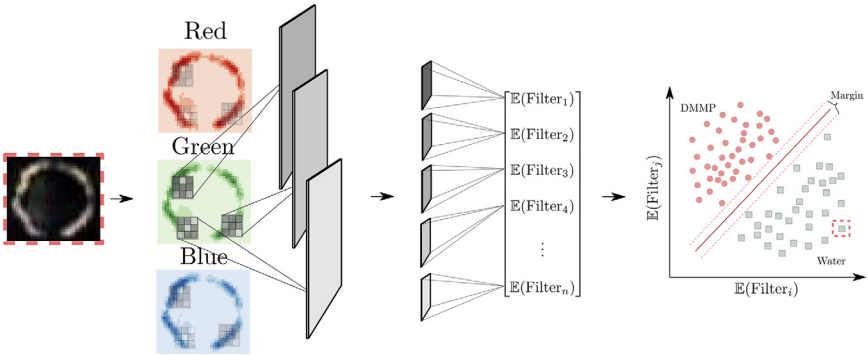


Figure 3.16 Schematic of a machine learning (ML) framework including feature extraction and classification based on SVM. *(Modified from [42] with permission from the American Chemical Society).*

lipophilicity [53,54]. A fundamental step in the use of ML algorithms is the predefinition or precalculation of molecular descriptors [55–58]; such descriptors are used as input data to develop quantitative structure-property relationship models [59]. Among the various ML techniques, graph CNNs (GNNs) [16,17] have gained special popularity because they can directly incorporate molecular graph representations, thus retaining key structural information on molecules while avoiding the need to precalculate/predefine molecular descriptors for which density functional theory (DFT) or molecular dynamics (MD) simulations may be required. Overall, GNNs have shown strong predictive power in molecular property predictions, and have great potential to be applied to other fields for more accurate model development as well as enabling high-throughput screening of materials for manufacturing.

We show how to use GNNs for predicting critical micelle concentrations (CMCs) of surfactants. This study is based on the work presented by Qin et al. [60]. When dissolved in water, surfactant monomers will undergo a cooperative aggregation process, called self-assembly, to form spherical micelles or related aggregate structures [61] (Fig. 3.17). The formation of micelles in a solution can induce significant changes in key solution properties including electrical conductivity, surface tension, light scattering, and reactivity [61,62]. Consequently, predicting conditions under which surfactants self-assemble is important for surfactant selection and design [63]. A critical parameter that characterizes surfactant self-assembly behavior is the CMC, which is the minimum surfactant concentration at which self-assembly occurs.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

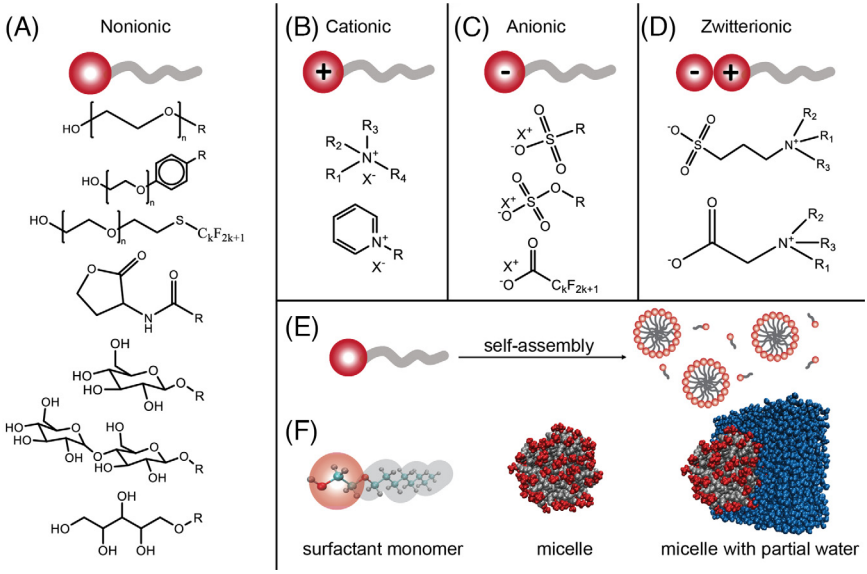


Figure 3.17 Overview of surfactant molecular structures and self-assembly process in micelles. (A–D) Sample structures of four classes of surfactants. Surfactants are categorized by the properties of their head groups as nonionic (A), cationic (B), anionic (C), or zwitterionic (D). (E) Surfactant monomers aggregate into spherical micelles in water with hydrophilic head groups facing toward the solvent and hydrophobic tail groups sequestered inside the micelle core. (F) Snapshots of a surfactant micelle from a representative molecular dynamics (MD) simulation with water shown in blue. *(Modified from [60] with permission from the American Chemical Society).*

Conventionally, the CMCs are obtained experimentally by tensiometry, but it is laborious and expensive [64–66]. Here, we show that GNNs can predict CMC values directly from the molecular graph of a surfactant monomer. We gathered experimental CMC data measured at room temperature in water for 202 surfactants [62,64,67,68] covering all surfactant classes. The proposed GNN architecture consists of two graph convolutional layers, one average pooling layer, two fully connected hidden layers, and one final output layer. A graph convolution layer updates each atom by aggregating the features of itself and its neighbors and maps the updated features into a hidden layer with 256 hidden features. The GCN model contains a total of 216,833 trainable parameters.

The cross-validation (CV) RMSE on all classes of surfactants has a mean value of 0.39 with no significant outliers. We tested the model performance on a test data set, which contains samples from each surfactant class. We verified the distribution of the test samples using t-distributed stochastic

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

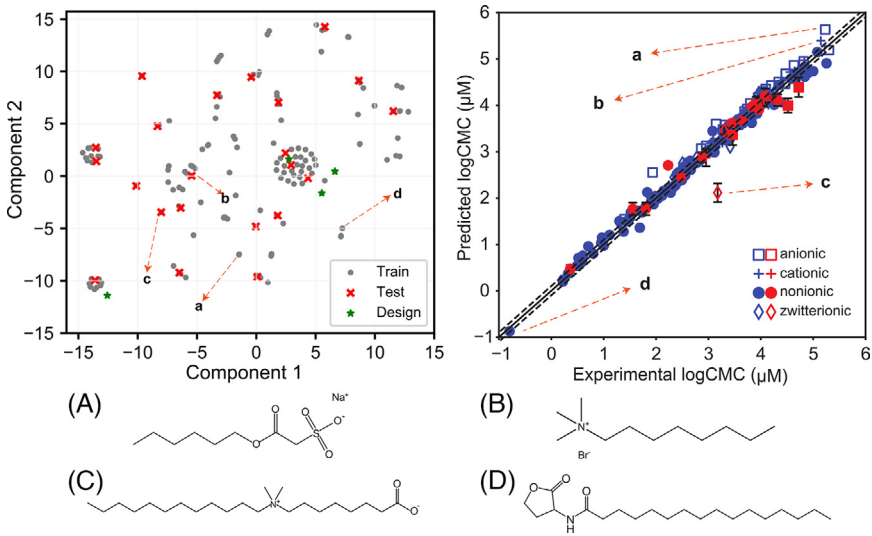


Figure 3.18 Graph neural networks (GNN) predictions for all classes of surfactants. *Left*: low-dimensional distribution of surfactant fingerprints using t-distributed stochastic neighbor embedding (t-SNE). The test samples (*red crosses*) are widespread, and most of the designed surfactants (*green points*) fall outside of the clusters of the existing data set. *Right*: parity plot between the predicted and experimental log critical micelle concentration (CMC) values (training data in *blue* and test data in *red*). The best-fit slope of the test data is 0.91 ($R^2 = 0.92$), and the test RMSE is 0.30. Molecular structures are shown for the selected extreme points. Structure (A) is an anionic surfactant (*minor outlier*) with a high log CMC value. Structure (B) is a cationic surfactant (*minor outlier*) with a high log CMC value. Structure (C) is a zwitterionic surfactant (*major outlier*). Structure (D) is a nonionic surfactant with a low log CMC value. *(Modified from [60] with permission from the American Chemical Society).*

neighbor embedding (t-SNE) [69], a nonlinear dimension reduction technique to visualize high-dimensional data, on the molecular fingerprints of the surfactants [56]. Fig. 3.18 illustrates that the test samples are widespread, indicating the inclusion of unlike surfactant structures and classes in the test data set, which cover a much more diverse spectrum of surfactants than the data sets used in previous QSPR models.

Fig. 3.18 also shows a parity plot between the experimental and predicted log CMC values for the training and testing sets. Cationic surfactants have the lowest test RMSE (0.07) followed by nonionic (0.18) and anionic (0.32) surfactants, and the model performs worst for zwitterionic surfactants (0.76). The parity plot also suggested a slightly lower accuracy for surfactants with relatively large log CMC values (>4.5). The overall predictability of the GCN model outperforms that of a prior QSPR model reported in the

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

literature [64]. The differences in the molecular structures found in our dataset highlight the wide variety of surfactants that the GCN model can capture.

3.4.3 Decoding of spectra

We now show how to use CNNs to decode real-time spectra; specifically, we show how to characterize plastic components using real-time ATR-FTIR spectra. This case study also aims to illustrate how innovative (nonintuitive) data representations can be used to obtain more information from spectra. This study is based on the work reported by Jiang et al. [70].

Plastics are used in a wide range of applications such as food packaging, construction, transportation, health care, and electronics. Notably, only 20% of all plastics produced were recycled [71]; this recycling rate is notably low compared to that of other materials (e.g., aluminum has a recycling rate of nearly 100%). A key obstacle that arises here is the characterization of plastic components in mixed-platic-waste (MPW) streams. ATR-FTIR (attenuated total reflection-Fourier transform infrared spectroscopy) is an instrumentation technique that can be used for analyzing plastic components found in MPW in real-time; as such, one can envision the development of fast, online ML techniques that can analyze ATR-FTIR spectra to characterize MPW streams. Here, we show that CNNs can characterize plastic components of MPW by decoding ATR-FTIR spectra.

Experimental data was obtained by preparing small sheets of plastics of different shapes and using ATR-FTIR to scan sheets for 10 different types; this data collection approach mimics how rigid waste plastics are found in the online processing of MPW streams (Fig. 3.19). The spectra collected can be represented as 1D vectors and analyzed by using 1D CNNs [72]. The 1D CNN extracts features of a spectrum by convolving it with different filters. However, the 1D representation might fail to capture correlations across frequencies. Gramian angular fields (GAF) have been recently used to encode 1D series data into matrices that capture correlation structures and that are processed using 2D CNNs; this data transformation approach has been shown to improve classification accuracy for time series data [73].

A Gramian angular field represents vectors in a polar coordinate system and converts these angles into symmetric matrices using various operations. There are a couple of GAF types: Gramian angular summation fields (GASF) and Gramian angular difference fields (GADF). The conversion of spectra to GASF and GADF matrices is illustrated in Fig. 3.20; here, the GAF matrices are represented as grayscale images.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

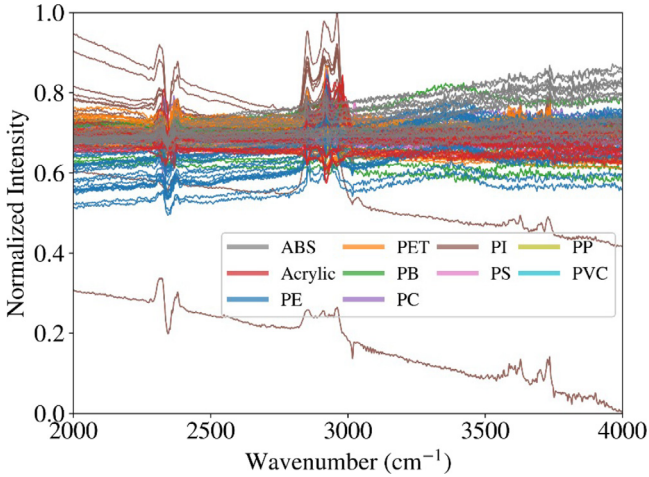


Figure 3.19 Normalized infrared spectral intensities of various plastic materials. Each spectrum is a vector of length 4150. The resulting spectra contain significant noise and systematic errors. *(Modified from [70] with permission from Elsevier).*

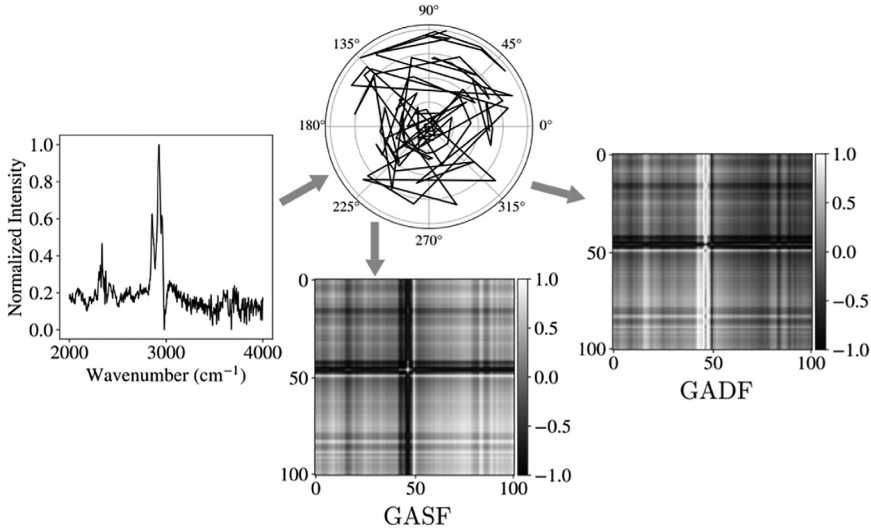


Figure 3.20 Conversion from 1D signal to Gramian angular summation fields (GASF) and Gramian angular difference fields (GADF) matrices. The 1D signal is first mapped to the polar coordinate system and finally converted to GASF and GADF matrices. Encoding the 1D signal into Gramian angular fields (GAF) matrices captures the relationship between the signal intensity at different wavenumbers. *(Modified from [70] with permission from Elsevier).*

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

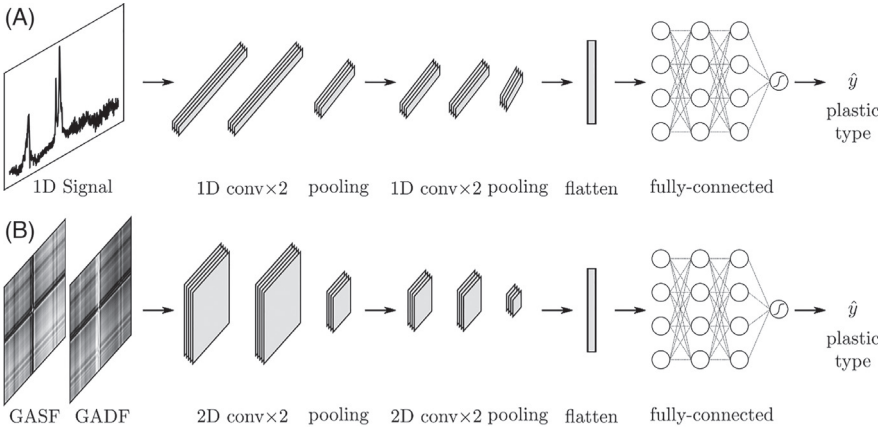


Figure 3.21 Architectures of (A) PlasticNet (1D) and (B) PlasticNet (2D). PlasticNet (1D) inputs a vector of 4150 and outputs the predicted plastic type. It contains 4 1D convolutional layers (each has 64 filters of dim 3), 1D max-pooling layers (each has a pooling window size of 2), a flatten layer, and 3 fully-connected layers (each has 64 units and a dropout ratio of 0.2). The activation functions between the layers are ReLUs. The final output activation function is softmax. PlasticNet (2D) inputs a Gramian angular summation fields (GASF) and a Gramian angular difference fields (GADF) matrix. The input size varies from $50 \times 50 \times 2$ to $250 \times 250 \times 2$. It has four 2D convolutional layers (each has 64 filters of 3×3), two 2D max-pooling layers (each has a pooling window size of 2×2). The flatten, fully-connected layers and activation function setups are the same as the ones of PlasticNet (1D). *(Modified from [70] with permission from Elsevier).*

The comparison of architectures of PlasticNet (1D) and (2D) is shown in Fig. 3.21. Classification results for PlasticNet (1D) and (2D) are presented in Fig. 3.22. PlasticNet (2D) has a higher accuracy when the input size is larger than 100,100, compared to PlasticNet (1D) on raw IR spectra (77.7%). Specifically, PlasticNet (2D) increases the accuracy of the PlasticNet (1D) by 12.4% (Fig. 3.23); this confirms that correlation information in spectra is important for classification. To validate the effectiveness of the proposed CNN models, we compared them with four commonly used ML classifiers, including radial basis function (RBF) based support vector machine (RBF-SVM), Random Forest (RF), k-Nearest Neighbors (kNN), Gaussian process classifier (GPC). The accuracy of PlasticNet (2D) is slightly higher (1%) than that of RBF-SVM when the input size is larger than 200×200. This indicates that RBF-SVM is comparable to CNN-based methods; however, SVMs offer limited flexibility to capture different representations for IR data. The accuracy of all methods saturates at 87%, which suggests that the dataset itself contains significant errors that neither the CNN-based nor the SVM methods can explain. In subsequent work, we have shown that decoding fast

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

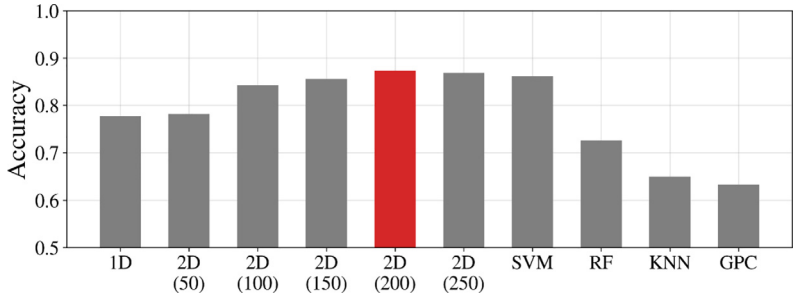


Figure 3.22 Comparison of the accuracy of convolutional neural network (CNN)-based methods and other machine learning (ML) algorithms. PlasticNet (2D) with an input size of 200 × 200 × 2 has the highest accuracy of 87.29%. SVM with RBF kernels has a comparable accuracy of 86.14%. The accuracy of PlasticNet (2D) is higher than that of PlasticNet (1D), indicating that the conversion from the original 1D signal to 2D Gramian angular fields (GAF) matrices captures more information. The accuracy of PlasticNet (2D) increases as the input matrix increases, indicating that a larger input matrix contains more information. *(Modified from [70] with permission from Elsevier).*

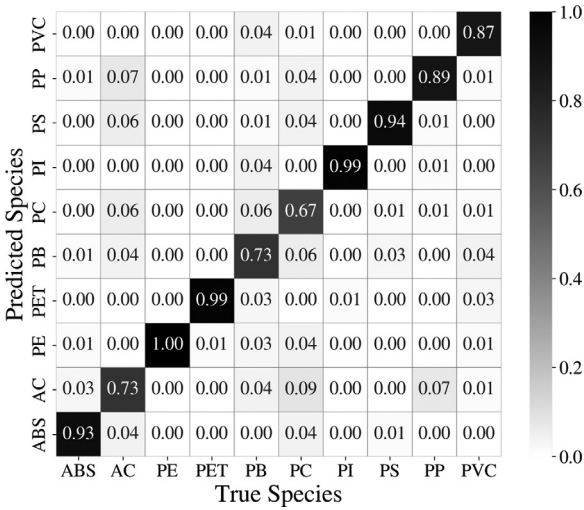


Figure 3.23 Confusion matrix for PlasticNet (2D) with an input size of 200 × 200 × 2. The overall accuracy is 87.3%. Each column represents a true plastic species, and each row represents a model predicted plastic species. The entries along the diagonal are where the plastic species are correctly classified. Many diagonal entries are close to one, indicating that the PlasticNet (2D) has excellent classification accuracy. However, some plastic types cannot be classified with high accuracy (e.g., PC and AC). *(Modified from [70] with permission from Elsevier).*

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

MIR (mid-infrared spectroscopy) spectra using CNNs can achieve a nearly perfect plastic classification accuracy [74].

3.4.4 CNNs for multivariate process monitoring

Multivariate process monitoring is a common task performed in manufacturing to identify abnormal/faulty behavior. Here, the idea is to collect multivariate time series (for different process variables) under different modes of operation (each mode is induced by a specific fault). The goal is to identify features (signatures) in the time series to determine if the process is a given mode. The case study presented here uses benchmark data for the Tennessee Eastman (TE) process [75]. This study was based on the work presented by Wu and Zhao [76], and Jiang and Zavala [77].

The Tennessee Eastman process units include a reactor, condenser, compressor, separator, and stripper. The TE process produces two products (G and H) and a byproduct (F) from four reactants (A, C, D, and E). Component B is an inert compound. In total, the TE process contains a total of 52 measured variables; 41 of them are process variables and 11 are manipulated variables. This process exhibits 20 different types of faults related to changes in feed temperatures, compositions, reaction kinetics, and so on.

The Tennessee Eastman process data is obtained from Harvard Dataverse [78]. The 52 process variables are sampled every 3 minutes; the transformation of multivariate signal data to matrices is shown in Fig. 3.24. We construct an input data sample by using 52 signal vectors (each vector contains 60 time points) that are combined into a 52×60 matrix (V). We have a total of 6947 input samples and the training:validation:testing data ratio used is 11:4:5. Fig. 3.25 is the confusion matrix obtained for the CNN; the overall classification accuracy was 0.7561. With the exception of faults 3, 4, 5, 9, and 15, most faults can be identified accurately. Faults 3, 9, and 15 are especially difficult to detect because the mean, variance, and higher-order variances do not vary significantly.

3.4.5 CNNs for image-based feedback control

In this study, we consider techniques for incorporating CNN sensors (CNNs that map image signals to controllable state signals) in feedback control systems. We place a particular focus on the need for real-time novelty/anomaly detection approaches that provide robustness in effectively mitigating the consequences of visual disturbances. The concepts discussed here are a

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

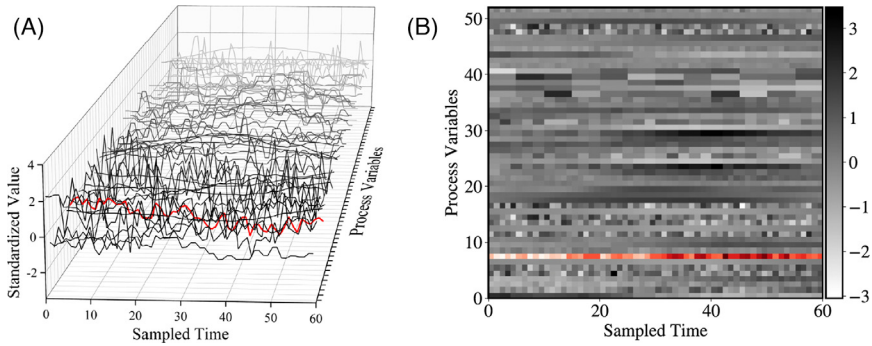


Figure 3.24 Representation of a multivariate time series as a 2D image. (A) 52 process variables collected 60 times over a 3-hour period. The variables are normalized with a zero mean and a unit variance. (B) 52 × 60 matrix (visualized as an image). The *red line* in (A) and *red row* in (B) indicate the same data. Each row of the image represents one of the time series in (A). The fault number is 7 for (A) and (B), 9 for (C), and 15 for (D). We can see that (C) and (D) are visually similar but belong to different fault groups. *(Modified from [77] with permission from Wiley publishing).*

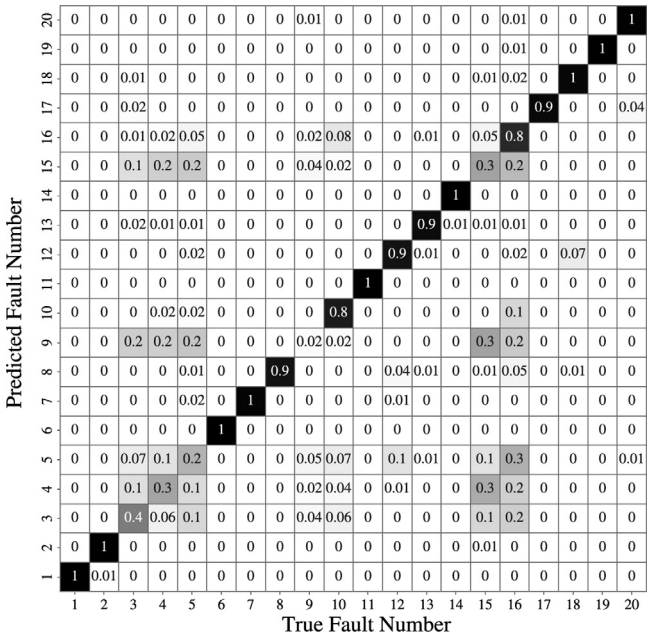


Figure 3.25 Confusion matrix from the convolutional neural network (CNN) prediction. Each column represents a true fault type, and each row represents a CNN predicted fault type. The entries along the diagonal are where the fault types are correctly classified. Most of the diagonal entries are close to 1, indicating that the CNN has good classification accuracy. *(Modified from [77] with permission from Wiley publishing).*

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

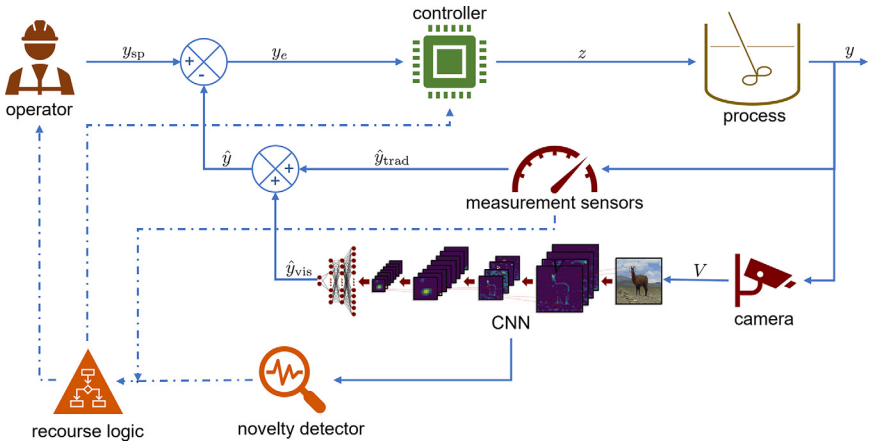


Figure 3.26 A feedback control system that incorporates a convolutional neural network (CNN) sensor to convert image data into a controllable measurement signal that can be used for feedback control. *(Modified from [79] with permission from Elsevier).*

summary of the work presented by Pulsipher et al. [79]. This work in turn draws upon existing emergent applications of image data in control, as those presented by Villalba-Diez et al. [80–83].

We consider leveraging a CNN sensor to autonomously map image data to a controllable state variable such that we achieve closed-loop control. With this, we remove the human operator from the control loop in the sense that they will no longer need to actively interpret and act upon visual process data. Such a control system is depicted in Fig. 3.26. Here, the camera and the CNN work together to form a computer vision sensor that is able to measure states y_{vis} that otherwise would not be available using traditional process measurement devices. Hence, following this new paradigm, we obtain a fully automatic control system that can exhibit improved setpoint tracking performance which promotes increased consistency and performance.

The paradigm shift from an operator-centric control system to the CNN-aided system of Fig. 3.26 introduces a significant vulnerability: poor prediction accuracy of y_{vis} when the image V is novel relative to the training data used to prepare the CNN (i.e., the CNN sensor makes a highly inaccurate prediction because it is extrapolating). Injecting erroneous measurement data into a feedback control architecture can have severe consequences. Thus, we require an appropriate novelty detection approach (depicted in Fig. 3.26) to automatically recognize in real-time when the visual data V is novel relative to the CNN sensor being used. Novelty detection denotes a set of unsupervised learning methods that differentiate between novel and

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

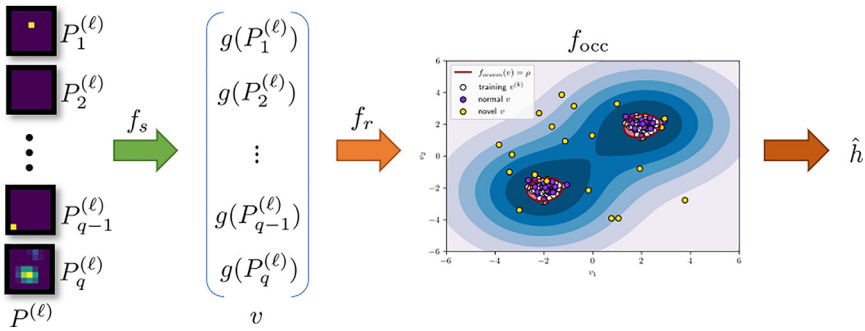


Figure 3.27 A summary of the sensor-activated feature extraction once-class classification (SAFE-OCC) novelty detection framework that operates on a feature map to produce a novelty signal. *(Modified from [79] with permission from Elsevier).*

normal data [84]. A couple of paradigms are reconstruction models and one-class classification. One-class classification (OCC) denotes an area of methods that learn a single class of normal instances from unlabeled training data (typically assumed to be comprised of normal instances). These then identify novel data instances by determining if they lie outside the learned class. Here, we discuss the sensor-activated feature extraction once-class classification (SAFE-OCC) novelty detection framework [79]. The SAFE-OCC framework leverages the native feature space of a CNN sensor to achieve novelty detection that is complimentary. The SAFE-OCC novelty detector involves three steps: feature extraction via the feature maps of a CNN sensor, feature refinement, and novelty detection via OCC (see Fig. 3.27).

We illustrate the application of SAFE-OCC to control the `CartPole-v1` environment from OpenAI-Gym [85], which corresponds to the classic cart-pole control problem introduced by Barto et al. [86]. Here, we seek to balance a pendulum above a cart which we can move either right or left at a mixed rate. We consider the angle of the pendulum (measured in degrees relative to vertical alignment) as the state variable γ (ignoring the position of the cart), and we take the cart movement direction to be the control variable $z \in \{0, 1\} \subset \mathbb{Z}$ (where 0 is left and 1 is right). Thus, we have a single-input single-output (SISO) process to control. With this simplification, we implement a PID controller with a derivative filter.

We conducted four simulations: a base case that uses unperturbed images and three others that invoke a particular simulated visual disturbance after 150 time steps. The three disturbance types are produced via `ImgAug` using the Fog, Spatter, and `Cutout` methods which correspond to fog, splattering,

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

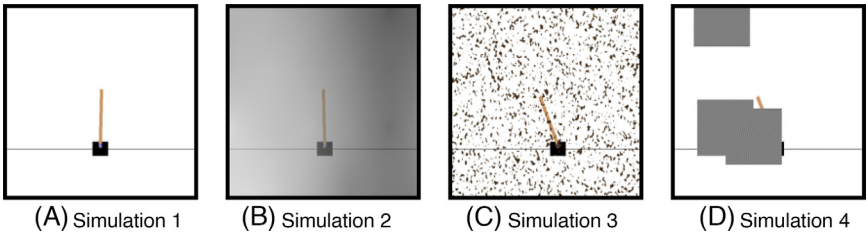


Figure 3.28 Representative snapshots from simulations used in the cart-pole case study. *(Modified from [79] with permission from Elsevier).*

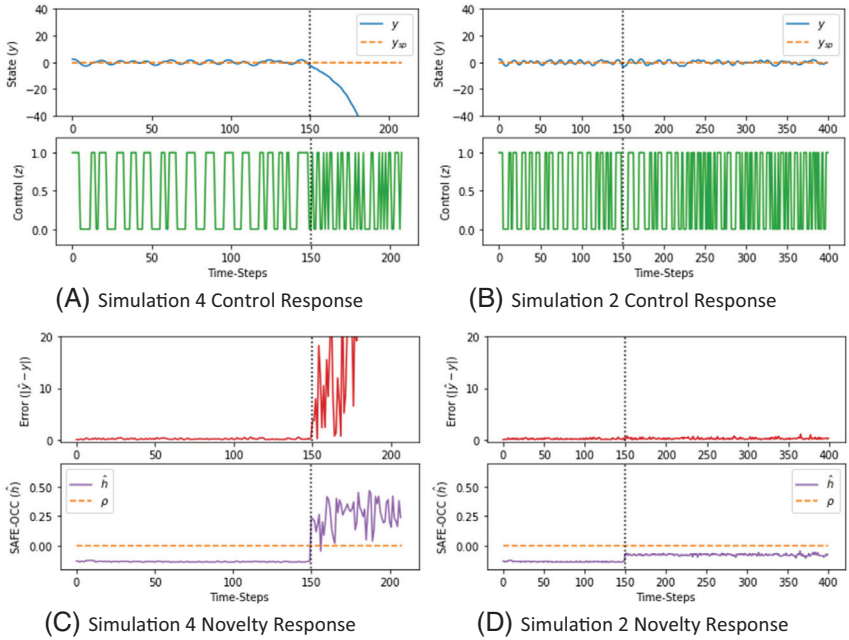


Figure 3.29 The control response trajectories of Simulations 2 (foggy disturbance) and 4 (blockage disturbance). The vertical dotted line at time-step 150 indicates when the fog disturbance is introduced in Simulation 2 and the blockage disturbance is introduced in Simulation 4. Effective control is maintained in Simulation 2 because these are normal images for the convolutional neural network (CNN). Ineffective control, however, is obtained in Simulation 4 because the images are novel to the CNN. The sensor-activated feature extraction once-class classification (SAFE-OCC) framework detects these novel images to prevent this type of behavior. *(Modified from [79] with permission from Elsevier).*

and square blockages, respectively. Fig. 3.28 shows representative images of these simulations. Fig. 3.29 shows the responses exhibited in Simulations 4 (blockage) and 2 (fog). In Fig. 3.29B, we observe that effective control in tracking the set point is achieved for both simulations. This behavior can be

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

attributed to the CNN sensor being trained with clear and fogged images which means that its predictions $\hat{y}$ incur a low error relative to y as shown in Fig. 3.29D. Fig. 3.29C shows the response in the fourth simulation; here, the blockage disturbance is *novel* relative to the CNN sensor and thus, significant prediction error is incurred in each case once the sensor is subjected to the disturbance. SAFE-OCC accurately identifies the novel images once they are injected into the CNN sensor and can avoid catastrophic control failure.

3.5 Conclusion

This chapter put into perspective the use of convolutional neural networks (CNNs) for extracting information from complex data sources that are commonly encountered in manufacturing. A set of selected case studies demonstrated how these machine learning (ML) techniques can be used to conduct a variety of tasks such as classification, anomaly detection, and prediction of properties.

The discussion outlined in this chapter is heavily biased by research conducted by the lead author's research group and is not meant to provide an exhaustive review. The field of ML learning is quickly evolving, with many different applications and techniques being developed. For instance, transformer models are nowadays being developed for computer vision and provide an alternative to address the limitations of CNNs [87]. Moreover, CNNs are actively being used to tackle a wide range of challenges arising in catalysis, materials, healthcare, and biology [88–90].

An area that is still in its infancy is the use of computer vision techniques to enable feedback control; specifically, closing the loop between image data and feedback control is a technical challenge. This is because computer vision signals are inherently infinite-dimensional objects (e.g., fields) that cannot be controlled directly by control systems. As such, it is necessary to extract key descriptors that effectively summarize these objects and to develop techniques for building dynamical models directly from computer vision signals. Techniques such as autoencoders, recurrent neural networks, and dynamic mode decomposition provide some alternatives [83,91]. In this context, it is also important to re-think how to design control architectures that can properly act on image data, given that most manufacturing systems have limited actuation. This is particularly relevant in 3D-printing and additive manufacturing applications, in which it is necessary to achieve a high level of precision in shaping 3D objects. Moreover, it is important to think about how to leverage these image data sources in the development

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

of physical models [92]. The use of hyperspectral imaging in manufacturing is also an exciting direction [93]; this type of data can reveal properties of systems, materials, and products that provide rich information about quality and health. A fundamental challenge in dealing with hyperspectral imaging, however, is its inherent high dimensionality. Specifically, hyperspectral images contain many color channels and are challenging to process using CNNs.

Another exciting area of current research is speech recognition [94]; one can envision a future in which instructions are provided to an automation system in the form of voice or in which the automation system summarizes the behavior of the system in narrative form or explains the reasoning behind a decision. Along these lines, the use of audio signals (e.g., as those obtained from vibration sensors) is currently being investigated to detect faults [95].

Acknowledgment

We acknowledge funding from the U.S. National Science Foundation under BIGDATA grant IIS-1837812 and support from the members of the Texas-Wisconsin-California Control Consortium (TWCCC).

References

[1] K. Fukushima, Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, Biol. Cybern. 36 (4) (1980) 193–202.
[2] Y. Le Cun, L.D. Jackel, B. Boser, J.S. Denker, H.P. Graf, I. Guyon, D. Henderson, R.E. Howard, W. Hubbard, Handwritten digit recognition: applications of neural network chips and automatic learning, IEEE Commun. Magaz. 27 (11) (1989) 41–46.
[3] H.A. Rowley, S. Baluja, T. Kanade, Neural network-based face detection, IEEE Trans. Pattern Anal. Mach. Intell. 20 (1) (1998) 23–38.
[4] J. Nickolls, I. Buck, M. Garland, K. Skadron, Scalable parallel programming with Cuda, Queue 6 (2) (2008) 40–53.
[5] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Advances in Neural Information Processing Systems, Curan Associates Inc., 2012, pp. 1097–1105.
[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, Going deeper with convolutions, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.
[7] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition. In: arXiv preprint arXiv:1409.1556 (2014).
[8] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[9] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, Advances in Neural Information Processing Systems, MIT Press, 2015, pp. 91–99.
[10] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[11] K. Simonyan, A. Zisserman, Two-stream convolutional networks for action recognition in videos, Advances in Neural Information Processing Systems, MIT Press, 2014, pp. 568–576.
[12] S. Ji, W. Xu, M. Yang, K. Yu, 3D convolutional neural networks for human action recognition, IEEE Trans. Pattern Anal. Mach. Intell. 35 (1) (2012) 221–231.
[13] Computer Vision Machine Learning Team, An on-device deep neural network for face detection, Apple Machine Learning Research. https://machinelearning.apple.com/research/face-detection, 2017. (Accessed 23 July 2021).
[14] M. Hirohara, Y. Saito, Y. Koda, K. Sato, Y. Sakakibara, Convolutional neural network based on smiles representation of compounds for detecting chemical motif, BMC Bioinformat. 19 (19) (2018) 526.
[15] W. Xie, J.A. Noble, A. Zisserman, Microscopy cell counting and detection with fully convolutional regression networks, Comput. Methods Biomech. Biomed. Eng.: Imaging Vis. 6 (3) (2018) 283–292.
[16] D.K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, R.P. Adams, Convolutional networks on graphs for learning molecular fingerprints, Adv. Neural. Inf. Process. Syst. 28 (2015).
[17] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: International Conference on Machine Learning, PMLR, 2017, pp. 1263–1272.
[18] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[19] L. Je, G.W. Huber, R.C. Van Lehn, V.M. Zavala, On the integration of molecular dynamics, data science, and experiments for studying solvent effects on catalysis, Curr. Opin. Chem. Eng. 36 (100796) (2022).
[20] K. Yang, K. Swanson, W. Jin, C. Coley, P. Eiden, H. Gao, A. Guzman-Perez, T. Hopper, B. Kelley, M. Mathea, et al., Analyzing learned molecular representations for property prediction, J. Chem. Infor. Model. 59 (8) (2019) 3370–3388.
[21] M.M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, P. Vandergheynst, Geometric deep learning: going beyond euclidean data, IEEE Signal Process. Mag. 34 (4) (2017) 18–42.
[22] A.D. Smith, P. Dłotko, V.M. Zavala, Topological data analysis: concepts, computation, and applications in chemical engineering, Comput. Chem. Eng. 146 (107202) (2021).
[23] A. Smith, V.M. Zavala, The Euler characteristic: a general topological descriptor for complex data, Comput. Chem. Eng. 154 (107463) (2021).
[24] R. Hirsch, Exploring Colour Photography: A Complete Guide, Laurence King Publishing, 2004.
[25] R.W.G. Hunt, The Reproduction of Colour, John Wiley & Sons, 2005.
[26] C.-I. Chang, Hyperspectral Imaging: Techniques for Spectral Detection and Classification, 1, Springer Science & Business Media, 2003.
[27] B. Lu, P.D. Dao, J. Liu, Y. He, J. Shang, Recent advances of hyperspectral imaging technology and applications in agriculture, Remote Sens. 12 (16) (2020) 2659.
[28] B. Fei, Hyperspectral imaging in medical applications, Data Handling in Science and Technology, 32, Elsevier, 2020, pp. 523–565.
[29] M. Manley, Near-infrared spectroscopy and hyperspectral imaging: non-destructive analysis of biological materials, Chem. Soc. Rev. 43 (24) (2014) 8200–8214.
[30] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: a review of methods and applications, AI Open 1 (2020) 57–81.
[31] V. Nair, G.E. Hinton, Rectified linear units improve restricted boltzmann machines, in: Proceedings of the 27th International Conference on Machine Learning (ICML-10), Curran Associates, Inc., 2010, pp. 807–814.
[32] J. Nagi, F. Ducatelle, G.A. Di Caro, D. Cires¸an, U. Meier, A. Giusti, F. Nagi, J. Schmidhuber, L.M. Gambardella, Max-pooling convolutional neural networks for vision-based

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

hand gesture recognition, in: 2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA), IEEE, 2011, pp. 342–347.
[33] F. Chollet, Deep Learning with Python, Simon and Schuster, 2017.
[34] A. Géron, Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, O'Reilly Media, 2019.
[35] A.B. Jung, K. Wada, J. Crall, S. Tanaka, J. Graving, C. Reinders, S. Yadav, J. Banerjee, G. Vecsei, A. Kraft, Z. Rui, J. Borovec, C. Vallentin, S. Zhydenko, K. Pfeiffer, B. Cook, I. Fernndez, F.-M. De Rainville, C.-H. Weng, A. Ayala-Acevedo, R. Meudec, M. Laporte, et al. Imgaug. https://github.com/aleju/imgaug, 2020. (Accessed 23 July 2021).
[36] E. Real, S. Moore, A. Selle, S. Saxena, Y.L. Suematsu, J. Tan, Q. Le, A. Kurakin, Large-scale evolution of image classifiers. In: arXiv preprint arXiv:1703.01041, (2017). https://arxiv.org/abs/1703.01041.
[37] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A.L. Yuille, L. Fei-Fei, Auto-deeplab: hierarchical neural architecture search for semantic image segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Curran Associates, Inc., 2019, pp. 82–92.
[38] Y. Fan, F. Tian, Y. Xia, T. Qin, X.-Y. Li, T.-Y. Liu, Searching better architectures for neural machine translation, IEEE/ACM Transactions on Audio, Speech, and Language Processing 28 (2020) 1574–1585.
[39] R. Maulik, R. Egele, B. Lusch, P. Balaprakash. Recurrent neural network architecture search for geophysical emulation. In arXiv preprint arXiv:2004.10928 (2020). https://arxiv.org/abs/2004.10928.
[40] S. Jiang, J. Noh, C. Park, A.D. Smith, N.L. Abbott, V.M. Zavala, Using machine learning and liquid crystal droplets to identify and quantify endotoxins from different bacterial species, Analyst 146 (4) (2021) 1224–1233.
[41] F. Borek, Handbook of endotoxin, vol. 1, chemistry of endotoxin, J. Immunolog. Methods 82 (2) (1985).
[42] A.D. Smith, N. Abbott, V.M. Zavala, Convolutional network analysis of optical micrographs for liquid crystal sensors, J. Phys. Chem. C 124 (28) (2020) 15152–15161.
[43] R.R. Shah, N.L. Abbott, Principles for measurement of chemical exposure based on recognition-driven anchoring transitions in liquid crystals, Science 293 (5533) (2001) 1296–1299.
[44] D. Mulder, A. Schenning, C. Bastiaansen, Chiral-nematic liquid crystals as one dimensional photonic materials in optical sensors, J. Mater. Chem. C 2 (33) (2014) 6695–6705.
[45] P. Ireland, T. Jones, Liquid crystal measurements of heat transfer and surface shear stress, Meas. Sci. Technol. 11 (7) (2000) 969.
[46] K.-L. Yang, K. Cadwell, N.L. Abbott, Use of self-assembled monolayers, metal ions and smectic liquid crystals to detect organophosphonates, Sens. Actuators B: Chem. 104 (1) (2005) 50–56.
[47] J. Huuskonen, M. Salo, J. Taskinen, Aqueous solubility prediction of drugs based on molecular topology and neural network modeling, J. Chem. Inf. Comput. 38 (3) (1998) 450–456.
[48] A. Lusci, G. Pollastri, P. Baldi, Deep architectures and deep learning in chemoinformatics: the prediction of aqueous solubility for drug-like molecules, J. Chem. Infor. Model. 53 (7) (2013) 1563–1575.
[49] S. Boobier, D.R. Hose, A.J. Blacker, B.N. Nguyen, Machine learning with physicochemical relationships: solubility prediction in organic solvents and water, Nat. Commun. 11 (1) (2020) 1–10.
[50] A. Mayr, G. Klambauer, T. Unterthiner, S. Hochreiter, Deeptox: toxicity prediction using deep learning, Front. Environ. Sci. 3 (2016) 80.
[51] P. Banerjee, A.O. Eckert, A.K. Schrey, R. Preissner, Protox-ii: a webserver for the prediction of toxicity of chemicals, Nucleic Acids Res. 46 (W1) (2018) W257–W263.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[52] J. Jiang, R. Wang, G.-W. Wei, Ggl-tox: geometric graph learning for toxicity prediction, J. Chem. Infor. Model. 61 (4) (2021) 1691–1700.
[53] T. Schroeter, A. Schwaighofer, S. Mika, A. Ter Laak, D. Suelzle, U. Ganzer, N. Heinrich, K.-R. Mu¨ ller, Machine learning models for lipophilicity and their domain of applicability, Mol. Pharmaceutics 4 (4) (2007) 524–538.
[54] B. Tang, S.T. Kramer, M. Fang, Y. Qiu, Z. Wu, D. Xu, A self-attention based message passing neural network for predicting molecular lipophilicity and aqueous solubility, J. Cheminformat. 12 (1) (2020) 1–9.
[55] H.L. Morgan, The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service, J. Chem. Doc. 5 (2) (1965) 107–113.
[56] D. Rogers, M. Hahn, Extended-connectivity fingerprints, J. Chem. Infor. Model. 50 (5) (2010) 742–754.
[57] M. Karelson, V.S. Lobanov, A.R. Katritzky, Quantum-chemical descriptors in qsar/qspr studies, Chem. Rev. 96 (3) (1996) 1027–1044.
[58] H. Moriwaki, Y.-S. Tian, N. Kawashita, T. Takagi, Mordred: a molecular descriptor calculator, J. Cheminformat. 10 (1) (2018) 1–14.
[59] Y.-C. Lo, S.E. Rensi, W. Torng, R.B. Altman, Machine learning in chemoinformatics and drug discovery, Drug Discovery Today 23 (8) (2018) 1538–1546.
[60] S. Qin, T. Jin, R.C. Van Lehn, V.M. Zavala, Predicting critical micelle concentrations for surfactants using graph convolutional neural networks, J. Phys. Chem. B 125 (37) (2021) 10610–10620.
[61] J.N. Israelachvili, Intermolecular and Surface Forces, Academic press, 2011.
[62] M.J. Rosen, J.T. Kunjappu, Surfactants and Interfacial Phenomena, John Wiley & Sons, 2012.
[63] K.C. Cheng, Z.S. Khoo, N.W. Lo, W.J. Tan, N.G. Chemmangattuvalappil, Design and performance optimisation of detergent product containing binary mixture of anionic-nonionic surfactants, Heliyon 6 (5) (2020) e03861.
[64] T. Gaudin, P. Rotureau, I. Pezron, G. Fayet, New qspr models to predict the critical micelle concentration of sugar-based surfactants, Ind. Eng. Chem. Res. 55 (45) (2016) 11716–11726.
[65] N. Scholz, T. Behnke, U. Resch-Genger, Determination of the critical micelle concentration of neutral and ionic surfactants with fluorometry, conductometry, and surface tensiona method comparison, J. Fluoresc. 28 (1) (2018) 465–476.
[66] A. Fluksman, O. Benny, A robust method for critical micelle concentration determination using coumarin-6 as a fluorescent probe, Anal. Methods 11 (30) (2019) 3810–3818.
[67] C.G. Gahan, S.J. Patel, M.E. Boursier, K.E. Nyffeler, J. Jennings, N.L. Abbott, H.E. Blackwell, R.C. Van Lehn, D.M. Lynn, Bacterial quorum sensing signals self-assemble in aqueous media to form micelles and vesicles: an integrated experimental and molecular dynamics study, J. Phys. Chem. B 124 (18) (2020) 3616–3628.
[68] P. Mukerjee, K.J. Mysels, Technical report, National Standard Reference Data System, 1971.
[69] L. Van der Maaten, G. Hinton, Visualizing data using t-sne, J. Mach. Learn. Res. 9 (11) (2008).
[70] S. Jiang, Z. Xu, M. Kamran, S. Zinchik, S. Paheding, A.G. McDonald, E. Bar-Ziv, V.M. Zavala, Using atr-ftir spectra and convolutional neural networks for characterizing mixed plastic waste, Comput. Chem. Eng. 155 (107547) (2021).
[71] H. Ritchie, M. Roser, Plastic pollution, Our World in Data, 2018. https://ourworldindata.org/plastic-pollution.
[72] X. Chen, Q. Chai, N. Lin, X. Li, W. Wang, 1D convolutional neural network for the discrimination of aristolochic acids and their analogues based on near-infrared spectroscopy, Anal. Methods 11 (40) (2019) 5118–5125.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[73] Z. Wang, T. Oates, Encoding time series as images for visual inspection and classification using tiled convolutional neural networks, in: Workshops at the Twenty-ninth AAAI Conference on Artificial Intelligence, AAAI Press, 2015.
[74] S. Zinchik, S. Jiang, S. Friis, F. Long, L. Høgstedt, V.M. Zavala, E. Bar-Ziv, Accurate characterization of mixed plastic waste using machine learning and fast infrared spectroscopy, ACS Sustain. Chem. Eng. 9 (42) (2021) 14143–14151.
[75] J.J. Downs, E.F. Vogel, A plant-wide industrial process control problem, Comput. Chem. Eng. 17 (3) (1993) 245–255.
[76] H. Wu, J. Zhao, Deep convolutional neural network model based chemical process fault diagnosis, Comput. Chem. Eng. 115 (2018) 185–197.
[77] S. Jiang, V.M. Zavala, Convolutional neural nets in chemical engineering: foundations, computations, and applications, AIChE J. 67 (9) (2021) e17282.
[78] R. Cory A., A. Ben D., T. Randy, C. Maia B., Additional tennessee eastman process simulation data for anomaly detection evaluation, Harvard Dataverse, V1, 2017. https://doi.org/10.7910/DVN/6C3JR1.
[79] J.L. Pulsipher, L.D. Coutinho, T.A. Soderstrom, V.M. Zavala, Safe-occ: A novelty detection framework for convolutional neural network sensors and its application in process control, J. Process Control 117 (2022) 78–97.
[80] J. Villalba-Diez, D. Schmidt, R. Gevers, J. Ordieres-Meré, M. Buchwitz, W. Wellbrock, Deep learning for industrial computer vision quality control in the printing industry 4.0, Sensors 19 (18) (2019) 3987.
[81] A. Martynenko, Computer vision for real-time control in drying, Food Eng. Rev. 9 (2) (2017) 91–111.
[82] B.A. Rizkin, K. Popovich, R.L. Hartman, Artificial neural network control of thermoelectrically-cooled microfluidics using computer vision based on ir thermography, Comput. Chem. Eng. 121 (2019) 584–593.
[83] Q. Lu, V.M. Zavala, Image-based model predictive control via dynamic mode decomposition, J. Process Control. 104 (2021) 146–157.
[84] L. Ruff, J.R. Kauffmann, R.A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T.G. Dietterich, K.-R. Müller, A unifying review of deep and shallow anomaly detection, in: Proceedings of the IEEE, IEEE, 2021.
[85] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, OpenAI gym. In: arXiv preprint arXiv:1703.01041, (2017). https://arxiv.org/abs/1606.01540.
[86] A.G. Barto, R.S. Sutton, C.W. Anderson, Neuronlike adaptive elements that can solve difficult learning control problems, IEEE Trans. Syst. Man Cybern. Syst. (5) (1983) 834–846.
[87] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, S. Zagoruyko, End-to-end object detection with transformers, in: European Conference on Computer Vision, Springer, 2020, pp. 213–229.
[88] H.A. Haenssle, C. Fink, R. Schneiderbauer, F. Toberer, T. Buhl, A. Blum, A. Kalloo, A.B.H. Hassen, L. Thomas, A. Enk, et al., Man against machine: diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists, Ann. Oncol. 29 (8) (2018) 1836–1842.
[89] A.E. Bruno, P. Charbonneau, J. Newman, E.H. Snell, D.R. So, V. Vanhoucke, C.J. Watkins, S. Williams, J. Wilson, Classification of crystallization outcomes using deep convolutional neural networks, PLoS One 13 (6) (2018) e0198883.
[90] A. Cecen, H. Dai, Y.C. Yabansu, S.R. Kalidindi, L. Song, Material structure-property linkages using three-dimensional convolutional neural networks, Acta Mater. 146 (2018) 76–84.
[91] D. Masti, A. Bemporad, Learning nonlinear state–space models using autoencoders, Automatica 129 (109666) (2021).

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[92] J. Li, R. Jin, Z.Y. Hang, Integration of physically-based and data-driven approaches for thermal field prediction in additive manufacturing, Mater. Design 139 (2018) 473–485.
[93] J. Bøtker, J.X. Wu, J. Rantanen, Hyperspectral imaging as a part of pharmaceutical product design, Data Handling in Science and Technology, 32, Elsevier, 2020, pp. 567–581.
[94] T. Dimitrov, C. Kreisbeck, J.S. Becker, A. Aspuru-Guzik, S.K. Saikin, Autonomous molecular design: then and now, ACS Appl. Mater. Interf. 11 (28) (2019) 24825–24836.
[95] R.M. Scheffel, A.A. Fröhlich, M. Silvestri, Automated fault detection for additive manufacturing using vibration sensors, Int. J. Comput. Integr. Manufact. 34 (5) (2021) 500–514.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

CHAPTER FOUR

Sparse mathematical programming for fundamental learning of governing equations

Fernando Lejarza[a] and Michael Baldea[a,b]
[a]McKetta Department of Chemical Engineering, The University of Texas at Austin, TX, United States
[b]Oden Institute for Computational Engineering and Sciences, The University of Texas at Austin, TX, United States

4.1 Introduction

The use of mathematical models is now prevalent across all manufacturing industries. They are essential decision-support tools throughout the life of manufacturing facilities, starting from the design phase, continuing to the engineering and construction steps; production startup, control, and optimization; fault detection and mitigation; supply chain management, and ultimately through decommissioning and site remediation. Likewise, mathematical models support the development of new products to be made in these facilities. Fundamentally, models can be derived from first-principles arguments (in general, manufacturing-related decisions such as the ones listed above tend to rely on macroscopic models that comprise material and energy balances) or from data collected from a physical system (be it an entire production facility or a laboratory-scale experiment). These modeling avenues are not mutually exclusive, as first-principles models include parameters that are determined experimentally, while the functional form of data-driven models is often chosen using physical arguments.

The past decades have witnessed several favorable trends from the point of view of building, validating, and deploying models to guide manufacturing activities. The lower costs for sensors and data storage have allowed for massive troves of plant data to be gathered. Computing equipment performance has increased dramatically, and significant advances have been made in analysis, simulation, and optimization algorithms, that are often implemented in user-friendly packages. Together, these developments have democratized the ability to extract information from data, build larger and more detailed first-principles models, and leverage their outputs either offline or in real-time to improve all aspects of the manufacturing lifecycle.

Artificial Intelligence in Manufacturing: Concepts and Methods.
DOI: https://doi.org/10.1016/B978-0-323-99134-6.00002-5
Copyright © 2024 Elsevier Inc.
All rights are reserved, including those for text and data mining, AI training, and similar technologies.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY
Copyright Elsevier 2024

It has been argued that a new industrial revolution has been triggered, referred to as "Smart Manufacturing" or "Industry 4.0." Mathematical modeling lies at the center of this revolution, with the concept of a "digital twin" being used to denote an in-silico representation of the behavior of a physical system (either existing or to be built) at all the relevant time and space scales, that is sufficiently accurate and yet sufficiently parsimonious to support all decision-making activities mentioned earlier, particularly those that are carried out online and in real-time. The smart manufacturing literature [1,2] is not prescriptive regarding the derivation and format of digital twins. Nevertheless, in addition to the aforementioned accuracy and computational speed needs, it is intuitive that such models should be (1) easy to derive or at least easy to replicate and apply to systems or cases that (closely) resemble the first-of-a-kind instance from which they were derived, (2) generalizable to systems that are similar but not close to the original instance (equivalently, these models should extrapolate), and (3) interpretable and transparent to their user base. These requirements suggest that first-principles arguments and representations can and should be used in deriving digital twins, either alone or in conjunction with data-driven techniques.

Within the realm of first-principles models, *governing equations* play a particularly important role. We define them as minimal-order representations of the fundamental behavior of a system; as such, they constitute the building blocks of generalizable and parsimonious first-principles mathematical models and of "digital twins." While governing equations have in many cases found interpretation as the macroscopic closures of theoretical (e.g., kinetic) arguments at the microscopic level, they have been typically derived (or discovered) by analyzing data sets collected from careful experimentation on prototypical macroscopic physical systems. Historically, the discovery process has been slow and deliberate, and may involve a fortuitous *Eureka!* moment experienced by some uniquely gifted scientists and engineers. The potential for leveraging data analysis techniques and machine learning tools for fundamental learning, that is for using data obtained from a physical system to extract governing equations rather than to construct approximate/surrogate models, has been recognized only recently [3,4]. The appeal of this discovery pathway is very high and the development of new techniques is proceeding apace.

The purpose of this chapter is to provide an overview of the status of tools that leverage machine learning to discover governing equations from data. We will also identify current gaps and shortcomings, and suggest avenues for

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

further development. Illustrative examples will be presented to emphasize the applicability of these techniques to the development of digital twins that are relevant to smart manufacturing.

4.2 Problem definitions

In this section, we provide definitions for the problem at hand (i.e., distilling governing equations from data), and contrast it with the related (but notably different) tasks of surrogate modeling, system identification, and parameter estimation.

4.2.1 Distilling governing equations

In general terms, we consider a governing equation as a mathematical expression of the evolution of the states of a system over a certain domain (e.g., temporal, spatial) given by:

$$\mathcal{D}\mathbf{x} = \mathrm{f}(\mathrm{x}, \mathrm{u}, \mathrm{p}) \tag{4.1}$$

where, $\mathrm{x} \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ are the system states, $\mathrm{u} \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$ are the system (manipulated) inputs, $\mathrm{p} \in \mathcal{P} \subseteq \mathbb{R}^{n_p}$ are a set of parameters, and $\mathcal{D} : \mathcal{X} \rightarrow \mathbb{R}^{n_x}$ represents the differential operator applied on the state variables, e.g.,

$$\mathcal{D} = \frac{d}{dt}(\cdot)$$

denotes an instance of an ordinary differential equation over the time domain, while

$$\mathcal{D} = \frac{\partial}{\partial t}(\cdot) + \sum_{j=1}^{n_\epsilon} \frac{\partial}{\partial \epsilon_j}(\cdot)$$

denotes an instance of a partial differential equation (PDE) and $\epsilon \in \mathbb{R}^{n_\epsilon}$ denotes, e.g., spatial coordinates. It is assumed that the vector field $\mathrm{f} : \mathcal{X} \times \mathcal{U} \times \mathcal{P} \rightarrow \mathbb{R}^{n_x}$ is sufficiently smooth, and that the true form of the governing equation in Eq. (4.1) is unknown.

In this context, the fundamental learning problem we are interested in is defined as follows: given a set of (noisy) measurements of the state and input variables, denoted respectively as $\hat{\mathrm{x}}\left(\hat{t}_k, \hat{\epsilon}_k\right)$ and $\hat{\mathrm{u}}\left(\hat{t}_k, \hat{\epsilon}_k\right)$ available at domain points $\left\{\left(\hat{t}_k, \epsilon_k\right)\right\}_{k=1}^{m}$, find the (nonlinear) governing equations in the form of Eq. (4.1) ensuring that they have high predictive accuracy (for some error metric of choice) and are of low complexity (i.e., the function $f(\cdot)$ and operator $\mathcal{D}$ have as few terms/elements as possible). We emphasize that we

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

envision this task as being separate and different from related established and well-studied problems such as surrogate modeling, system identification, and parameter estimation, to which we contrast fundamental learning next.

Remark 1. Not all the system states may be observable in practice, calling for a more general representation of the system by considering a vector of system outputs denoted by $y \in \mathcal{Y} \subseteq \mathbb{R}^{n_y}$. The relationship between inputs, states, and outputs is possibly governed by some other (nonlinear) function of the form of $y = g(x, u, q)$ for some set of parameters q. State observability is not explicitly addressed in this work, and thus for now the assumption that $y = x$ (i.e., full state measurement) is made, particularly for comparing against well-established system identification methods. Relaxing this observability assumption will be discussed in further detail in a later section.

4.2.2 Surrogate modeling and system identification

When there is limited knowledge available regarding the (often complex) phenomena governed by Eq. (4.1) (i.e., to the extent that related functional forms and differential operators cannot be determined), surrogate modeling attempts to represent the unknown function $\mathbf{f}(\cdot)$ with an arbitrary approximation $\hat{\mathbf{f}}(\cdot) \sim \mathbf{f}(\cdot)$ using the available training data. The term *system identification* is typically used when a surrogate model is employed to approximate the behavior of a dynamical system, like the one in the governing equation in Eq. (4.1), by capturing the relationship between inputs and outputs. As an example, the general structure of a surrogate model in discrete time (for simplicity without spatial dependence over coordinates $\in$) can be of the form:

$$\mathbf{x}(t_k) = \hat{\mathbf{f}}\left(\mathbf{x}\left(t_{k-\tau_x}\right), \mathbf{u}\left(t_{k-\tau_u}\right), t_k, \hat{\mathbf{p}}\right) \tag{4.2}$$

where, $\tau_x, \tau_u \in \mathbb{R}_{\geq 0}$ indicate the lag in the data being used to predict the states at a future time point t_k, noting that the number of lagged terms considered in the model might be different for the states and the input variables (i.e., $\tau_x \neq \tau_u$). For models of the form of Eq. (4.2), the state/output variables might be directly modeled without attempting to estimate their derivative(s) as in Eq. (4.1), for which there are typically no direct measurements available. Then, the problem at hand becomes fitting the parameters $\hat{\mathrm{p}}$ (as well as determining the best choice of τ_x,τ_u) of the chosen model $\hat{\mathbf{f}}(\cdot)$, which has a known structure and convenient functional form in Eq. (4.2). Model fitting is typically formulated such that the difference between the measured data $\hat{\mathrm{x}}$ and predictions of $\hat{\mathbf{f}}(\cdot)$ is minimized in a norm

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

sense. Model parsimony is desirable, for which regularization (by means of penalizing the training objective by some function of the coefficient magnitudes) can be employed, and minimal-order models (with the lowest number of state variables) are generally targeted (dimensionality reduction can be performed on the training data, via, e.g., principal component analysis or neural network embeddings, to identify new coordinates or state variables $\hat{\mathcal{X}}$ on which to train the surrogate model). It should be noted that the functional form of the identified model $\hat{\mathbf{f}}(\cdot)$ and the governing equations in Eq. (4.1) could be indeed quite different. This observation implies that surrogate models and system identification approaches are often "black-box" in nature, and provide little physical insight regarding the intrinsic dynamics of the system under study.

There is a wide range of surrogate model options and selecting the correct one (and optimally tuning the associated hyperparamters) is often a difficult task. In this case, we refer to model hyperparameters as external decisions and parameters that dictate the configuration of the model and are not learned during the training process (e.g., the number of hidden layers in a neural network, the tree depth and number of estimators in tree ensembles like random forests or gradient boosted trees). A common approach for tuning machine learning model hyperparameters is through Bayesian optimization techniques which are reviewed in the work of Frazier [5]. The correct model choice depends on multiple factors such as problem dimensions (e.g., number of state variables), data type (e.g., features being discrete or continuous, data being static or dynamic), data quality (e.g., measurement noise and sampling frequency) and data quantity (e.g., number of observations across the entire feature and output spaces) of available for training and testing. Common examples of surrogate models include linear regression models, linear state-space models, nonlinear Hammerstein-Wiener models, neural networks architectures, decision tress (and ensembles thereof). In the time-series forecasting literature [6], autoregressive models having a form similar to Eq. (4.2), for example, capture future variations in the state variables as a function of their own previous (i.e., lagged) values and a residual term. For example, an autogressive model with extra input (ARX) has the following structure:

$$\mathbf{x}(t_k) = a_1\mathbf{x}(t_{k-1}) + \cdots + a_n\mathbf{x}(t_{k-n}) + b_1\mathbf{u}\left(t_{k-\eta}\right) + \cdots + b_m\mathbf{u}\left(t_{k-\eta-m+1}\right) + \varepsilon(t_k)$$

where, a and b correspond to the parameters to estimate from data, n and m define the order of the ARX model (i.e., the number of lagged terms which is typically considered a hyperparaemter and is determined via some from

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

of cross-validation), η defines the time delay or the dead time of the system, and $\varepsilon \in \mathbb{R}^{n_x}$ corresponds to a white-noise disturbance.

4.2.3 Parameter estimation

In contrast to surrogate modeling and system identification, *parameter estimation* assumes that the functional form of the governing Eq. (4.1) is known exactly (for both differential operator $\mathcal{D}$ and function $\mathbf{f}(\cdot)$) while parameters p are unknown. In such cases, when the intrinsic physics of the system are available, training data are used to estimate the values of the associated parameters, again, by minimizing in the norm sense the error between state measurements and model predictions. A well-established example of parameter estimation in the chemical manufacturing industry is fitting the kinetic parameters for a set of known reactions using available experimental or process data [7]. In this case, the state variables correspond to the species concentration (potentially at different reaction temperatures) and the dynamics correspond to the species balances dictated by the stoichiometry of the set of reactions. While surrogate modeling typically lacks physical interpretability, parameter estimation as defined here assumes complete knowledge and understanding of the phenomena underpinning the evolution of the system (e.g., the exact mechanism dictating the transformation of reactants to products), and are thus both different from the intent of fundamental learning task as defined above.

Remark 2. Established methods in statistics, data analytics, and estimation can be leveraged synergistically with fundamental learning frameworks, as well as surrogate modeling and parameter estimation, to develop more accurate models and governing equations. For example, identifying (and removing) outliers in the training data, as well as de-noising measurements can significantly beneft the training procedure. As will be illustrated in a subsequent section, correlation analysis, ordinary least squares regression, and filtering are employed to preprocess training data for discovering governing equations by Lejarza and Baldea [8].

4.3 Physics-informed machine learning

4.3.1 General principle

A natural first step toward using machine learning techniques to discover governing equations is to imbue the learning algorithm with some physical insight. Physics-informed machine learning frameworks [9] are a form of

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

"grey-box" models that aim to integrate fundamental physical laws and domain knowledge in the form of "informative priors" during the training procedure. The resulting schemes thus improve the interpretability of the discovered dynamics and reduce extrapolation bias that may result in poor generalization performance. There are several avenues for making a machine learning (ML) algorithm "physics-informed" and for introducing appropriate biases that can direct the learning process to identifying physically consistent models [9]: (1) observational biases that involve using data that embodies the underlying system dynamics or that are carefully processed using data augmentation procedures, (2) inductive biases that correspond to modifications to the ML architecture such that the inferred governing equations satisfy a given set of mathematical constraints that reflect domain knowledge (e.g., conservation laws), and (3) learning biases that impose such constraints in a soft manner in the form of penalties to the loss function minimized during training.

Among physics-informed machine learning frameworks, neural networks (NN), and in particular deep learning architectures, have witnessed growing popularity [9]. Neural networks have long been recognized as universal function approximators since, given sufficient complexity (e.g., in the form of a number of hidden layers, number of neurons per layer, or particular choices of activation functions), NNs can approximate a given nonlinear function arbitrarily well, and thus are an enticing modeling tool for capturing the dynamics of complex systems arising in scientific applications. Furthermore, the flexibility and ease of implementation offered by open-source deep learning software (e.g., PyTorch [10] and TensorFlow [11]) streamline the coding requirements for building NN architectures regularized by informative priors. Physics-informed neural networks (PINNs) [12] are a recent and representative example of deep learning frameworks that seamlessly integrate data and abstract mathematical operators to generate accurate solutions to PDEs and/or uncover unknown dynamics (a form of the learning biases discussed above).

An example PINN architecture is illustrated in Fig. 4.1. In this case, the first NN used in PINN (left-most network shown in Fig. 4.1) contains no physical knowledge about the system and serves as a general surrogate model of the solution $x(\epsilon, u, t)$ to some (partial) differential equation. The subsequent part of the network (right-most layers in Fig. 4.1) represents the PDE residual, whose loss can be thought of as a regularization penalty that is added to the loss function from the previous "physics-uninformed" layer [9,12]. In training PINNs, a linear combination of the two corresponding

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

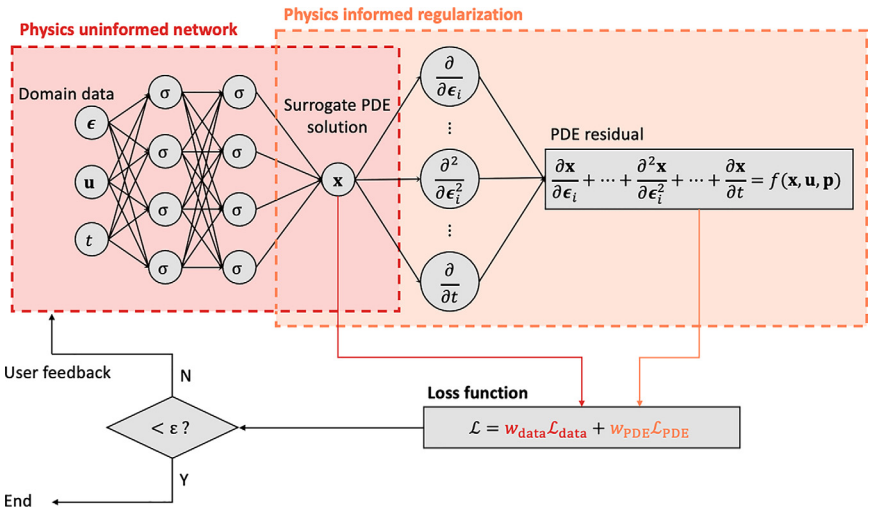


Figure 4.1 Illustration of physics-informed neural networks (PINNs) (modified from [9]). The two main components include a generic neural network to predict the system states *x*, as well as a regularization loss term to include physical knowledge. In addition to the network weights and biases, parameters *p* can also be learned when minimizing the loss function if measurements of the states are provided over the domain covered by ϵ and time *t*.

loss functions ($\mathcal{L}_{\text{data}}$ and $\mathcal{L}_{\text{PDE}}$) is minimized (e.g., by using variants of the gradient descent algorithm), and by employing weights ($w_{\text{data}} \in \mathbb{R}_{\geq 0}$ and $w_{\text{PDE}} \in \mathbb{R}_{\geq 0}$) that balance their relative importance.

Physics-informed neural networks have been successfully demonstrated for finding high-accuracy solutions to PDEs with known parameterizations, as well as for infering the PDE parameters that best describe the observed data. PINNs have been shown to be robust when trained on coarse and noisy data [12]. However, like all deep learning architectures, PINNs and related schemes require extensive training data sets, and suffer from a lack of transparency and interpretability regarding the solution $\mathrm{x}(\epsilon, \mathrm{u}, t)$ that is constructed from the observed data. Their use in decision-making and optimization routines is yet to be demonstrated. Further, PINNs rely on substantial structural assumptions and significant domain knowledge regarding the underlying system equations (e.g., the PDE operators used for NN regularization are known *a priori*) and their discovery task is thus more akin to a parameter estimation problem. Last but not least, it has been previously observed that for certain structural forms of the underlying PDEs, PINNs might be indeed quite difficult (and even fail) to train [13,14].

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

4.3.2 Example application of PINN to the Burgers equation

We consider an example implementation of the PINNs framework to compute a data-driven solution to the well-known one-dimensional Burgers equation over space and time. The Burgers equation, which can be derived from the Navier–Stokes equation, is used for modeling fluid flow and gas dynamics, which are relevant to several manufacturing processes (e.g., liquid and gas flow in pipes). The PDE and example boundary and initial conditions are given by:

$$\begin{aligned} &\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial \epsilon} - \lambda\frac{\partial^2 u}{\partial \epsilon^2} = 0 \quad \forall(t,\epsilon) \in (0,1] \times (-1,1) \\ &u(0,\epsilon) = -\sin(\pi\epsilon) \quad \forall\epsilon \in [-1,1] \\ &u(t,-1) = u(t,1) = 0 \quad \forall t \in (0,1] \end{aligned} \tag{4.3}$$

where, $u(t,\varepsilon)$ corresponds to the fluid velocity as a function of spatial coordinate ε and time t, and λ is a parameter corresponding to the viscosity of the fluid. The goal is to approximate the solution $u(t,\varepsilon)$ to Eq. (4.3) using a neural network $u_\xi(t,\varepsilon) \approx u(t,\varepsilon)$, where ξ is used to denote the parametrization (e.g., weights and biases) of the neural network.

The parameters ξ are obtained by minimizing the loss function $\phi_\xi(X)$ comprising three components $\phi_\xi(X) = \phi_\xi^r(X^r) + \phi_\xi^0\left(X^0\right) + \phi_\xi^b\left(X^b\right)$. The mean squared residual $\phi_\xi^r(X^r)$ is given by:

$$\phi_\xi^r(X^r) = \frac{1}{N_r}\sum_{i=0}^{N_r}\left(r_\xi\left(t_i^r, \epsilon_i^r\right)\right)^2$$

where, X^r is a set of collocation points $X^r = \left\{\left(t_i^r, \epsilon_i^r\right)\right\}_{i=1}^{N_r}$, and $r_\xi(t,\varepsilon)$ denotes the residual function of the neural network approximation to the PDE. That is:

$$r_\xi(t,\epsilon) = \frac{\partial u_\xi}{\partial t} + u_\xi\frac{\partial u_\xi}{\partial \epsilon} - \lambda\frac{\partial^2 u_\xi}{\partial \epsilon^2}$$

which is evaluated over collocation points X^r. The residuals with respect to data at the initial and boundary conditions is respectively denoted by ϕ_ξ^0 and ϕ_ξ^b and are given by:

$$\phi_\xi^0\left(X^0\right) = \frac{1}{N_0}\sum_{i=1}^{N_0}\left(u_\xi\left(t_i^0, \epsilon_i^0\right) - u_0\left(\epsilon_i^0\right)\right)^2$$

$$\phi_\xi^b\left(X^b\right) = \frac{1}{N_b}\sum_{i=1}^{N_b}\left(u_\xi\left(t_i^b \epsilon_i^b\right) - u_b\left(t_i^b, \epsilon_i^b\right)\right)^2$$

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

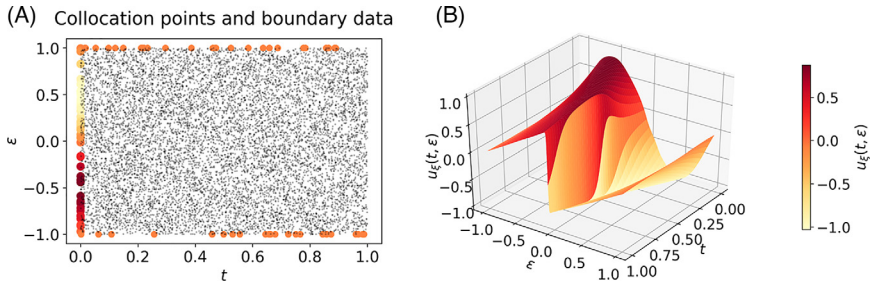


Figure 4.2 Implementation of physics-informed neural networks (PINNs) to solve Burgers equation using boundary and initial condition data. (A) Shows the data used for training corresponding to initial conditions $t = 0$, and boundary conditions $\epsilon \in \{-1, 1\}$, which were sampled from a uniform distribution. Points in *grey* indicate collocation points at which the solution was estimated. (B) Shows the solution to the Burgers equation over the temporal and spatial domains obtained using PINNs. This implementation of PINNs and Burgers equation example are based (in a modified from) on the work in [15].

where, $u_0(t,\varepsilon)$ and $u_b(t,\varepsilon)$ denote respectively initial and boundary functions given in Eq. (4.3), X_0 corresponds to data points at the initial condition (i.e., $X^0 = \left\{\left(0, \epsilon_i^0\right)\right\}_{i=1}^{N_0}\Big)$ and X_b corresponds to all data points collected at the boundary (i.e., $X^b = \left\{\left(t_i^b, \epsilon_i^b\right)\right\}_{i=1}^{N_b}$ for $x_i^b \in [-1, 1]$). Note that based on the definition of the residuals above, only initial and boundary data are used for training the neural network, with the data sets having sizes $N_0 = N_b = 50$ and $N_r = 10000$. The network architecture consisted of 8 fully connected layers with 20 units and each followed by a hyperbolic tangent activation function, and one fully connected output layer. The model was implemented in Python using the TensorFlow library [11]. The model was trained using 5000 epochs, taking 461 seconds, and achieving a loss value of $\phi_\xi(X) = 3.44 \times 10^{-4}$. The training data and resulting solution to the Burgers equation are shown in Fig. 4.2.

4.4 Regression-based approaches

4.4.1 General principle

Established linear and nonlinear regression tools are the basis of another family of approaches for discovering governing equations from data [16]. Here, the core element is a dictionary of candidate *basis functions* (i.e., nonlinear operators) given by $\left\{\theta_j(\mathrm{x}, \mathrm{u}) \mid \theta_j : \mathcal{X} \times \mathcal{U} \to \mathbb{R} \forall j \in [1, n_\theta]\right\}$. The key assumption is that the governing equations of the system can be constructed

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

as a linear combination of these basis functions. That is, following the notation in Brunton et al. [17], the dynamical system is governed by:

$$\mathcal{D}\mathbf{x} = \Xi^T \Theta(\mathbf{x}, \mathbf{u})^T \tag{4.4}$$

where, $\Theta(\mathrm{x}, \mathrm{u})^T = \left[\theta_1(\mathrm{x}, \mathrm{u}) \ldots \theta_{n_\theta}(\mathrm{x}, \mathrm{u})\right]^T \in \mathbb{R}^{n_\theta}$ is the vector form of the dictionary of basis functions, and $\Xi \in \mathbb{R}^{n_\theta} \times \mathbb{R}^{n_x}$ is a matrix of coefficients that determines the contribution of the nonlinear basis functions in the discovered governing equations for each state variable. Using the structural form defined in Eq. (4.4), which inherently assumes that the model is linear with respect to the unknown coefficients Ξ, the discovery task can be formulated as a nonlinear regression problem of the form of:

$$\begin{bmatrix} \widehat{\mathcal{D}\mathbf{x}}(\hat{t}_1)^T \\ \vdots \\ \hat{\mathcal{D}}(\hat{t}_m)^T \end{bmatrix} = \begin{bmatrix} \Theta\left(\hat{\mathbf{x}}(\hat{t}_1), \hat{\mathbf{u}}(\hat{t}_1)\right) \\ \vdots \\ \Theta\left(\hat{\mathbf{x}}(\hat{t}_m), \hat{\mathbf{u}}(\hat{t}_m)\right) \end{bmatrix} \Xi \tag{4.5}$$

where, the values of Ξ are the unknowns to be estimated and $\Theta(\hat{\mathrm{x}}(t_k), \hat{\mathrm{u}}(t_k))$ represents the basis function matrix evaluated at the values of the manipulated inputs and the state measurements at sampling times $\hat{t}_k$. While measurements of the state derivatives (denoted as $\widehat{\mathcal{D}\mathbf{x}}$) may not be directly available in most practical scenarios, their estimates (denoted as $\mathcal{D}\hat{\mathrm{x}}$) can in principle be obtained by using numerical differentiation schemes under the usual conditions that the data are sufficiently granular, the dynamics are nonstiff, and that there is a manageable amount of measurement noise. In case these conditions are not met, estimating derivatives from the data could introduce substantial error in the above regression process, creating numerical difficulties and/or leading to the wrong result in terms of discovering the governing equations. Further, because NLR approaches generally lack the flexibility of the automatic differentiation tools used in, e.g., Raissi et al. [12] (which also offer the possibility to consider various differential operators), their applicability to systems governed by partial differential equations has been studied to a lesser degree, and emphasis has been placed on the discovery of ordinary differential equations. Nonetheless, their interpretable and general additive structure, as well as the likely more modest data requirements than the previously discussed PINNs, make NLR approaches an appealing avenue for distilling governing equations from data. Furthermore, domain knowledge can be readily incorporated in the discovery task by making an informed selection of the candidate basis

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

functions and selecting which state variables to measure (similarly to the observational biases mentioned in Section 4.3).

Remark 3. The formulation Eq. (4.5) can be straightforwardly adjusted to cope with the discovery of the overall differential operator on the left-hand side of model Eq. (4.4). This could be accomplished, for example, by considering basis functions that correspond to differential operators (e.g., $\partial^2(\cdot)/\partial\boldsymbol{\epsilon}^2$), although their evaluation on noisy or sparsely sampled data can pose a significant challenge.

Nonlinear regression-based strategies can be broadly categorized as "expansive" and "contractive" depending on the way that the set of basis functions $\Theta(\cdot)$ is defined. The former case involves a typically small initial set of basis functions which is iteratively enlarged until a model of desired complexity and predictive ability is recovered (e.g., Udrescu and Tegmark [18]). Conversely, contractive strategies rely on a large initial dictionary of candidate functions, which is contracted by setting some of the elements of Ξ in Eq. (4.5) to zero (e.g., Brunton et al. [17]). We discuss these ideas in more detail below.

4.4.2 Expansive strategies

Initial data-driven efforts for discovering governing equations were based on symbolic regression strategies, which are typically expansive in nature. Symbolic regression consists of simultaneously finding *both* the form of the equations (i.e., the best combination of symbolic basis functions) *and* the values of the parameters (i.e., the coefficients associated with the selected basis functions) that best represent the data [19]. Genetic programming algorithms are employed for searching the basis functions as combinations of terms that minimize various error metrics. Typical algorithms [4] consist of evaluating and refining combinations of symbolic functions, and terminate by returning a (potentially large) set of equations that are relatively low in complexity and predict the measurement data with sufficiently good accuracy (i.e., the ones that are the most likely to reflect the underlying governing dynamics of a system). Fig. 4.3 illustrates the key steps in distilling governing equations from data via symbolic regression as proposed by Schmidt and Lipson [4].

Intuitively, expanding the search space (i.e., increasing the number of candidate basis functions) improves the probability of correctly retrieving the true governing equation set, but comes at the expense of increased computational complexity. This is particularly problematic for expansive

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

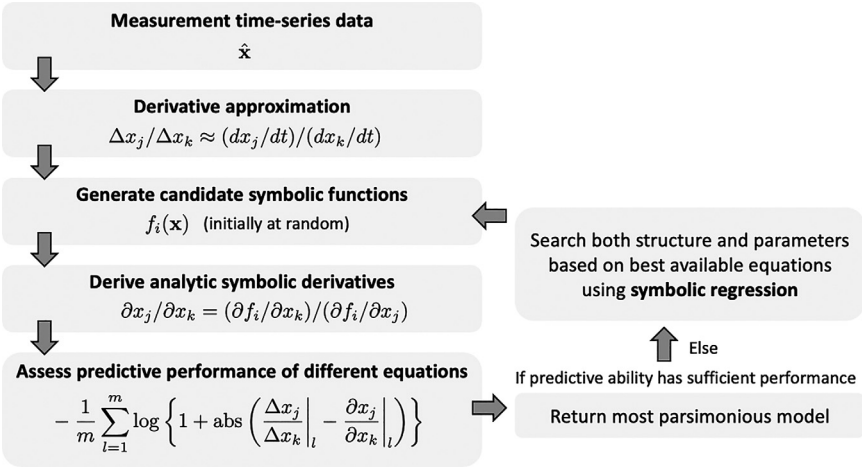


Figure 4.3 Expansive symbolic regression-based discovery framework (modified from [4]) using the mean-log error of the residual between measured data and model predictions as a metric for model performance.

regression-based schemes, since their combinatorial nature may render them computationally prohibitive even for low-dimensional systems and small-scale data sets. Furthermore, the expansive paradigm is prone to overfitting, driving the search process to overly complex expressions (where more terms continue to be included in order to reduce the prediction error for the training data) that are very different from the true governing equations. The trade-off between the predictive ability and parsimony of a proposed model can be considered from a Pareto perspective, in the form of a termination criterion that attempts to balance between these two desirable model properties [4].

More recent works in this area [18,20] combine established model fitting techniques (e.g., dimensional analysis, polynomial, and symbolic regression) with neural network-based approaches. For example, Udrescu and Tegmark [18] proposed using neural networks to predict (in the surrogate sense) the unknown function $\mathbf{f}(\cdot)$ in Eq. (4.1) depending on a set of measurable variables. The trained network is subsequently used to discover translational symmetries and (additive and multiplicative) separability in the unknown equations, providing an additional benchmark to assess if a candidate model is suitable as a governing equation. The proposed enhanced symbolic regression framework iteratively reduces the equation search space and is demonstrated to have substantial performance improvements relative to prior works (e.g., Schmidt and Lipson [4]; Dubčáková [21]) in terms of

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

both correct discovery rate and computation time. Nevertheless, training the neural network component requires extensive data sets, and the implementation of these ideas to the discovery of the governing equations of (spatially distributed) dynamical systems remains to be investigated.

4.4.3 Contractive strategies

Contractive strategies represent a potentially more efficient approach to the discovery problem and frame this task as a matter of determining the (few) correct nonzero entries of Ξ in Eq. (4.4) by developing sparse or regularized regression schemes [16]. From a regression perspective, inducing sparsity means exploiting the hypothesis that, out of a large set of potential predictors, only a small number are part of the model and have nonzero coefficients. In our context, the active basis functions (the "predictors") in the governing Eq. (4.5) are determined by the *sparse* vectors $\xi_i \in \mathbb{R}^{n_\theta}$ which correspond to the columns of matrix Ξ, and are obtained by solving the following convex unconstrained optimization problem for each state variable $i \in \{1, \ldots, n_x\}$.

$$\min_{\boldsymbol{\xi}_i} \frac{1}{2m} \sum_{j=1}^{m} \left\| \Theta\left(\hat{\mathbf{x}}(\hat{t}_j), \hat{\mathbf{u}}(\hat{t}_j)\right)\boldsymbol{\xi}_i - \dot{\hat{\mathbf{x}}}_i(\hat{t}_j) \right\|_2^2 + \lambda\rho \left\| \boldsymbol{\xi}_i \right\|_1 + \frac{\lambda(1-\rho)}{2} \left\| \boldsymbol{\xi}_i \right\|_2^2 \tag{4.6}$$

The objective function in Eq. (4.6) comprises two parts: first, the ℓ_2-norms of the difference between the value of the state derivatives predicted by the model $\Theta\left(\hat{\mathrm{x}}(t_j), \hat{\mathrm{u}}(t_j)\right)\xi_i$ and the corresponding values $\dot{\hat{\mathrm{x}}}_i(t_j)$ estimated from the m data samples at each sample time t_j, and second, a regularization penalty that is included by employing the ℓ_1 and/or ℓ_2-norms of the coefficient vectors ξ_i, thereby ensuring that the optimal solution is sparse. The parameter $\lambda \geq 0$ controls the extent of this sparsification, while $0 \leq \rho < 1$ allows for balancing between ℓ_1 and/or ℓ_2 norms. It is important to note that formulations of the type of Eq. (4.6) inherently assume that reliable estimates of the derivative $\dot{\hat{\mathrm{x}}}_i(\hat{t}_j)$ can be obtained from measurement data, an assumption that is subject to the precautions listed earlier.

The choice of regularization penalty is paramount in ensuring that the discovered models have the desired degree of complexity (as determined by the number of active basis functions), while also having good predictive ability. The least absolute shrinkage and selection operator (LASSO) [22] (i.e., $\lambda > 0$ and $\rho = 1$ in Eq. (4.6)) was employed by Brunton et al. [17], and it is known that under certain conditions LASSO can converge to the true dynamics by setting all irrelevant basis function coefficients to zero.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Nonetheless, due to the nondifferentiable nature of the LASSO objective function, specific numerical solution schemes (e.g., Kim et al. [23]) must be implemented to obtain Ξ, which can be computationally expensive for large data sets (particularly if the parameter λ needs to be determined via cross-validation). An additional challenge with LASSO regression is feature colinearity (which arises when the columns of matrix $\Theta\left(\hat{\mathbf{x}}\left(t_j\right), \hat{\mathbf{u}}\left(t_j\right)\right)$ are approximately linear dependent), which can result in spurious basis functions being included in the discovered governing equations. On the other hand, ridge regression [24] (i.e., $\lambda > 0$ and $\rho = 0$ in Eq. (4.6)), has been shown to reduce overfitting and has an explicit analytic solution similar to the one corresponding to the ordinary least squares (OLS) problem (i.e., $\lambda = 0$ in Eq. (4.6)). Despite its favorable computational properties, ridge regression is not effective at variable selection and does not result in governing models that are as sparse as those obtained via LASSO. To address the aforementioned computational issues with LASSO together with the lack of sparsity stemming from ridge regression schemes, elastic-nets [25] comprise a linear combination of both regularization penalties (i.e., $\lambda > 0$ and $0 < \rho < 1$ in Eq. (4.6)) and have been recently employed for identifying/learning the models of nonlinear dynamical systems [26]. Other nonconvex regularization penalties will be discussed in a subsequent section.

In a similar vein, thresholding techniques have been proposed [17,27] as a means to induce sparsity in the recovered governing equations. Thresholding attempts to eliminate basis functions whose corresponding parameters meet certain criteria, such as being low in magnitude. Conceptually, a thresholding-based algorithm begins with solving an ordinary least squares (OLS) regression problem where the predictors are the values of the basis functions estimated using the measured states at each time sample, and the response variable is the estimated value of the state derivative at the same time points. All basis functions whose coefficients are less than some threshold value are eliminated from the basis, and the OLS regression problem is solved again with the contracted basis. The algorithm converges once no further basis functions are eliminated from the discovered dynamics. Similar to conventional hyperparameter tuning strategies in machine learning routines, a cross-validation type approach can be used to determine the value of the threshold [17]. A key difficulty with thresholding methodologies is that, since they are based solely on the magnitude of the coefficients obtained via OLS, they may not lend themselves to the discovery of dynamical systems whose evolution features multiple time scales (or, in the simplest instance,

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

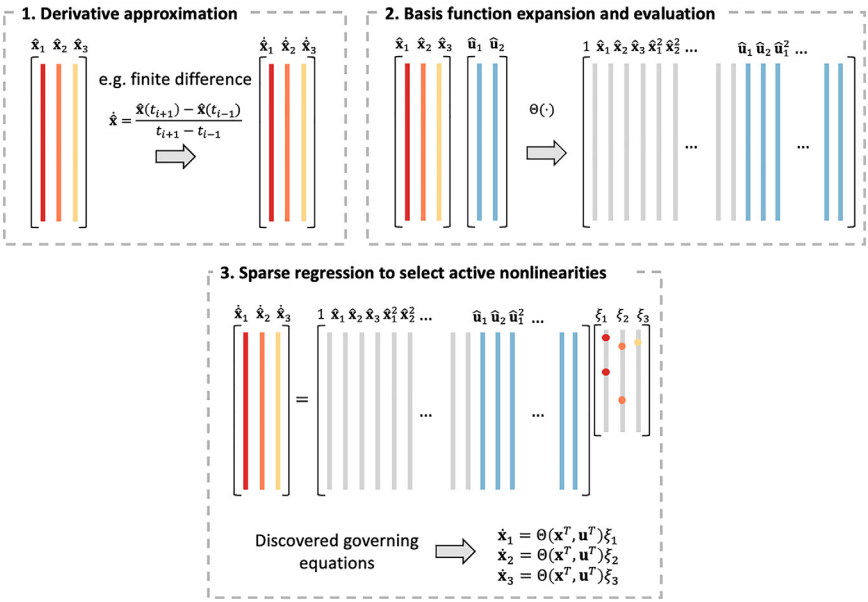


Figure 4.4 Representation of sparse regression-based discovery frameworks modified from Brunton et al. [17] and Sun and Braatz [26].

two-time scale systems whose dynamics have both fast and slow components). In this case, the coefficients Ξ in Eq. (4.4) may have significantly different magnitudes and the slow(er) component(s) of the system behavior may not be captured in the resulting model [28,29].

Contractive strategies (summarized conceptually in Fig. 4.4) have been studied extensively and an open-source implementation of the key algorithms is available [30]. Furthermore, sufficient conditions for convergence and convergence rate proofs were provided [31]. A recent related development is the work of Sun and Braatz [26], which employs a nonlinear autoregressive model with exogenous inputs (NARX) to identify models relevant to manufacturing applications. While the resulting models are interpretable and of relatively low complexity, they do not, in general, correspond to the system's governing equations.

4.5 Techniques based on mathematical programming

4.5.1 General principle

While optimization problems lie at the core of most of the aforementioned strategies, formulating the discovery problem directly and explicitly as a (constrained) mathematical program offers increased versatility and flexibility. In particular, it allows for relaxing several of the assumptions

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

inherent to the regression methods introduced in Section 4.4 including: (1) considering different objective functions (e.g., minimizing "customized" error metrics, potentially with physically relevant regularization penalties), (2) introducing constrains reflecting, e.g., domain knowledge or physically relevant relationships to improve convergence, and (3) discovering dynamics that depart from the linearity and convexity assumptions introduced in the structure of models Eqs. (4.4) and (4.6). Mathematical programming techniques are amenable to discovering governing equations of the following more general form:

$$\mathcal{D}\mathrm{x} = \Xi^T \Theta(\mathrm{x}, \mathbf{u}, \mathbf{c})^T \tag{4.7}$$

where, the vector $\mathrm{c} \in \mathbb{R}^{n_x \times n_c}$ represents an unknown potentially nonlinear parametrization of the basis functions such that the dictionary of functions defined as $\left\{\theta_j\left(\mathrm{x}, \mathrm{u}, \mathrm{c}_{i,j}\right) \mid \theta_j : \mathcal{X} \times \mathcal{U} \times \mathcal{C} \to \mathbb{R} \forall j \in [1, n_\theta]\right\}$. It is important to note that the dynamics in Eq. (4.7) are no longer linear with respect to both the unknown model coefficients Ξ and basis function parameters c, making a significant departure from the sparse regression formulations in Section 4.4.3 (which are unsuitable for discovering governing equations that include nonlinear functions of unknown parameters).

The framework introduced by Cozad et al. [32], referred to as "automated learning of algebraic models for optimization" (ALAMO), is an example of such optimization-based learning approaches. ALAMO focuses on surrogate modeling, and the aim is to generate parsimonious data-driven representations of high-dimensional models that are "expensive" to simulate and use in, e.g., optimization calculations. These representations are iteratively validated and improved via an adaptive sampling simulation scheme. The surrogate model is constructed as a linear combination of a basis of candidate nonlinear functions, in the same manner as discussed in Section 4.4. However, contrary to the contractive sparse regression strategies reviewed in Section 4.4.3, the surrogate model is identified via a mixed-integer linear programming (MILP) reformulation of the best subset problem [32]. This model generation strategy was shown to have better performance relative to ordinary least squares regression, LASSO, and exhaustive search best subset methods [33], in terms of predictive power, model parsimony and CPU time requirements. Despite the typically high accuracy and low complexity of the surrogate models obtained through ALAMO, the discovered equations can be vastly different from the true governing equations of the system [34], although it should be stressed that the discovery of the governing equations is not the explicitly stated purpose of the work by Cozad et al. [32].

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

More recently, (nonconvex) optimization-based extensions of the sparse regression discovery strategies introduced in Section 4.4.3 have been proposed. Champion et al. [35] introduced a relaxed version of the optimization problem Eq. (4.6) of the form:

$$\min_{\boldsymbol{\xi}_i, \mathbf{c}_i} \frac{1}{2m} \sum_{j=1}^{m} \Theta\left(\hat{\mathbf{x}}\left(\hat{t}_j\right), \hat{\mathbf{u}}\left(\hat{t}_j\right), \mathbf{c}_i\right)\boldsymbol{\xi}_i - \dot{\hat{\mathbf{x}}}_i\left(\hat{t}_j\right)_2^2 + \lambda \ell(\Xi) \tag{4.8}$$

where, $\ell(\Xi)$ represents a nonconvex regularization penalty such as, e.g., the ℓ_0 norm, which improves the sparsity of the identified models relative to the aforementioned LASSO and elastic net formulations. Furthermore, via a slight modification of the formulation in Eq. (4.8), Champion et al. [35] provided additional functionality to perform data trimming and cope with high levels of data corruption and outliers, as well as to handle physical constrains of the form of $\mathbf{C}\boldsymbol{\xi}_i = \mathbf{d}_i i \in \{1, \ldots, n_x\}$. Along these lines, and related to the approach proposed by Wilson and Sahinidis [34], an explicit upper bound on the ℓ_0-norm regularization function (imposed as an inequality constraint) can enable direct control on the number of basis functions selected in the discovered governing equations.

Goyal and Benner [27] blended sparse identification methods with Runge-Kutta integration schemes. The proposed approach does not require access to or approximation of temporal gradient data, thus avoiding approximation errors and improving convergence to the governing dynamics when data are sparsely measured and corrupted with noise. The general form of the l_1-regularized optimization problem is given by [27]:

$$\min_{\boldsymbol{\xi}_i, \mathbf{c}_i} \frac{1}{2m} \sum_{j=1}^{m-1} \left\| \mathbf{g}_i\left(\Theta\left(\mathbf{x}\left(\hat{t}_j\right), \mathbf{u}\left(\hat{t}_j\right), \mathbf{c}_i\right), \boldsymbol{\xi}_i\right) - \hat{\mathbf{x}}_i\left(\hat{t}_{j+1}\right) \right\|_2^2 + \lambda \parallel \Xi \parallel_1 \tag{4.9}$$

where, the operator $\mathrm{g}_i(\cdot) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_\theta} \to \mathbb{R}$ denotes the numerical integration scheme used to represent the continuous time dynamics in discrete-time form. That is, for the explicit Runge-Kutta 4th order scheme used by Goyal and Benner [27] to discover dynamics of the form of $\dot{\mathrm{x}}_i = \mathrm{f}_i(\mathrm{x}(t))$, the compact notation $\mathrm{x}_i(t_{j+1}) = \mathrm{g}_i(\mathrm{x}(t_j))$ reflects:

$$\begin{aligned}
&\mathbf{x}_i\left(t_{j+1}\right) \approx \mathrm{x}_i\left(t_j\right) + \frac{1}{6} h_j\left(\mathbf{k}_{i,1} + \mathbf{k}_{i,2} + \mathbf{k}_{i,3} + \mathbf{k}_{i,4}\right) \\
&h_j = t_{j+1} - t_j \\
&\mathbf{k}_{i,1} = \mathbf{f}_i\left(\mathrm{x}\left(t_j\right)\right), \mathbf{k}_{i,2} = \mathbf{f}_i\left(\mathrm{x}\left(t_j + h_j \frac{\mathbf{k}_{i,1}}{2}\right)\right) \\
&\mathbf{k}_{i,3} = \mathbf{f}_i\left(\mathrm{x}\left(t_j + h_j \frac{\mathbf{k}_{i,2}}{2}\right)\right), \mathbf{k}_{i,4} = \mathbf{f}_i\left(\mathrm{x}\left(t_j + h_j \mathbf{k}_{i,3}\right)\right)
\end{aligned} \tag{4.10}$$

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

The resulting optimization problem is nonlinear and nonconvex, but the framework can in principle handle the discovery of governing equations involving, e.g., rational functions, external control inputs, and basis functions with nonlinear parametrization.

4.5.2 Moving horizon formulations

It should be noted that finding globally optimal solutions (or even good locally optimal solutions) to the constrained nonlinear, nonconvex optimization problems of the type Eqs. (4.8) and (4.9) is challenging. Intuitively, the computational complexity of the optimization problem Eq. (4.9) increases with: (1) the size of the data set available (m) and/or the granularity of the discretization grid or mesh, (2) the dimension of the system (in terms of the number of states n_x and manipulated inputs n_u), and (3) the number of basis functions considered initially in the dictionary (n_θ). Evidently, the circumstances that lead to greater computational complexity are also expected to improve the odds of correct discovery (e.g., larger amounts of training data can help reduce model bias, more granular discretizations reduce differentiation errors, and a larger dictionary increases the probability that the governing equations is contained within the basis). Furthermore, because of the nonconvex nature of the problems, convergence to different local optima is sensitive to the initial values of the coefficients Ξ and parameters c in optimization problems Eqs. (4.8) and (4.9), and the development of systematic methods for identifying these initial values has not been completely resolved in the literature.

Rooted in moving horizon state estimation and control theory, the discovery of dynamical systems via moving horizon optimization [8] leverages a time horizon decomposition strategy to alleviate the computational burden of such large-scale nonconvex problems. A key element is partitioning the data set $\left\{\left(\hat{\mathrm{x}}\left(\hat{t}_k\right), \hat{\mathrm{u}}\left(\hat{t}_k\right)\right)\forall k \in \{1, \ldots, m\}\right\}$ into (typically) sequential subsets or folds $\left\{\left(\hat{\mathrm{x}}\left(\hat{t}_k\right), \hat{\mathrm{u}}\left(\hat{t}_k\right)\right)\forall k \in \left\{j, j+1, \ldots, j+H\right\}\right\}$ of smaller size, i.e., defined for a horizon spanning $H \leq m$ measurements. The algorithm is initialized with $j = 0$ and at each iteration, after solving optimization problem in Eq. (4.11), the "oldest" $\Delta\mathcal{D}$ measurements are discarded and a new data subset is obtained by shifting the horizon forward in time (i.e., $\leftarrow j + \Delta\mathcal{D}$), where $\Delta\mathcal{D}$ is the data step size. The difference between H and $\Delta\mathcal{D}$ is illustrated in Fig. 4.5, which provides a graphical representation of the moving horizon algorithm based on the framework introduced in Lejarza and Baldea [8,36]. It should be noted that the above operations need not be performed online, and the way in which the subsets are selected can be altered (e.g., they can be chosen at random from the available data).

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

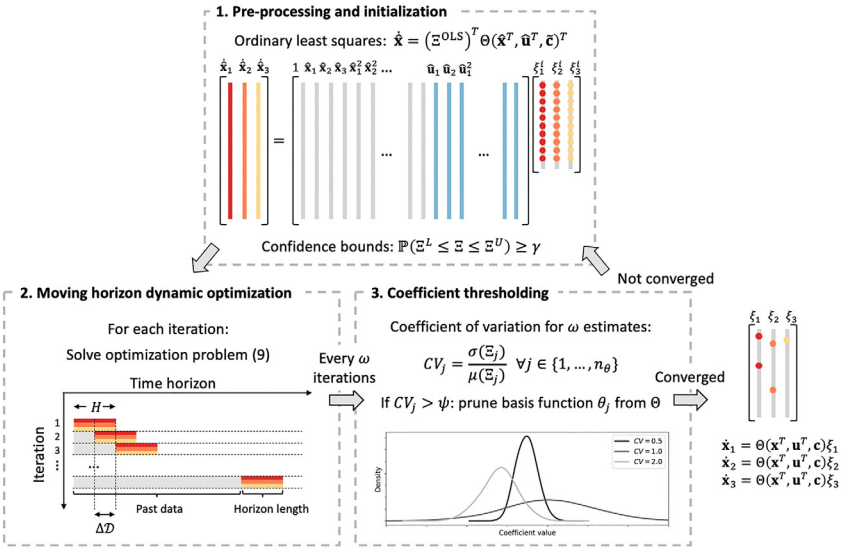


Figure 4.5 Illustration of the core elements of the algorithm in moving horizon discovery, adapted from Lejarza and Baldea [8], where $\tilde{c}$ denotes the initial value of parameter c, γ is the confidence level used to derive Ξ^L and Ξ^U, and ψ is the variability threshold used to prune basis functions.

Since the model Eq. (4.7) is continuous (in time and spatial coordinates), discretization techniques such as the Runge-Kutta method discussed above [27], must be employed to derive a tractable optimization problem. In this case, following the framework proposed by Lejarza and Baldea [8] we employ an orthogonal collocation on finite elements scheme [37] to represent the candidate governing equations discretely. Here, the governing equations are discovered by repeatedly/sequentially solving a nonlinear program (NLP) of the form:

$$\begin{aligned} \min_{\Xi, \mathbf{c}, \mathbf{x}} \quad & \frac{1}{2NK} \sum_{i=0}^{N} \sum_{j=0}^{K} \mathrm{x}\left\| \left(t_{ij}\right) - \tilde{\mathbf{x}}\left(t_{ij}\right) \right\|_2^2 + \lambda \ell(\Xi) \\ \text{s.t.} \quad & \mathrm{g}\left(\Theta\left(\mathrm{x}\left(t_{ij}\right)^T, \tilde{\mathbf{u}}\left(t_{ij}\right)^T, \mathbf{c}\right), \Xi\right) = 0 \\ & \Xi \in \left[\Xi^L, \Xi^U\right], \quad \mathbf{c} \in \left[\mathbf{c}^L, \mathbf{c}^U\right] \end{aligned} \tag{4.11}$$

where, $\mathrm{x}(t_{ij})$ are the states predicted at t_{ij} in finite element i and collocation point j, the algebraic constrains $\mathrm{g}(\cdot) = 0$ denote the discretized version of the nonlinear dynamics given in Eq. (4.7), Ξ^L,c^L and Ξ^U,c^U are respectively the lower and upper bounds of the estimated coefficients and parameters

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

(derived using domain knowledge or statistical methods as will be discussed subsequently), and K is the number of collocation points on each of the N finite elements. The collocation equations form a set of algebraic constraints given by:

$$\begin{aligned}
\left.\frac{d\mathrm{x}(t)}{dt}\right|_{t_{ij}} &= \frac{1}{h_i}\sum_{k=0}^{K}\mathrm{x}_{ik}\frac{d\mathcal{L}_k(\tau_j)}{d\tau}, \quad j\in\{1,\ldots,K\}, i\in\{1,\ldots,N\}\\
\left.\frac{d\mathrm{x}(t)}{dt}\right|_{t_{ij}} &= \Xi^T\Theta\left(\mathrm{x}(t_{ij})^T, \mathbf{u}(t_{ij})^T, \mathbf{c}\right)^T, \quad j\in\{1,\ldots,K\}, i\in\{1,\ldots,N\}\\
\mathrm{x}(t_{i+1,0}) &= \sum_{k=0}^{K}\mathcal{L}_k(1)\mathrm{x}(t_{ik}), \quad i\in\{1,\ldots,N-1\}
\end{aligned} \tag{4.12}$$

where, h_i is the length of finite element i, $t_{i-1} = t_{ij} - \tau_j h_i$, and the state variable $\mathrm{x}(t)$ is interpolated using Lagrange polynomials $\mathcal{L}_k$ as follows:

$$\begin{aligned}
\mathbf{x}(t) &= \sum_{k=0}^{K}\mathcal{L}_k(\tau)\mathbf{x}_{ik}, \quad t\in[t_{i-1}, t_i], \tau\in[0,1]\\
\mathcal{L}_k(\tau) &= \prod_{j=0,\neq k}^{K}\frac{\tau-\tau_j}{\tau_k-\tau_j}
\end{aligned} \tag{4.13}$$

The optimal choice of the location of interpolation points (given by τ_j) is derived in detail by Biegler [37] (Theorem 10.1). In compact form, we denote Eqs. (4.12) and (4.13) by $\mathrm{g}\left(\Theta\left(\mathrm{x}(t_{ij})^T, \tilde{\mathrm{u}}(t_{ij})^T, \mathrm{c}\right), \Xi\right) = 0$. The use of orthogonal collocation methods allows for dealing with stiff nonlinear dynamical systems that feature dynamics that evolve over multiple time scales. We note that leveraging this type of advanced numerical discretization techniques further motivates the moving horizon decomposition, as they come at the expense of larger problem dimensions. The reader is referred to the paper by Lejarza and Baldea [8] for a more in depth discussion on the algorithmic aspect of moving horizon discovery and for a series of illustrative examples. We note that using a discretization of the dynamics may require the use of interpolation and/or resampling techniques to estimate the values of the state and input variables at the relevant discretization points $t_k \forall k \in \{1, \ldots, M\}$ (unless the sampling time is used as the differentiation time step (i.e., $\hat{t}_{k+1} - \hat{t}_k$) as in Goyal and Benner [27], which can potentially introduce significant approximation error).

In addition to the computational benefits associated with the fact that instances of Eq. (4.11) are solved with data subsets of smaller size, obtaining *a*

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

sequence of estimates of coefficients Ξ and parameters c (rather than a single estimate as is the case with all the techniques reviewed thus far) allows for pruning the basis function dictionary (i.e., eliminating basis functions from the model) in a systematic manner. Lejarza and Baldea [8] introduced a pruning strategy based on the observed variability of the estimated coefficients (as opposed to simply using the magnitude of the estimates as discussed in Section 4.4.3). This idea is rooted in the assertion that the estimated values of nonbasic coefficients (i.e., corresponding to basis functions that do not belong in the governing equations of the underlying system, which are referred as "nonbasic") are likely to be not only low in magnitude, but also vary significantly each time they are estimated from successive folds of data. The variability of these nonbasic coefficients is linked to the likely case that nonbasic functions contribute to "overfitting," i.e., serve to reproduce measurement noise rather than the signal in the data. Thus, the variability of the nonbasic coefficients will be driven by the variance of the noise component of the data, which will be reflected in each iteration as new portions of the original data set are considered. On the other hand, estimates for basic coefficients (i.e., corresponding to basis functions that do appear in the true governing equations) are likely to be more consistent (and have lower variance) when computed on different subsets of the original data. These coefficients contribute to predicting the underlying signal, and are expected to be less sensitive to noise corrupting the training data.

4.6 Demonstration of moving horizon discovery for a batch chemical process

We consider a liquid-phase, irreversible reaction $A \rightarrow B$ carried out in a *nonisothermal batch* reactor system, which is depicted in Fig. 4.6. The governing equations for this process are obtained from the material and energy balances, and the true model is assumed to be given by the following set of nonlinear ordinary differential equations:

$$\begin{aligned} \frac{dC_A(t)}{dt} &= -k_0 e^{-E_a/RT(t)} C_A(t) \\ \frac{dT(t)}{dt} &= \frac{(-\Delta H_R)}{\rho C} k_0 e^{-E_a/RT(t)} C_A(t) + \frac{U\mathcal{A}}{V\rho C}(T_c - T(t)) \end{aligned} \tag{4.14}$$

where, the state variables are the concentration of component A in molar units denoted as $C_A(t)$, and the liquid temperature T in degrees Kelvin. The

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

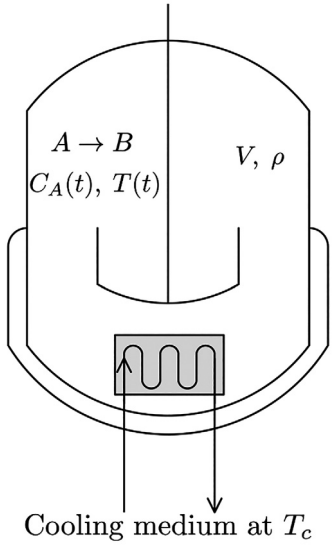


Figure 4.6 Schematic of the nonisothermal batch reactor system whose material and energy balances are discovered from data as a case study.

Table 4.1 Nonisothermal batch reactor parameter values used for numerical experiments.

Parameter	Value	Units
V	100	L
ρ	1000	g L^{-1}
C	0.239	J g^{-1} K^{-1}
$-\Delta H_R$	5×10^4	J mol^{-1}
E_a/R	8750	K
k_0	7.2×10^10	$min^{-}1$
$U\mathcal{A}$	5×10^4	J min^{-1} K^{-1}
T_c	300	K

system is assumed to be well mixed and thus there are no spatial variations in concentration or temperature. In Eq. (4.14), ρ is the density of the liquid, C is the heat capacity, ΔH_R is the heat of reaction per mole of A reacted, T_c is the temperature of the coolant, U is the overall heat transfer coefficient, $\mathcal{A}$ is the heat transfer area, k_0 is the pre-exponential factor, E_a is the activation energy, and R is the universal gas constant. A list of all model parameters and their values in the true model is provided in Table 4.1. Evidently, the governing Eq. (4.14) is nonlinear with respect to some of the unknown

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

parameters (specifically, the exponential term the Arrhenius expression of the reaction rate in Eq. (4.14) is a nonlinear function of the parameter E_a/R), and thus their correct structure cannot be obtained when employing the sparse regression schemes presented in Section 4.4. Rather, the discovery effort is amenable to the more general representation of the governing equations in Eq. (4.7), and to the methods introduced in Section 4.5.

A continuous process similar to the one shown in Fig. 4.6 was studied by Lejarza and Baldea [36]. In the isothermal instance of the continuous system, the dynamics of the process could be excited by manipulating the inlet flow rate; in the nonisothermal instance, the system was shown to achieve a periodic steady-state [36]. The resulting data sets in the continuous case are particularly well suited for implementing the moving horizon discovery concept as each data subset is likely to reflect a broader dynamic behavior and contain different observed values (not at steady-state) of the state variables. The discontinuous nature of batch manufacturing, on the other hand, renders the collection of data that are suited for the discovery mission more difficult for a number of reasons: (1) there may be fewer manipulated inputs available to excite the dynamics to generate sufficiently informative data sets as mentioned previously, and (2) batch-to-batch variations may require batch data synchronization procedures (e.g., using time warping techniques [38]).

While the dynamics of continuous processes might be more straightforward to excite (e.g., by manipulating inlet conditions), thus facilitating the collection of informative data sets), in the case of batch processess, data corresponding to multiple batches run at different conditions (e.g., initial species concentration and initial mixture temperature) generally need to be collected. In practical settings, extensive experimentation might be a costly and time-consuming task, and the physical process configurations may not be different enough to excite and reveal all the dynamic components present in the true governing equations. A possible solution to these data collection obstacles is to build a synthetic bootstrapped data set out of only a few batch experiments. These experiments could consist of varying, e.g., the initial conditions of each batch, as well as other process parameters like batch volume and coolant temperature (within feasible ranges). It should be noted that when these variables are changed across experiments, the basis functions in the governing equations should explicitly account for these different configurations (e.g., the term $T(t)/V$ should be explicitly included in the dictionary of basis functions). Otherwise, the data sets generated with different reactor configurations may lead the process of learning the governing equations to inconsistent results.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

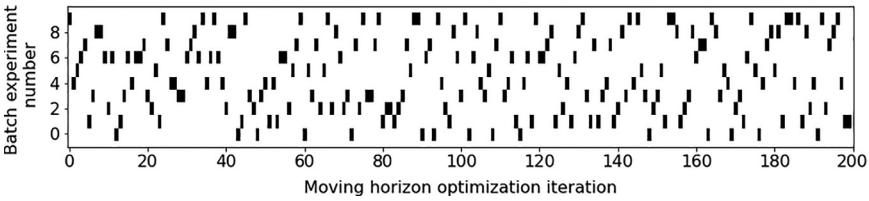


Figure 4.7 Bootstrapped data sequence generated from ten individual batch experiments each having different initial conditions and batch process times. In this case, the experiments consisted of integrating the governing equations and subsequently corrupting the obtained trajectories with Gaussian noise.

Once these data are available, bootstrapping with replacement can be used to construct a large data sequence of batch experiments. While the framework allows for considering multiple batch experiments simultaneously (which could be a potential avenue for improving convergence), this would involve defining the discrete dynamics $\mathbf{g}(\cdot)$ for each data set and thus increasing the problem dimensions. Fig. 4.7 shows a bootstrapped sequence generated from ten batch experiments, which is used with the moving horizon discovery algorithm [8] to discover the governing equations. Each iteration of the proposed scheme consists of solving the problem in Eq. (4.11) using the data from one of the randomly selected batch experiments from the available set as depicted in Fig. 4.7, which allows for leveraging the aforementioned variance arguments to prune basis functions from the dictionary. Another advantage of this approach is that each batch experiment is considered independently without the need for applying time warping techniques, which in practice could also allow for learning from data gathered from, e.g., interrupted and off-spec batches that will likely have shorter durations than a successfully completed batch.

The moving horizon optimization algorithm proposed by Lejarza and Baldea [8] (whose open-source implementation is publicly available [8]) was applied for discovering the governing equations of both $C_A(t)$ and $T(t)$ using the following dictionary of nonlinear basis functions:

$$\Theta(C_A, T, c) = \left\{1, C_A, T, C_A T, C_A^2, T^2, C_A^2 T, C_A T^2, C_A^{-1}, T^{-1}, C_A^{-1} T^{-1}, e^{c/T}\right\} \tag{4.15}$$

where, parameter c represents $-E_a/R$ in Eq. (4.14). The same preprocessing steps were followed as by Lejarza and Baldea [36]. In brief, the data were first smoothed using the Savitzky-Golay filter with quadratic polynomials and window size determined iteratively as described by Lejarza and Baldea [8]. The Pearson correlation coefficient between the derivative of the state

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Table 4.2 Discovery results for the nonisothermal batch reactor for a range of simulated measurement noise levels (σ), corresponding to the governing equations for composition ($C_A(t)$) and temperature ($T(t)$). Results shown include the average mean squared error (MSE) of the final 50 iterations with respective standard deviation in parentheses, as well as complexity as indicated by the number of basis functions present in the resulting discovered equation.

Variable	Measurement noise (σ)	Mean squared error	Complexity
	$\sigma_C A = 0.1\% C_A(0)$, $\sigma_T = 0.1\% T(0)$	1.96×10^{-7} (7.00×10^{-8})	1
Composition (C_A)	$\sigma_C A = 0.5\% C_A(0)$, $\sigma_T = 0.5\% T(0)$	2.15×10^{-6} (1.11×10^{-6})	1
	$\sigma_C A = 1.0\% C_A(0)$, $\sigma_T = 1.0\% T(0)$	7.32×10^{-6} (2.97×10^{-6})	1
Temperature (T)	$\sigma_C A = 0.1\% C_A(0)$, $\sigma_T = 0.1\% T(0)$	2.24×10^{-2} (7.68×10^{-3})	3
	$\sigma_C A = 0.5\% C_A(0)$, $\sigma_T = 0.5\% T(0)$	1.89×10^{-1} (4.40×10^{-2})	3
	$\sigma_C A = 1.0\% C_A(0)$, $\sigma_T = 1.0\% T(0)$	5.64×10^{-1} (1.17×10^{-1})	3

variables (approximated via central finite differences) and $e^{c/\hat{T}}$ for a fine grid of possible values of c, was used to find a good initial estimate of parameter c. In this case, the exact value of c was recovered for the governing equation of $C_A(t)$ since this is the single basis function in the composition dynamics Eq. (4.14) (i.e., the true value of parameter c exhibits the highest correlation with respect to the estimated derivative $\dot{\hat{C}}_A$). Next, Granger causality and ordinary least squares regression preprocessing steps were performed as a preliminary pruning procedure and to derive confidence intervals $[\Xi^L, \Xi^U]$ to improve the convergence of the moving horizon optimization scheme. Basis functions having coefficients whose coefficient of variation (CV), defined as the standard deviation divided by the mean of the estimated values for a certain number of batch experiments, was greater than a threshold ψ, were eliminated from the basis. A thresholding tolerance of $\psi = 1$ was used to eliminate basis functions, for which the coefficient of variation was computed using ten successive estimates of the model coefficients. This thresholding procedure is illustrated graphically in Fig. 4.5, and more details can be found in the paper of Lejarza and Baldea [8].

Results for the discovery of the batch process governing equations under increasing levels of simulated measurement noise are shown in Table 4.2. The

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

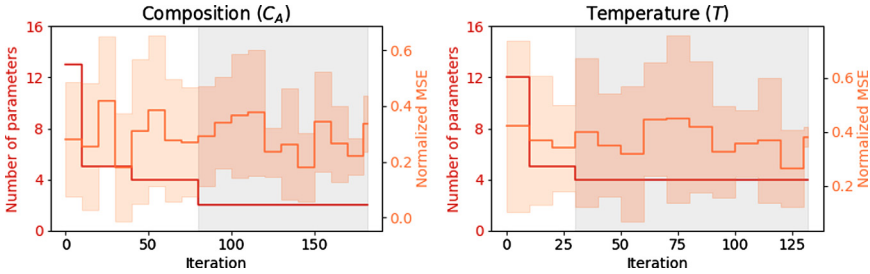


Figure 4.8 Number of parameters (including Ξ and c) and average mean squared error (MSE) as a function of moving horizon iterations (averages are computed every ten iterations corresponding to the thresholding frequency used in DySMHO). The shaded area surrounding MSE lines indicates one standard deviation away from the mean ($mu \pm \sigma$). The shaded grey area indicates the convergence of the algorithm to the true governing equations. Results displayed correspond to the instance $\sigma = (1\% C_A(0), 1\% T(0))$.

variance of the measurement noise is determined as a percentage of the initial conditions ($C_A(t = 0)$ and $T(t = 0)$) used to generate the batch data, and the performance of the moving horizon discovery technique is quantified in terms of the mean squared error (MSE) between the trajectories predicted form the discovered model and that of the data, as well as with respect to model complexity (i.e., the number of basis functions present in the model). The correct structure of the governing equations was discovered for all noise instances and for both state variables, but we observe that increasing noise levels result in less accurate parameter estimates and thus larger MSE values.

The results displayed in Fig. 4.8 represent the evolution of the discovered model's complexity and predictive ability as a function of the number of iterations of the discovery algorithm. Intuitively, model complexity is high initially and decreases significantly as basis functions are pruned from the dictionary. Conversely, the predictive ability does not change dramatically as the algorithm proceeds, even when the number of features (i.e., basis functions) substantially decreases. Initially, when there is a larger number of candidate basis functions, the discovered equations are expected to have very low error and likely overfit the training data. The fact that the average MSE remains virtually constant (but with a noticeably lower variance) when the nonbasic basis functions in the dictionary have been pruned is an indicator of a successful learning process and suggests that the algorithm has identified the most parsimonious model within the dictionary of basis functions.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

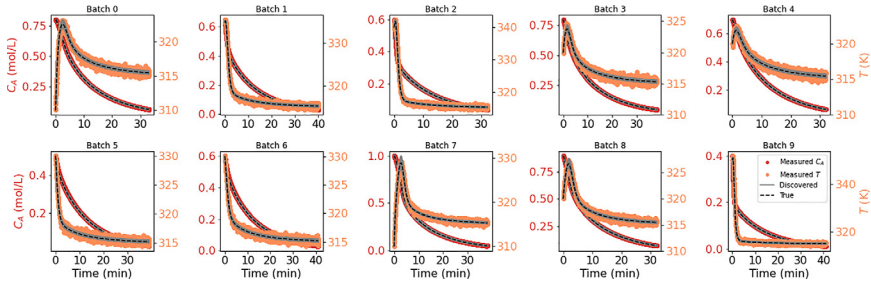


Figure 4.9 Simulations of the discovered equations (evolution of state variables over time) in comparison to the true governing equations and the measured data for each of the 10 batch experiments $\sigma = (1\% C_A(0), 1\% T(0))$. Plots show 500 trajectories for the discovered dynamics generated with random parameters sampled from a multivariate Gaussian distribution.

When implementing fundamental learning schemes to discover dynamical systems whose governing equations are unknown, it is paramount to have proper validation methods in place to assess the recovered models. The predictive ability and complexity analysis discussed previously and shown in Fig. 4.8 is one such approach to gauge algorithm performance, and to evaluate its final output. Another commonly employed validation strategy involves simulating the recovered equations (e.g., solving the discovered differential equation for a set of initial and boundary conditions) and comparing the resulting trajectories to the measured data. These results are shown in Fig. 4.9 for the ten data sets (batches) used for training. The parameter estimates where tested for normality and then maximum likelihood estimation was used to estimate the covariance of the coefficients obtained. The estimated mean and covariance were used to generate 500 independent and identically distributed model parameter samples, which were in turn used to generate multiple plausible discovered trajectories (grey lines) for each of the ten batch experiments. Clearly, the discovered model accurately captures the dynamics observed for all batches despite the noticeable amount of noise in the data. An alternative visualization to the one in Fig. 4.9 is the phase plane representation (a coordinate plane with the axes being the values of the state variables) as shown in Fig. 4.10. This representation can help to validate whether the discovered governing equations correctly capture the general dynamic behavior of the observed system, and can be particularly useful in the case where the system behavior is periodic. Table 4.3 shows a comparison of the discovered equations relative to the true model used to generate the data.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

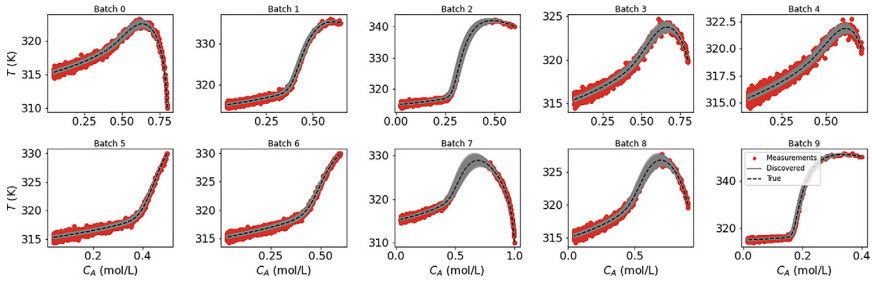


Figure 4.10 Simulations of the discovered equations in comparison to the true governing equations and the measured data for each of the ten batch experiments $\sigma = (1\%C_A(0), 1\%T(0))$. Phase plane representation of the results in Fig. 4.9.

Table 4.3 Summary of discovered equations and associated parameters relative to the true first-principles model for concentration and temperature. The results for the discovered equations correspond to the highest noise instance in Table 4.2.

Dynamics	Governing equations	
dC_A/dt	*True*	$-7.20 \times 10^{10} \exp(-8750/T(t))C_A(t)$
	Discovered	$-7.23 \times 10^{10} \exp(-8752/T(t))C_A(t)$
dT/dt	*True*	$669 - 2.09T(t) + 1.51 \times 10^{13} \exp(-8750/T(\mathrm{t}))C_A(t)$
	Discovered	$657 - 2.08T(t) + 1.59 \times 10^{13} \exp(-8768/T(t))C_A(t)$

4.7 Conclusion

Data-driven methods are paramount in developing decision-support models at all stages of the life of a manufacturing facility. A key role is played by governing equations, that provide significant advantages (e.g., interpretability and extrapolation capabilities) relative to black-box data-driven models. This chapter highlights some of the most recent advances in using machine learning methods to derive such governing equations. Available works are broadly categorized into physics-informed machine learning, nonlinear regression (NLR) (including symbolic regression), and mathematical programming-based techniques, including moving horizon discovery. An illustrative example pertaining to a nonisothermal batch reactor system is presented to highlight some of the main properties of moving horizon discovery. Future work should be directed to continue to improve on the scalability of governing equations discovery schemes to cope with larger volumes of data and higher dimensional systems. Similarly, the application to more complex industrially relevant processes constitutes an important next step in implementing machine learning for manufacturing

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

and in discovering governing equations of newer systems and technologies. Further integration of the aforementioned schemes with state estimation tools could lead to improved capabilities for discovering the governing equations of systems with, e.g., unmeasured states and large amounts of measurement noise.

References

[1] M. Soroush, M. Baldea, T.F. Edgar, Smart Manufacturing: Concepts and Methods, Elsevier Science, 2020.
[2] M. Soroush, M. Baldea, T.F. Edgar, Smart Manufacturing: Applications and Case Studies, Elsevier Science, 2020.
[3] S. Forrest, Genetic algorithms: principles of natural selection applied to computation, Science 261 (1993) 872–878.
[4] M. Schmidt, H. Lipson, Distilling free-form natural laws from experimental data, Science 324 (2009) 81–85.
[5] P.I. Frazier, A tutorial on Bayesian optimization, in: arXiv preprint arXiv:1807.02811 (2018).
[6] R.J. Hyndman, G. Athanasopoulos, Forecasting: principles and practice, OTexts (2018).
[7] I.B. Tjoa, L.T. Biegler, Simultaneous solution and optimization strategies for parameter estimation of differential-algebraic equation systems, Ind. Eng. Chem. Res. 30 (1991) 376–385.
[8] F. Lejarza, M. Baldea, Data-driven discovery of the governing equations of dynamical systems via moving horizon optimization, Sci. Rep. 12 (2022) 1–15. https://github.com/Baldea-Group/DySMHO.
[9] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (2021) 422–440.
[10] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: an imperative style, high-performance deep learning library, Adv. Neural. Inf. Process Syst. 32 (2019) 8026–8037.
[11] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. https://www.tensorflow.org/.
[12] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[13] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, M.W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, Adv. Neural. Inf. Process Syst. 34 (2021) 26548–26560.
[14] S. Wang, X. Yu, P. Perdikaris, When and why pinns fail to train: a neural tangent kernel perspective, J. Comput. Phys. 449 (2022) 110768.
[15] J. Blechschmidt, O.G. Ernst, Three ways to solve partial differential equations with neural networksa review, GAMM-Mitteilungen 44 (2021) e202100006.
[16] G. James, D. Witten, T. Hastie, R. Tibshirani, An Introduction to Statistical Learning, 112, Springer, 2013.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[17] S.L. Brunton, J.L. Proctor, J.N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. 113 (2016) 3932–3937.
[18] S.-M. Udrescu, M. Tegmark, AI Feynman: a physics-inspired method for symbolic regression, Sci. Adv. 6 (2020) eaay2631.
[19] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, 1, MIT Press, 1992.
[20] M. Cranmer, A. Sanchez-Gonzalez, P. Battaglia, R. Xu, K. Cranmer, D. Spergel, S. Ho, Discovering symbolic models from deep learning with inductive biases, in: arXiv preprint arXiv:2006.11287 (2020).
[21] R. Dubčáková, Eureqa: software review, Genet Program Evolv. Mach. 12 (2011) 173–178.
[22] R. Tibshirani, Regression shrinkage and selection via the lasso, J. R. Stat. Soc. Series B Stat. Methodol. 58 (1996) 267–288.
[23] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, D. Gorinevsky, An interior-point method for largescale l_1-regularized least squares, IEEE J. Sel. Top Signal Process 1 (2007) 606–617.
[24] A.E. Hoerl, R.W. Kennard, Ridge regression: biased estimation for nonorthogonal problems, Technometrics 12 (1970) 55–67.
[25] H. Zou, T. Hastie, Regularization and variable selection via the elastic net, J. R. Stat. Soc. Series B Stat. Methodol. 67 (2005) 301–320.
[26] W. Sun, R.D. Braatz, ALVEN: Algebraic learning via elastic net for static and dynamic nonlinear model identification, Comput. Chem. Eng. 143 (2020) 107103.
[27] P. Goyal, P. Benner, Discovery of nonlinear dynamical systems using a runge-kutta inspired dictionary-based sparse regression approach, in: arXiv preprint arXiv:2105.04869 (2021).
[28] M. Baldea, P. Daoutidis, Dynamics and Nonlinear Control of Integrated Process Systems, Cambridge University Press, 2012.
[29] M. Baldea, P. Daoutidis, Control of integrated process networksa multi-time scale perspective, Comput. Chem. Eng. 31 (2007) 426–444.
[30] B. de Silva, K. Champion, M. Quade, J.-C. Loiseau, J. Kutz, S. Brunton, PySINDy: A python package for the sparse identification of nonlinear dynamical systems from data. J. Open Source Softw. 5 (49) (2020) 2104.
[31] L. Zhang, H. Schaeffer, On the convergence of the SINDy algorithm, Multiscale Model. Simul. 17 (2019) 948–972.
[32] A. Cozad, N.V. Sahinidis, D.C. Miller, Learning surrogate models for simulation-based optimization, AlChE J. 60 (2014) 2211–2227.
[33] K.P. Burnham, D.R. Anderson, Practical use of the information-theoretic approach, in: Model Selection and Inference, Springer, 1998, pp. 75–117.
[34] Z.T. Wilson, N.V. Sahinidis, The ALAMO approach to machine learning, Comput. Chem. Eng. 106 (2017) 785–795.
[35] K. Champion, P. Zheng, A.Y. Aravkin, S.L. Brunton, J.N. Kutz, A unified sparse optimization framework to learn parsimonious physics-informed models from data, IEEE Access 8 (2020) 169259–169271.
[36] F. Lejarza, M. Baldea, Discovering governing equations via moving horizon learning: the case of reacting systems (Submitted), AlChE J. (2021).
[37] L.T. Biegler, Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes, SIAM, 2010.
[38] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, IEEE Trans. Acoust. Speech Signal Process. 26 (1978) 43–49.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY

Copyright Elsevier 2024

CHAPTER FIVE

Data-driven optimization algorithms

Burcu Beykal[a,b] and Efstratios N. Pistikopoulos[c,d]
[a]Department of Chemical and Biomolecular Engineering, University of Connecticut, Storrs, CT, United States
[b]Center for Clean Energy Engineering, University of Connecticut, Storrs, CT, United States
[c]Texas A&M Energy Institute, Texas A&M University, College Station, TX, United States
[d]Artie McFerrin Department of Chemical Engineering, Texas A&M University, College Station, TX, United States

5.1 Introduction

With the fourth industrial revolution (a.k.a Industry 4.0), a spotlight is put on the traditional manufacturing processes. Transitioning from disconnected industrial practices to more interconnected "smart" digital systems that can communicate with one another through data is starting to become the norm in manufacturing. An essential component of this "smart" practice is through the creation of digital twins [1], which allows you to collect vast amounts of data within seconds, test operating conditions, improve existing processes, and design new ones in a safe and cost-effective manner. With the increasing computation power, the digital twins of any process can now be modeled in greater detail by considering complex physical, chemical, and mechanical phenomena across multiple temporal and spatial scales. While the abundance of data collected through the automation systems and the digital twin creates an untapped potential for deriving insight from the manufacturing system, the combination of this information with first-principles expressions (typically presented in the form of ordinary or partial differential equations) in a multi-scale modeling framework creates a unique challenge for finding the best operational conditions or new process routes through mathematical optimization. This challenge stems from the sheer complexity of the representative models, noise, and lack of explicit analytical forms in the collected data, which require excessive computational time and effort to find a viable solution using traditional optimization techniques.

Data-driven optimization emerged as a computationally efficient alternative to traditional approaches that require explicit analytical forms (i.e., dataless or white-box computation) and the first-order optimality

Artificial Intelligence in Manufacturing: Concepts and Methods.
DOI: https://doi.org/10.1016/B978-0-323-99134-6.00006-2
Copyright © 2024 Elsevier Inc.
All rights are reserved, including those for text and data mining, AI training, and similar technologies.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY
Copyright Elsevier 2024

conditions of the original problem to retrieve the solution. Data-driven optimization problems can be of different varieties and have the following general mathematical form:

$$\begin{aligned} \min_{x,y} \quad & f \\ s.t.\ & g_m \leq 0 \qquad \forall m \in \{1, \ldots, M\} \\ & g_k(\boldsymbol{x}, \boldsymbol{y}) \leq 0 \qquad \forall k \in \{1, \ldots, K\} \\ & \boldsymbol{x} \in \left[\boldsymbol{x}^L, \boldsymbol{x}^U\right] \\ & \boldsymbol{y} \in \ integer \\ & \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{y} \in \mathbb{Z}^b \end{aligned} \tag{5.1}$$

The goal of Eq. (5.1) is to find the optimal set of continuous decision variables $\boldsymbol{x}$ and integer variables $\boldsymbol{y}$ which minimize an objective function f with an unknown closed-form and subject to constraints with known closed-form, $g_k(\boldsymbol{x},\boldsymbol{y})$, defined by the set $k \in \{1, \ldots, K\}$, and constraints with unknown form (i.e., black-box constraints or output constraints), g_m, defined by the set $m \in \{1, \ldots, M\}$. In the absence of any integer variables, these problems are continuous in nature and can be: (1) unconstrained without g_k, g_m, and the bounds on the continuous decision variables; (2) box-constrained with $\boldsymbol{x} \in [\boldsymbol{x}^L, \boldsymbol{x}^U]$; or (3) constrained problems with explicit g_k and/or black-box g_m inequality constraints. The bulk of the data-driven optimization algorithms falls into the category of unconstrained and box-constrained continuous problems where recent efforts have also focused on handling constraints and mixed-integer decision variables within a data-driven optimization framework as later introduced in the chapter.

Data-driven algorithms are also categorized with respect to their solution strategy. Direct search algorithms seek the direction with the best objective function value, whereas model-based algorithms fit approximate surrogate models and use the surrogate's derivative information to guide its search toward promising regions. Several review articles and textbooks have also classified data-driven or derivative-free optimization algorithms with respect to their optimization capabilities such as local versus global optimization and deterministic versus stochastic approaches [2–4]. We will also be following these broadly accepted categorizations and provide an extensive literature review spanning from algorithmic developments to applications, as well as the most recent advances in data-driven optimization algorithms including extensions to mixed-integer formulations and various classes of optimization problems. For rich theory on data-driven optimization, the readers are guided to the excellent textbooks by Conn et al. [5] and Audet and

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Hare [6] and to comprehensive historic analysis on data-driven optimization provided by the authors of references [2–4,7–16].

5.2 Algorithmic approaches for data-driven optimization

5.2.1 Direct search algorithms

5.2.1.1 Local optimization techniques

One of the earliest and most popular algorithms of data-driven optimization is the Nelder–Mead algorithm which uses a *simplex-based search* methodology to find the minimum of an unconstrained optimization problem [17]. Based on the original work of Spendley et al. [18], the main idea of the Nelder–Mead algorithm is to perform operations on a simplex (i.e., the convex hull of an affine independent set of points) in such a way that it moves away from the point with the worst objective function value. Essentially, the next algorithmic action is decided purely on the value of the worst incumbent solution and not on the first-order optimality condition of the original problem. The algorithm is designed to be adaptive in such a way that the optimal point is approximated by replacing the worst vertex of the simplex through reflecting, expanding, contracting, or shrinking iterations (Fig. 5.1). Hence, the algorithm adapts the shape of the simplex in iterative procedures to capture the local contours of the objective function [19].

Because of the simplicity of the algorithm, Nelder–Mead is widely used and analyzed by the scientific community. The convergence properties of this algorithm are investigated by Lagarias et al. [20] and McKinnon [19]. A sensitivity analysis on the algorithm parameters is performed by Wang and Shoup [21]. A modified algorithm is proposed by Tseng [22], which enables a flexible and dynamic choice for trial simplices. Further modifications to the original deterministic algorithm are also proposed to handle noisy stochastic functions [23–27], combinatorial [28], and mixed-integer [29] formulations. The algorithm is also generalized to be executed over multiple processors for computational speed-up [30,31]. The classical Nelder–Mead is further extended to global optimization using probabilistic restart initialization [32,33] and constrained optimization using an adaptive linear penalty function [33].

One of the pitfalls of this popular algorithm is the loss of efficiency and convergence in high-dimensional problems. The complexity of a single Nelder–Mead iteration [34], the effect of the initialization procedure [35],

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

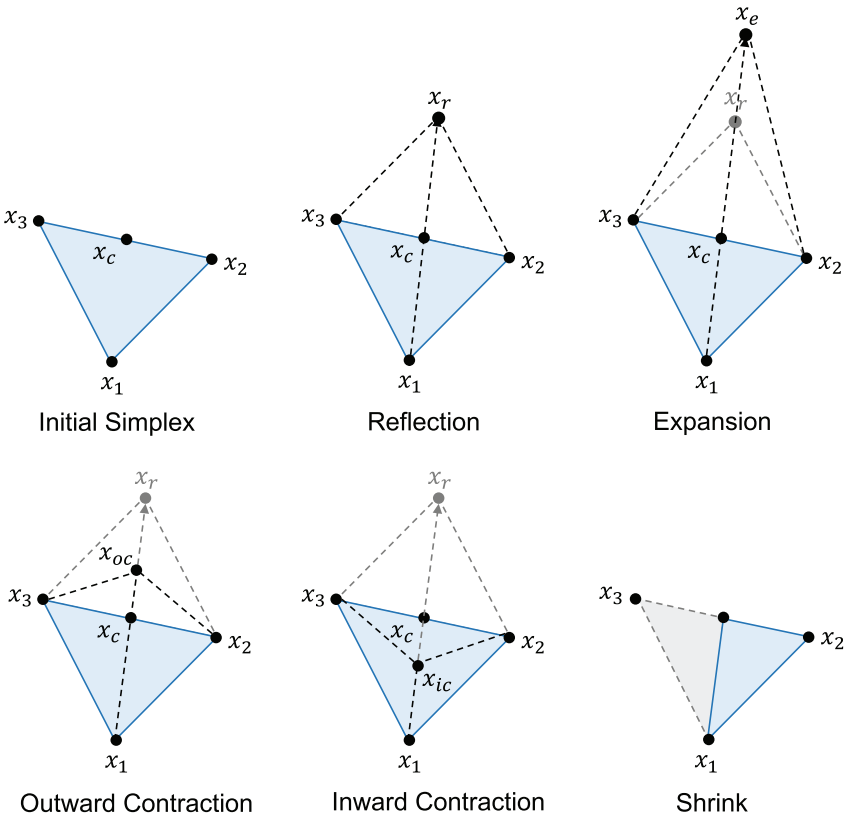


Figure 5.1 Reflection, expansion, contraction, and shrink operations in the Nelder–Mead algorithm shown on a two-dimensional simplex.

and the effect of problem dimensionality on the overall algorithm are investigated [36]. Gao and Han [37] provided insights on why the algorithm is inefficient in high-dimensional cases through an investigation of its iterative operations. Their results showed that the expansion and contraction steps provide a descent property when the objective function is uniformly convex but the efficiency of these steps is diminished with the increasing problem dimensionality. Also, their findings indicated that for this class of problems, the original algorithm never uses the shrink operation and performs a large number of reflection operations, which amplifies the inefficiency of the algorithm [37]. Furthermore, Fajfar et al. [38] proposed a perturbed centroid Nelder–Mead approach to overcome the convergence issues in high-dimensional problems. Other convergent variants of Nelder–Mead are presented by the authors of references [22,39–45].

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

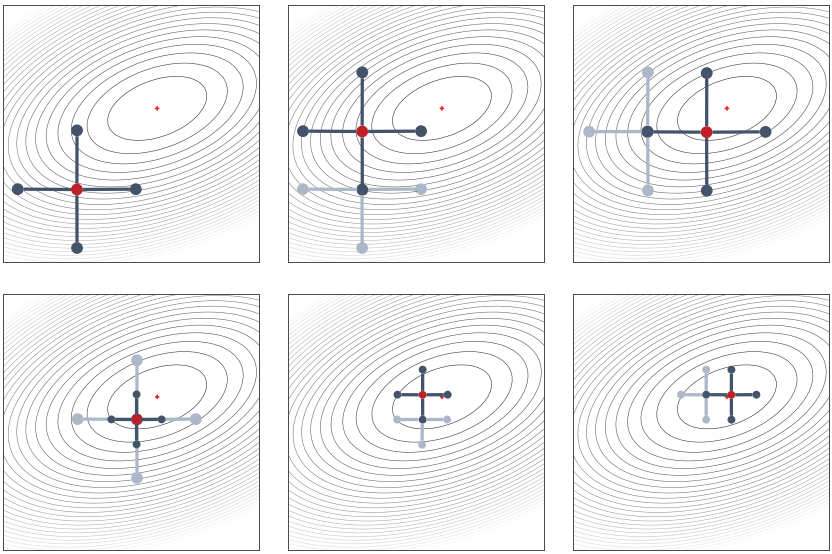


Figure 5.2 A simple demonstration of pattern search.

Another widely used technique in direct search algorithms is *pattern search*, which uses a pattern or a finite set of directions to find the minimum of an objective function (Fig. 5.2). One of the earliest pattern search algorithms is developed by Hooke and Jeeves [46], which performs sequences of exploratory moves to understand the local behavior of the objective function (i.e., discovering descending/ascending valleys) and pattern moves to improve on the incumbent solution by progressing along the explored valleys [47]. Multi-directional search is later proposed by Dennis and Torczon [48] to perform unconstrained optimization on parallel machines, which is also noted to be applicable to the Hooke and Jeeves' [46] pattern search algorithm. These earlier developments led to the generalization of the pattern search methods by Torzcon [49] and coined the term "Generalized Pattern Search (GPS)" for unconstrained optimization. The goal of this algorithm is to find a better objective function value by identifying a set of candidate points called the *mesh*. The mesh points are identified around the incumbent solution by multiplying the pattern vector with a scalar called the *mesh size* and adding it to the incumbent solution with the best objective function value. If this exploratory move is successful, where a point with a better objective function value is identified, the mesh is retained, if not, the mesh is refined and a new set of neighboring points is explored [50].

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

The GPS algorithm is extended to handle positive bases [51], nonsmooth optimization problems [52,53], bound-constrained [54], linearly constrained [55], and general constrained problems [56,57]. Audet and Dennis [58] also introduced distinct *search* and *poll* steps for GPS algorithms. The search step is the equivalent of the exploratory move of the initial GPS algorithm [49] where this is further relaxed to use heuristics or other techniques to identify mesh points. Like in the original algorithm, if the search step is successful and identifies a better solution, the mesh is retained and the search continues. However, if the search step fails to find a point with a better objective function value, the algorithm evaluates the points in the mesh and calculates their objective function value for polling. The polling step searches within the vicinity of the incumbent solution to improve the current solution. If the poll step finds a point with the better objective function value, the step is deemed successful and the incumbent solution is updated. The mesh size is also expanded and the current pattern is used for further exploration. If the polling fails, the mesh is said to be a local optimizer and refined to be smaller. The polling can be *complete*, where all points in the mesh are evaluated and the best of all points is selected to be the incumbent solution or it can be *opportunistic*, where the polling will stop when a better solution than the incumbent solution is identified. Detailed descriptions of search and poll steps are provided by Audet and Dennis [58], and Audet [59] with further modifications to benefit from the derivative information to improve the polling performance are provided by Abramson et al. [60].

Soon after the emergence of the initial GPS algorithm, Kolda et al. [9] introduced the "Generating Set Search (GSS)" algorithm which is a further generalization of the GPS and encapsulates not only the GPS algorithm [49] but also captures methods based on grids [61,62] and sufficient decrease conditions [63,64]. The GSS algorithm is also extended to piecewise smooth [65], linearly constrained [66–69], and general constrained problems [70]. The early algorithmic improvements further led to the development of the mesh adaptive direct search (MADS) algorithm which extended the GPS by polling in an asymptotically dense set of directions with the extreme barrier approach to handle general constraints [71]. This algorithm is also extended to handle constraints with the progressive barrier approach [72], to choose orthogonal polling directions deterministically [73], and implemented within the NOMAD software [74]. Also, quadratic [75] and treed Gaussian process surrogates [76] are explored within the MADS for constrained optimization. More recently, Audet et al. [77] proposed a new method to update the mesh to handle granular and discrete variables, whereas

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Seo et al. [78] used principal component analysis to derive the direction vector to have better exploration properties with the MADS algorithm.

5.2.1.2 Global optimization techniques

Pattern search algorithms are also used for data-driven global optimization. Global and Local Optimization using Direct Search (GLODS) is a constrained global pattern search algorithm that uses a multi-start directional search approach to explore all local minima of an objective function to identify the global solution among these solutions [79]. The algorithm starts by exploring promising subdomains using a search step where later these regions are exploited by a polling step. The extreme barrier approach is adopted to handle constraints. This algorithm is later extended to address constrained multi-objective multi-modal optimization problems [80].

Another popular technique for global optimization using search algorithms is Particle Swarm Optimization (PSO). PSO is one of the most famous population-based algorithms developed by Kennedy and Eberhart [81] to address the global optimization of continuous nonlinear functions. It was initially proposed to simulate the social behavior of species like bird flocks for cooperatively finding food. The main idea is that the members of a swarm of species will keep changing the search pattern based on their and other members' experiences, and ultimately find the optimum. Within an optimization context, this translates into having an initial number of particles in the decision space that are evaluated at their current location. Then, each particle moves through this space in a stochastic manner to the best-fitness locations that are identified by combining historical information from the particle's experiences and of the swarm, until the swarm finds its way closer to the optimal solution. A number of books [82–85], reviews [86–89], and research articles are dedicated to PSO, as well as algorithmic extensions are proposed to increase the efficiency of the original algorithm. Various fields of engineering and sciences also commonly use PSO for optimization. For example, Sorek et al. [90] used PSO as the main optimizer for waterflooding production optimization with parameterized control structures. Bi et al. [91] used a hybrid genetic and PSO algorithm to find the global solution of a steam cracking process. The algorithm is also extended to address mixed-integer [92–94], multi-objective [95–97], bi-level [98,99], and dynamic optimization [100–102] problems. Only a representative subset of applications and algorithmic developments are mentioned here, but it is important to note that a mass number of open literature is available on all aspects of the PSO algorithm.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

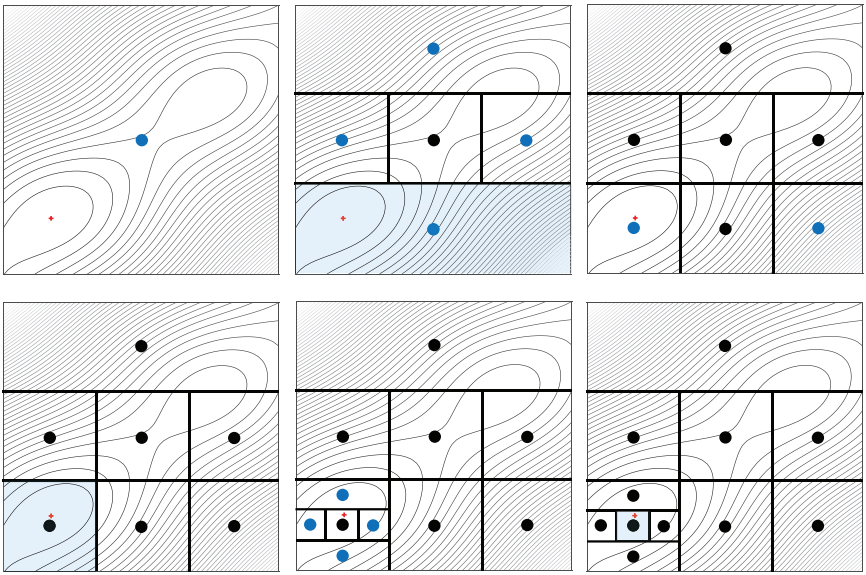


Figure 5.3 Iterations of the DIRECT algorithm shown on the minimization of the McCormick function.

Furthermore, subspace partitioning algorithms also play a key role in search-based optimization. DIviding RECTangles (DIRECT) is a global optimization algorithm for finding the minimum of a multivariate nonlinear function subject to bound constraints [103,104]. It is a modification of the algorithm introduced by Shubert [105], where the DIRECT approach uses the Lipschitz constant, which is the bound on the rate of change of the objective function, as a weight on the global exploration versus local exploitation. This allows the algorithm to have faster convergence and operating capabilities at the local and global levels simultaneously. The DIRECT algorithm works by partitioning the hypercube of decision variables into subrectangles. The center point of each subrectangle is a sampling point for the optimization where the objective function is evaluated. At every iteration, promising subrectangles are identified and explored for further search by diving them into even smaller rectangles. The iterations continue until the upper bound on the number of iterations or function evaluations is hit. A demonstration of the space partitioning is provided in Fig. 5.3.

As the sampling in DIRECT is carefully designed to be executed at the midpoint of each hyper-rectangle, as opposed to the vertices of the rectangle, it is noted that the algorithm faces slow convergence when the global solution lies on the boundary [104,106]. To improve the convergence

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

properties, the DIRECT algorithm is extended to have a biased search near the local optima [107], improved identification of potentially optimal rectangles [108], integer variable and constraint handling [109–113], and enhanced partitioning schemes [114–119]. This also inspired the development of the Multilevel Coordinate Search algorithm [106] for the global optimization of bound-constrained problems which allows irregular partitioning and placing sampling points anywhere in the hypercube, rather than just the center point as in the DIRECT algorithm.

5.2.1.3 Hybrid techniques

Many works of direct search algorithms have also focused on hybrid implementations of local and/or global techniques for improved performance. Lai and Achenbach [120] introduced the Discrete Variable Direct Search algorithm which combines ideas from the simplex-based constrained optimization algorithm of Box [121] and the pattern search algorithm of Hooke and Jeeves [46] to perform constrained nonlinear optimization. Genetic algorithms, Tabu search, simulated annealing, PSO, and their variants are commonly hybridized with the Nelder–Mead to address the global optimization of unconstrained and box-constrained nonlinear programs with good exploration and exploitation properties [122–130]. Such hybrid algorithms are also developed to address manufacturing problems. For example, a Nelder–Mead–PSO hybrid approach is used to solve economic and economic–statistical designs of control charts that are used for monitoring the quality of manufactured goods [131].

Hybrid techniques are also extended to handle constrained optimization problems with an embedded constraint operator, a penalty function, or a solution selection criterion [132–134]. Various alternatives of the GPS and GSS algorithms are also proposed in the open literature where simplex gradients are implemented in these algorithms to improve the convergence rate while sustaining the robustness of the original algorithm [135,136]. Pattern search algorithms are hybridized with PSO, variable neighborhood search, and their variants, similar to the Nelder–Mead, to have good global exploration and local exploitation properties, including in the PSwarm algorithm [137] and others [138–140].

5.2.2 Model-based algorithms

5.2.2.1 Local optimization techniques

In this section, we focus our attention on trust–region algorithms for model-based local data-driven optimization. Trust–region methods rely on a local

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

model-based approximation of the original function to perform nonlinear optimization. These methods work by creating a linear or quadratic approximation around the incumbent solution, which is then easily solved to identify the next iterate for building another approximate model centered around this new point. The models are only trusted within the neighborhood of the current solution (i.e., "trust–region") which is adaptively adjusted at every iteration based on how well the approximate model predicts the decrease in the objective function value. If the improvement is acceptable based on the evaluation of the ratio of the actual reduction to the predicted reduction in the objective function value, the trust–region is enlarged, if not, the region radius is contracted and the iterations will proceed until a convergence criterion is met. For algorithmic details, the readers are guided to the comprehensive textbook by Conn et al. [141], which provides an extensive discussion on trust–region methods and the review papers by Yuan [142,143].

Powell's methods are one of the most popular and widely used trust–region-based optimization techniques [144–147]. COBYLA [144] is one of the earliest trust–region-based constrained optimization strategies, which approximates the objectives and the constraints using linear multivariate interpolation at the vertices of a simplex. This is followed by the development of the unconstrained optimization by quadratic approximation (UOBYQA) algorithm for unconstrained optimization using quadratic interpolation [145] and the development of the NEWUOA algorithm [146] to address problems with expensive evaluations. In the UOBYQA algorithm $(n + 1)(n + 2)/2$, a number of interpolation points are required to build a unique quadratic approximation of an n-dimensional objective function. The NEWUOA algorithm builds quadratic approximations with a fewer fixed number of points where new points are added to the interpolation set by deleting another point. Later NEWUOA algorithm is extended to handle bound constraints in the new algorithm called BOBYQA [147].

Many algorithmic advancements are also proposed on Powell's methods. ORBIT is developed [148] to perform unconstrained local optimization using radial basis functions with a fully linear tail in a trust-region framework. The radial basis interpolating function serves as the surrogate model of the objective function which is adaptively updated over iterations by optimizing it over compact regions, rather than using a fixed number of interpolation points like in the NEWUOA algorithm. ORBIT is later extended as the constrained optimization by radial basis function interpolation in trust regions (CONORBIT) algorithm [149] which can handle inequality

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

constraints by modeling them using the radial basis function. Further, the NEWUOA algorithm is extended to handle linear inequality constraints [150] and to use the infinity norm rather than the Euclidean norm combined with an active set strategy for the trust-region subproblems [151]. BOBYQA algorithm is also extended to handle general constraints by using an Augmented Lagrangian approach in the trust-region subproblem optimization [152] and to mixed-integer nonlinear optimization with bound constraints using heuristic and deterministic approaches [153].

Several other notable trust–region algorithms are developed to address black-box optimization problems including: (1) the Derivative-Free Optimization (DFO) algorithm that addresses unconstrained problems using quadratic interpolation within a trust–region framework and with suggested extensions to general, "difficult," and "virtual" constrained problems by Conn et al. [154]; (2) the filter-based trust–region approach for unconstrained optimization introduced by Gould et al. [155]; (3) the hybrid trust–region and progressive barrier approach for inequality constrained optimization by Audet et al. [156]; (4) an inexact restoration filter approach used in a linear constrained trust–region algorithm for addressing equality constrained problems by Echebest et al. [157]; (5) a two-phase approach that does not require a feasible initial point to perform constrained optimization within a trust–region framework by Bajaj et al. [158]; (6) the trust-funnel approach by Sampaio and Toint [159] that makes use of polynomial interpolation models for the equality constrained nonlinear optimization while considering the objective and constraints in separate models and trust–regions; (7) a trust–region filter algorithm by Eason and Biegler [160,161] for general constrained glass-box (i.e., hybrid equation-oriented and computationally expensive black-box system) optimization; (8) the NOWPAC algorithm by Augustin and Marzouk [162] that uses linear functions for the objective and constraints in a trust–region framework, and an inner boundary path to adaptively handle nonlinear constraints; and (9) a general trust–region algorithm for optimizing smooth objective functions constrained in a closed convex domain with global convergence results presented by Conejo et al. [163].

Beyond the trust–region approach, it is also important to note that artificial neural networks (ANNs) are used as surrogate models within local optimization frameworks, especially in superstructure optimization problems to replace rigorous unit operations, thermodynamic models, or reaction mechanisms [164–167]. The appropriate number of samples required for training ANN surrogate models for optimization is studied by

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Nuchitprasittichai and Cremaschi [168], which also led to the development of adaptive sampling approaches for generating ANN surrogate models [169]. The neural networks and other surrogate modeling approaches are also investigated within data-driven global optimization frameworks which are discussed in detail in the following section.

5.2.2.2 Global optimization techniques

The global optimization of model-based approaches heavily relies on building a *global surrogate model* or a *global response surface* using the data collected from the high-fidelity black-box simulation. One of the key contributions in this subdomain is the development of the efficient global optimization (EGO) approach by Jones et al. [170] for nonlinear box-constrained optimization. EGO fits a Design and Analysis of Computer Experiments stochastic process model (i.e., a DACE model or a kriging model) to an initial space-filling design by using the maximum likelihood estimation. New sampling points are added by maximizing the expected improvement in the objective function using a branch-and-bound routine, and the optimization procedure continues iteratively through updating the sampling set and refitting the kriging surrogate model. The original algorithm is extended to solve multi-objective problems [171], constrained optimization problems [172,173], as well as stochastic and noisy black-box systems [174]. The effect of various infill sampling criteria on the performance of EGO is also investigated along with a penalty method to handle constrained problems [175]. One disadvantage of the algorithm is that only a single point can be identified per iteration in EGO which in return requires the algorithm to cycle over many iterations to find the optimal solution. Viana et al. [176] proposed a multi-surrogate approach to improve the efficiency of the EGO algorithm by adding multiple samples per iteration. Similarly, a parallel infill sample collection strategy is proposed by Hamza and Shalaby [177]. Further, Klejinen et al. [178] proposed an improvement for the EGO algorithm by modifying the kriging predictor variance estimator with a bootstrapped estimator.

The emergence of the EGO algorithm has also popularized the use of kriging models in black-box optimization. Caballero and Grossmann [179] used kriging to perform constrained black-box optimization on noisy systems. Kriging model is also hybridized with a local response surface methodology to identify the global optimum in nonlinear constrained optimization with noisy variables [180]. Boukouvala and Ierapetritou [181] developed a kriging-based noisy black-box optimization strategy with a feasibility mapping approach. Carpio et al. [182] modified the Probability of

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Improvement method to have an improved modeling and optimization framework for black-box systems using kriging models where constraints are handled using a probabilistic approach. Most recently, Paulson and Lu [183] developed the COBALT algorithm which uses kriging models (i.e., Gaussian Process models) and a generalization of the upper trust bound method using chance constraints for solving constrained black-box optimization problems. Also, Schweidtmann et al. [184] addressed the global optimization of kriging surrogate models using a reduced-space formulation.

Shortly after the development of EGO, radial basis functions are adapted within a global optimization framework by Gutmann [185] for box-constrained nonlinear optimization which used a bumpiness function to balance global exploration versus local exploitation. Radial basis functions are used as interpolants with the functional form provided in Eq. (5.2),

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} \lambda_i \phi(\|\boldsymbol{x} - \boldsymbol{x}_i\|) + p(\boldsymbol{x}) \tag{5.2}$$

where $\varphi(\bullet)$ is the basis type with choices of linear, cubic, thin plate spline, multiquadratic or Gaussian forms; $p(\boldsymbol{x})$ is the polynomial tail of the interpolant; and λ_i are the unknown parameters of the basis term. All the unknown parameters of the radial basis function, including λ_i and the ones contained in the polynomial tail, are obtained by solving a system of linear equations where the uniqueness is shown by Powell [186].

The use of radial basis functions in data-driven optimization extended thereafter. Bjorkman and Holmström [187] used a radial basis function with a linear tail for the box-constrained optimization of expensive black-box functions and demonstrated an implementation in MATLAB. Regis and Shoemaker [188] introduced the CORS algorithm for constrained optimization using radial basis functions. This method is also extended to handle noisy multi-objective formulations with box constraints [189]. The sampling method of the CORS–RBF framework is also modified to have better space-filling designs and to overcome the exponentially increasing number of sampling points in higher dimensional problems [190]. An improved version of Gutmann's approach [185] with a different restart strategy and global versus local exploration capabilities is also proposed [190]. Regis and Shoemaker [191] used dynamic coordinate search with radial basis functions to address the optimization of high-dimensional black-box systems. Most recently, Garud et al. [192] used a two-staged approach based on Delaunay Triangulation and radial basis functions to identify promising subregions and optimizing in the best one to recover the optimal solution.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Although these are significant developments for the model-based data-driven optimization theory, the aforementioned studies also show that one single type of surrogate cannot address all problems, that is, *one size does not fit all*. Hence, many recent works in the field focused on creating a selection of surrogate models or basis functions to select the one that outperforms other available options. Palmer and Realff [193,194] explored kriging and polynomial surrogate models for chemical process design optimization using simulated annealing. Won and Ray [195] studied kriging, radial basis, and co-kriging surrogate models for design optimization within a nondominated sorting genetic algorithm framework that can handle constraints through a ranking approach. Goel et al. [196] looked into an ensemble of surrogate models where they investigated combining multiple surrogates using a weighted approach for achieving a robust approximation. They also showed that the choice of the surrogate model is affected by the sampling design. Müller and Piché [197] investigated the combination of different surrogate models using the Dempster–Shafer theory. Cozad et al. [198] developed the Automated Learning of Algebraic Models (ALAMO) framework to address constrained nonlinear black-box optimization problems using a library of basis functions for identifying the optimal surrogate form that is highly accurate and tractable for optimization. Smolyak grids are used to train polynomial interpolants with Chebyshev basis functions for black-box optimization of box-constrained problems [199]. Schweidtmann and Mitsos [200] studied the global optimization of ANN surrogate models in the reduced space. Nascimento et al. [201] used ANNs to replace computationally expensive high-fidelity models as well as to model industrial data where explicit mathematical forms are lacking. In their study, the optimization of the resulting neural network is executed using grid search. The use of neural network surrogate models is also extended to multi-parametric programming to develop optimal receding horizon policies [202,203]. The ARGONAUT algorithm is also developed to address the global optimization of constrained gray-box optimization problems using sampling, surrogate modeling, and global optimization at several stages of the framework [204–206].

ARGONAUT embodies the idea of *one size does not fit all* and contains a surrogate function library composed of interpolating (kriging and radial basis functions) and noninterpolating (linear, full quadratic, and signomial) forms. The best surrogate model for each unknown equation (the objective function and constraints) is individually selected by examining the cross-validation error. The algorithm also addresses the global optimization of the parameter estimation problem and optimal constrained sampling with

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

known information. The use of ARGONAUT in data-driven optimization literature is not limited to constrained nonlinear optimization problems. It is extended to handle implicit constraints using Support Vector Machines (SVMs) [207,208], tuning the unknown parameters of dynamic surrogate models [209], and solving multi-objective optimization problems [210]. In addition, the original algorithm is incorporated within a mixed-integer nonlinear bi-level optimization framework called the DOMINO [211,212] to solve large-scale bi-level optimization problems to guaranteed feasibility. The algorithmic details of ARGONAUT and DOMINO are further demonstrated in Sections 5.4.1 and 5.4.2 on motivating multi-objective and mixed-integer nonlinear bi-level optimization problems, respectively.

Handling mixed-integer formulations. Several different model-based approaches are developed to handle mixed-integer formulations within a data-driven optimization framework. Davis and Ierapetritou [213,214] used a kriging-based approach to solve mixed-integer nonlinear formulations subject to noise. The integer complexity is handled via a branch-and-bound approach where the global relaxation of the nonlinear subproblem is obtained through kriging models, essentially decoupling the integer variables from the surrogate modeling problem. An adaptive radial basis function (ARBF) approach is proposed by Holmström et al. [215] that addresses mixed-integer global optimization of black-box models with expensive objective functions and constraints. This is an extension of Gutmann's approach [185] where target values for the optimal solution are identified across a range of values by globally optimizing a merit function several times at every iteration. In their formulation, they assume that the unknown objective function is continuous regardless of the fact that some decision variables might be an integer and add a penalty term to the objective function to handle expensive constraints. Costa and Nannicini [216] introduced the RBFOpt algorithm to solve box-constrained mixed-integer black-box problems. Similar to Holmström et al.'s [215] ARBF approach, they also assume that the objective function is continuous even when some decision variables only take integer values.

Rashid et al. [217] used adaptive multiquadratic radial basis functions for constrained mixed-integer nonlinear optimization with a multi-start approach. SO-MI algorithm [218] is developed for the global optimization of mixed-integer nonlinear optimization of black-box systems that may contain binary and/or nonbinary integer variables using radial basis functions. Sampling with integer variables is achieved by creating an initial symmetric

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Latin hypercube design in the continuous space which is then rounded to the nearest integer value to satisfy integrality in the samples. An extension of this algorithm to purely integer variables is also proposed, called the SO-I algorithm [219]. Different than the SO-MI, SO-I does not require a user-defined feasible point in the initial design and the algorithm goes through an optimization phase to identify a feasible point automatically. A further extension of the SO-MI algorithm is again proposed by Muller [220] called MISO, where various sampling strategies are used to sample points and the new algorithm includes a local search step to identify highly accurate solutions. Recently, Regis [221] introduced the CONDOR algorithm to address the constrained optimization of black-box problems with ordinal discrete variables and many black-box inequality constraints. The algorithm uses radial basis surrogate models to approximate the objective and constraints where new sampling points are generated by perturbing variables in the discrete vicinity of the best solution.

Although most algorithms benefited from the flexibility of radial basis functions, Kim and Boukouvala [222] used one-hot encoding to explicitly handle integer variables without relaxing the integrality constraints and developed mixed-variable Gaussian process and neural network models. They also studied the effect of sampling on the solution performance and provided insights on the most effective sampling strategy for solving mixed-integer black-box problems. Moreover, Thebelt et al. [223] developed the ENTMOOT framework for using tree-based models with uncertainty estimates in black-box optimization. The framework uses mixed-integer programming to recast tree-based models for optimization and can directly handle mixed variable spaces without any simplifying assumptions on the continuity/discontinuity of the objective function. Also, Ceccon et al. [224] developed the OMLT framework to translate pretrained mixed-integer surrogate models that are given in the form of neural networks or gradient-boosted trees into optimization formulations in Pyomo. Finally, Bliek et al. [225] presented the Mixed-Variable ReLU-based Surrogate Modelling (MVRSM) algorithm, where they used a linear combination of rectified linear units (ReLU) for developing surrogate models in mixed-integer spaces while satisfying integrality constraints at the locally optimal solution without imposing any numerical rounding. Keen readers are directed to a recent review article by Ploskas and Sahinidis [226] for computational performance analysis of various data-driven mixed-integer optimization software, whereas a brief summary of the open-source software applications discussed in this chapter is provided in Table 5.1.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Table 5.1 Summary of software applications with open source links.

Software	Approach	Continuous	Mixed-integer	General constraints	Sources
ALAMO	Global	√	–	√	[227]
ARBF	Global	√	√	√	[228]
BOBYQA	Local	√	–	–	[229,230]
COBALT	Global	√	–	√	[231]
COBYLA	Local	√	–	√	[229,230,232]
DIRECT	Global	√	–	–	[230,232,233]
EGO	Global	√	–	–	[234,228]
ENTMOOT	Global	√	√	√	[235]
MISO	Global	√	√	–	[236]
MVRSM	Local	√	√	–	[237]
Nelder–Mead	Local	√	–	–	[232,230]
NEWUOA	Local	√	–	–	[229,230]
NOMAD	Local	√	√	√	[238]
NOWPAC	Local	√	–	√	[239]
PSO	Global	√	–	–	[240]
PSwarm	Global	√	–	√*	[241,242]
RBFSolve	Global	√	√	√	[228]
UOBYQA	Local	√	–	–	[229]

* Only linear constraints.

5.3 Applications to large-scale manufacturing systems

Many of the aforementioned data-driven optimization algorithms are used to address the optimization of various batch and continuous large-scale manufacturing processes. For example, the optimization of the pressure swing adsorption (PSA) systems for air separation or CO_2 capture is commonly studied since their formulations pose unique mathematical challenges due to the presence of nonlinear algebraic partial differential equations that typically span over multiple temporal and spatial scales. Trust–region approaches [243–246], the ARGONAUT algorithm [204], the NOMAD algorithm [247], and quadratic surrogate-based optimization [245] are indicative examples of various data-driven optimization algorithms used in this area. Other major applications for data-driven optimization include pharmaceutical manufacturing where kriging-based algorithms (e.g., EGO and its variants) are commonly used to optimize these processes [181,248–250], as well as the PSO algorithm [251]. Data-driven optimization is also used in oil production optimization and enhanced oil recovery systems with both direct search [90,252–255] and model-based algorithms

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[206,256–258]. Other indicative application areas include extractive distillation for ethanol purification [182], bioethanol processing [259], biodiesel production [167], hydrocracking optimization [260], steam cracking optimization [207], steam reforming [261,262], air separation units [263,264], and scheduling optimization of manufacturing systems [265].

5.4 Extensions to other classes of problems

Although a plethora of data-driven optimization algorithms are designed to address single objective problems as given in the form of Eq. (5.1), manufacturing systems can have multiple competing objectives (e.g., maximizing profit vs minimizing CO_2 emissions or maximizing selectivity vs maximizing conversion in reaction systems) or require the coordination of interconnected elements (e.g., production planning and scheduling) to improve productivity and resiliency against changing market conditions. These additional considerations affect the overall decision-making process and can only be captured with appropriate mathematical representations, such as multi-objective programming and multi-level or bi-level programming, respectively.

These tailored classes of optimization problems create further algorithmic challenges to address within data-driven optimization frameworks. In the case of multi-objective optimization, the trade-offs between conflicting objectives make it impossible to find a single optimal solution. Hence, one needs to derive the set of trade-off optimal solutions that simultaneously optimize multiple objectives. In the case of bi-level optimization, even the most simplistic formulations (i.e., bi-level linear programming) are proven to be strongly NP-hard [266,267]. These challenges motivated many researchers to develop different data-driven optimization capabilities to address extended classes of optimization problems that are highly representative of realistic decision-making processes.

Genetic and evolutionary algorithms, simulated annealing, and their derivatives have been at the forefront of addressing multi-objective optimization problems in manufacturing systems [268–275]. Optimization of adsorption systems [276–281], cryogenic air separation units [282], steam cracking process [91,283], and manufacturing process of carbon fiber [284] are some exemplary areas where these algorithms are successfully implemented. Most recent algorithmic developments in this field focused on surrogate-assisted hybrid evolutionary algorithms, where the first-principles representations are replaced with SVMs, ANNs, kriging, or RBF surrogates to enable the

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

fast computation of high-fidelity models, while the algorithm is searching for the optimal solution [91,277–281,284–286].

Likewise for bi-level optimization, most data-driven optimization algorithms implement genetic or evolutionary-based solution strategies [287]. Early algorithmic developments considered trust–region approaches, where linear or quadratic approximations are undertaken to represent different layers of optimization problems in a bi-level setting with a limited number of constraints [288,289]. Genetic and evolutionary algorithms for integer-linear [290,291] and mixed-integer nonlinear [292,293] formulations are also proposed. Recently, the DOMINO framework was developed which integrated constrained model-based and search-based data-driven algorithms to guarantee feasibility in bi-level mixed-integer nonlinear optimization problems [211]. The framework can handle several hundred variables and general constraints at both levels and is applicable to mixed-integer nonlinear lower levels; however, the upper-level complexity is limited to nonlinear programs. DOMINO is shown to be an effective algorithm for solving deterministic and stochastic integrated production planning and scheduling problems with multi-follower lower levels which are highly relevant to batch manufacturing systems [294,295]. Especially in the stochastic formulation, the product demand uncertainty is captured with scenario analysis, and the integrated planning and scheduling formulation is solved for every scenario to compute the total stochastic objective of the overall problem [295].

Although population-based algorithms dominate the data-driven optimization space due to their availability in various software platforms and ease of implementation, there are a few important factors to put in perspective when selecting algorithms for solving extended classes of optimization problems, such as the computation time and ability to handle constraints. Population-based algorithms require a large number of samples for convergence due to their stochastic nature, typically in the order of 10^3 and 10^6 samples, and this could be a limiting factor in any problem formulation with expensive simulations, even with parallel computing capabilities. Their biggest advantage is that, in multi-objective optimization problems, they can immediately and simultaneously deal with many objectives without requiring any discretizations or reformulations. Yet, the computational cost will increase exponentially with the increasing number of objectives in population-based algorithms [296]. Furthermore, the ability to handle constraints also plays an important role in all formulations, but these are far more critical in bi-level optimization problems. In bi-level programs, the optimality of the lower-level problem is posed as a constraint on the upper level, which implies

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

that the lower level needs to be globally optimal to be able to retrieve feasible solutions to the overall bi-level problem (Eq. (5.8)). However, population-based algorithms cannot guarantee optimality and hence feasibility to this class of problems. On the other hand, model-based algorithms explicitly account for each constraint by representing them with surrogate models and can guarantee feasibility with much lower sampling requirements. Yet, with the growing number of constraints, the number of surrogate models to train, validate, and test also grows, and choosing the appropriate surrogate representation can also be computationally challenging without any a priori insights from the underlying formulation.

In the following sections, we will demonstrate how a model-based global-constrained data-driven optimization algorithm, namely the ARGONAUT framework, can be used to solve deterministic multi-objective and bi-level optimization problems on motivating examples. The applications of these approaches on larger-scale problems can be found in their respective publications as noted below.

5.4.1 Data-driven multi-objective optimization with ϵ-constraint

In this section, we provide a brief demonstration of multi-objective optimization using the ϵ-constraint reformulation technique and the ARGONAUT algorithm, which is based on the earlier work of Beykal et al. [210]. As a motivating example, we will be using the test case 1 benchmark problem by Binh and Korn [297] shown below:

$$\begin{aligned} \min_{x_1,x_2} \; U = & \begin{cases} f_1 = (x_1-2)^2 + (x_2-1)^2 + 2 \\ f_2 = 9x_1 - (x_2-1)^2 \end{cases} \\ s.t. \; & x_1^2 + x_2^2 - 225 \leq 0 \\ & x_1 - 3x_2 + 10 \leq 0 \\ & x_1, x_2 \in [-20, 20] \end{aligned} \tag{5.3}$$

In order to tackle this problem with the ARGONAUT algorithm, Eq. (5.3) is reformulated into a series of single-objective optimization problems using the ϵ-constraint methodology [210,298] where one of the objectives (or more in the case of systems with more than two objective functions) is moved to the constraint set as an inequality that is less than or equal to ϵ. For the motivating example, we move f_2 to the constraints set and reduce the multi-objective optimization to a single-objective optimization

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

problem as follows:

$$\begin{aligned} \min_{x_1,x_2} f_1 &= (x_1-2)^2+(x_2-1)^2+2 \\ s.t.\ \ g_1 &: 9x_1-(x_2-1)^2-\epsilon \le 0 \\ g_2 &: x_1^2+x_2^2-225 \le 0 \\ g_3 &: x_1-3x_2+10 \le 0 \\ x_1, x_2 &\in [-20, 20], \quad \epsilon \in \left[\epsilon^L, \epsilon^U\right] \end{aligned} \tag{5.4}$$

The value of the ϵ becomes an upper bound on f_2 and can be used to discretize the objective space to map the Pareto front at each discretization point by solving Eq. (5.4). The lower and upper bounds on the ϵ parameter are derived by minimizing f_1 and f_2 individually as single-objective problems. To get the lower bound on ϵ, the following optimization problem is solved:

$$\begin{aligned} \min_{x_1,x_2} f_2 &= 9x_1-(x_2-1)^2 \\ s.t.\ x_1^2+x_2^2-225 &\le 0 \\ x_1-3x_2+10 &\le 0 \\ x_1, x_2 &\in [-20, 20] \end{aligned} \tag{5.5}$$

The optimal solution of Eq. (5.5) is $x_1^* = -4.8410, x_2^* = 14.1974$. At this solution, the minimum possible value of f_2 is reached at -217.7390, which is also the lower bound on ϵ.

On the contrary, to get the upper bound on ϵ, the following optimization problem is solved:

$$\begin{aligned} \min_{x_1,x_2} f_1 &= (x_1-2)^2+(x_2-1)^2+2 \\ s.t.\ x_1^2+x_2^2-225 &\le 0 \\ x_1-3x_2+10 &\le 0 \\ x_1, x_2 &\in [-20, 20] \end{aligned} \tag{5.6}$$

The optimal solution of Eq. (5.6) is $x_1^* = 1.1$, $x_2^* = 3.7$. As these two objectives are competing with each other, the minimum of the first objective function will give the maximum value of f_2, since increasing the value of f_2 beyond this maximum will not affect f_1. Hence, the upper bound on the ϵ is calculated by evaluating the second objective function at this solution as: $\epsilon^U = 9(1.1) - (3.7-1)^2 = 2.61$.

Once the lower and upper bounds are derived, 30 discretization points are created with varying ϵ values and Eq. (5.4) is solved at these 30 subproblems. It is important to note that more discretization points can be selected, yet this will increase the total number of subproblems to be solved to

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

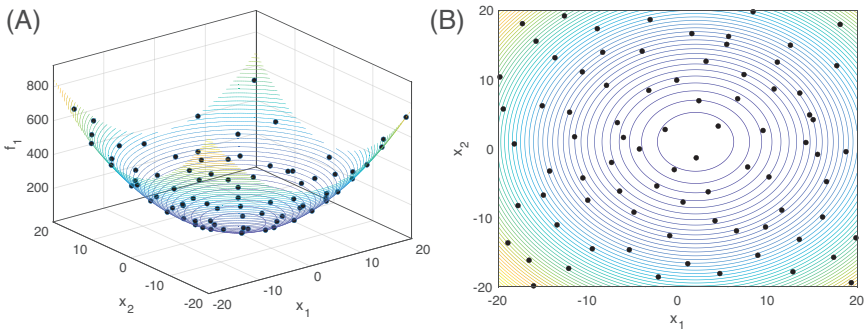


Figure 5.4 (A) 3D plot of the original objective function with the initial sampling points for the Latin hypercube design. (B) Top view of the original objective function and the initial sampling points.

generate the Pareto front. Also, an increasing number of objectives will result in a significant increase in the number of subproblems to address to retrieve the Pareto front, as more discretization points are needed to explore the additional dimension in the objective space. This is the major disadvantage of the ϵ-constraint approach.

To perform multi-objective optimization with the ARGONAUT algorithm, we assume that the objective function and the constraint expressions defining our subproblems are unknown. For demonstration purposes, we select the first discretization point with $\epsilon = 2.61$ and follow the step-wise procedure outlined below.

- *Step 1: Create a Latin hypercube design within the variable bounds*
  ARGONAUT creates a Latin hypercube design using the maximin approach to maximize the minimum distance between sampling points. The initial design follows the rule that was introduced by Beykal et al. [211], where $40k + 1$ samples are collected for gray-box problems with less than three decision variables ($k \leq 2$), $10k + 1$ samples for problems with $3 \leq k \leq 20$, and 251 samples for $k > 20$. As the motivating example has a dimensionality of $k = 2$, the first rule is utilized and a total of 81 initial sampling points are identified within the decision variable space (Fig. 5.4).
- *Step 2: Collect output data for surrogate modeling*
  Each sampling point is executed in the simulation file to get the output values of the objective function and the constraint violations of x_1 and x_2 pairs. The simulation file simply calculates output values and does not perform any optimization as we are assuming the functional forms are unknown. These input–output data are used to solve the parameter estimation problem to global optimality for surrogate modeling of the

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Table 5.2 First five rows of the input–output data collected to optimize the reformulated multi-objective optimization problem in Eq. (5.4).

x_1	x_2	f_1	g_1	g_2	g_3
15.116	4.207	123.145	21.193	12.493	184.314
5.471	15.100	−152.186	32.949	−29.830	212.865
−6.041	1.632	−57.378	−185.843	−0.938	67.057
−8.360	13.955	−245.680	39.629	−40.225	277.159
0.599	−7.718	−73.217	−165.080	33.752	79.960

Table 5.3 Surrogate model fitting results from the first iteration of ARGONAUT.

	Fitted surrogate	CVRMSE
Objective	$\hat{f}_1 = 1.788 - 2.711x_1 - 2.606x_2 + 2.087x_1^2 + 2.084x_2^2$	$2.41 \bullet 10^{-9}$
Constraint #1	$\hat{g}_1 : -0.236 + 0.479x_1 + 2.639x_2 - 2.11x_2^2 \leq 0.8796$	$5.41 \bullet 10^{-10}$
Constraint #2	$\hat{g}_2 : 1.613 - 2.538x_1 - 2.537x_2 + 2.117x_1^2 + 2.114x_2^2 \leq 0.3942$	$9.04 \bullet 10^{-9}$
Constraint #3	$\hat{g}_3 : 0.8996 + 0.266x_1 - 0.798x_2 \leq 0.5132$	$2.96 \bullet 10^{-10}$

unknown functions. An example of the input–output data is provided in Table 5.2.

- *Step 3: Cross-validated surrogate identification and parameter estimation*
  In the surrogate modeling phase, the input–output data are scaled and randomly partitioned to 90% training and 10% testing sets, and the parameter estimation problem is solved to global optimality over 5-fold cross-validation. For the motivating example,

  ARGONAUT identifies the surrogate models listed in Table 5.3 with minimal prediction error.
- *Step 4: Solving the gray-box optimization problem*
  The identified surrogates in Table 5.3 can now be used to formulate a gray-box optimization problem. It is important to note that the surrogate forms presented here are constructed with scaled variables. As a result, each function and the right-hand side of the constraints can be unscaled to switch to the original decision space. This resulting unscaled gray-box optimization problem is provided in Eq. (5.7), where the original set of equations that are provided in Eq. (5.4) is recovered by using only input–output data.

$$\begin{aligned} &\min_{x_1,x_2} f_1 = x_1^2 - 4x_1 + x_2^2 - 2x_2 + 7 \\ &s.t.\ 9x_1 - x_2^2 + 2x_2 - 3.61 \leq 0 \\ &\quad x_1^2 + x_2^2 - 225 \leq 0 \\ &\quad x_1 - 3x_2 + 10 \leq 0 \\ &\quad x_1, x_2 \in [-20, 20] \end{aligned} \tag{5.7}$$

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

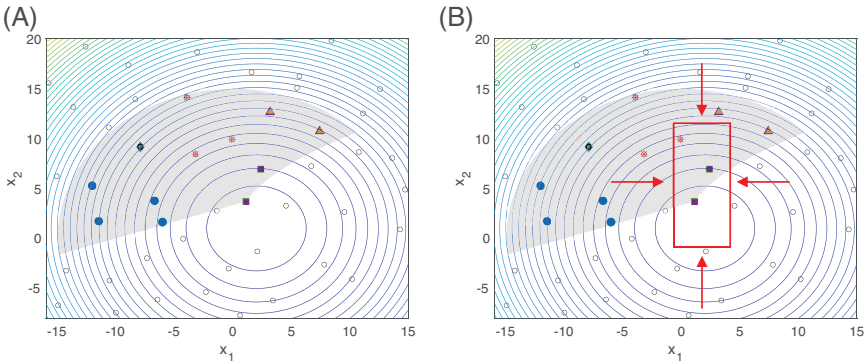


Figure 5.5 Clustering of the feasible points (shown in different shapes) for bound refinement with the best cluster shown in squares. Hollow circles show sampling points that are infeasible with respect to the modeled constraints. (A) The shaded area shows the feasible region defined by the fitted surrogates. (B) Bounds tightening around the best cluster.

This optimization problem (or the scaled one) is then solved to multiple local optima and the global optimum, and the unique optimal solutions become the new sampling points for the next iteration. At every iteration, the surrogate models are tuned with this new set of incoming samples, and the resulting gray-box formulation is solved repeatedly until a convergence criterion is met. Once convergence is reached within the initial bounds of the decision variables, a session is completed and ARGONAUT proceeds with a clustering-based bound refinement stage. Readers are guided by Boukouvala and Floudas [205] for ARGONAUT's convergence criteria.

- *Step 5: Bound refinement via hierarchical clustering*
  The feasible data points are clustered to identify the most promising region in the current search space using hierarchical clustering with complete linkage methodology (Fig. 5.5). For the motivating example, five clusters are identified and the variable bounds are tightened around the most promising cluster (i.e., the cluster with the best objective function value). In this new bounded region, ARGONAUT repeats its sampling, surrogate modeling, and global optimization steps. This second session of ARGONAUT terminates once the convergence is reached again, and the final solution is reported.
- *Step 6: Final solution and the Pareto front*
  ARGONAUT reports the global solution for the motivating subproblem as $x_1^* = 1.1$, $x_2^* = 3.7$ with the minimum objective function value of

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

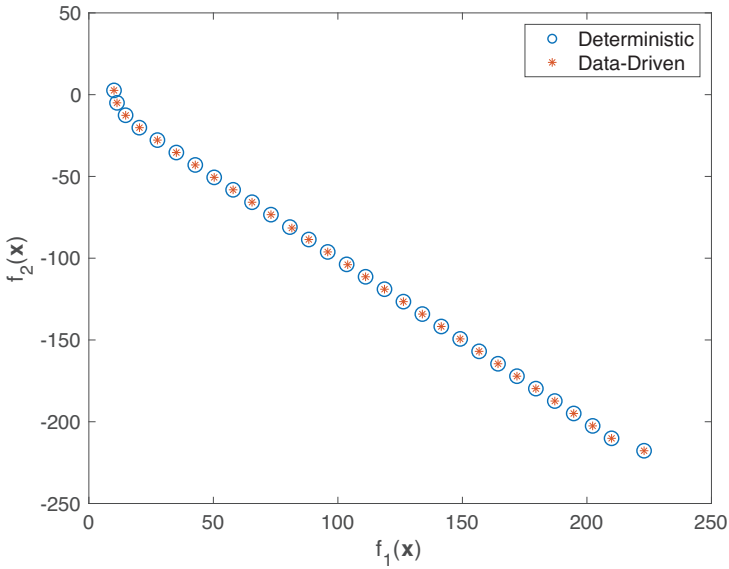


Figure 5.6 The Pareto solution for the motivating multi-objective example. The data-driven solution overlaps with the deterministic solution, showing the effectiveness of the approach.

10.1. This result is exactly the same as the deterministic solution; hence, we see that data-driven optimization via ARGONAUT can achieve the global solution which is consistent with its detailed benchmarking presented by Boukouvala and Floudas [205].

To generate the full Pareto front, steps 1 to 6 are repeated for each subproblem defined by the changing ϵ values. The final set of Pareto solutions over 30 discretization points is reported in Fig. 5.6. In the next section, the DOMINO algorithm is introduced to address another class of optimization problems, namely mixed-integer nonlinear bi-level optimization.

5.4.2 Data-driven mixed-integer nonlinear bi-level optimization with DOMINO

As shown in the previous section with multi-objective optimization, data-driven algorithms can be applied to a wide range of optimization problems with varying dimensionality, complexity, and type. In this section, we will be demonstrating the use of data-driven optimization algorithms for solving one of the most challenging classes of mathematical programming problems,

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

that is, bi-level optimization, based on the earlier work of Beykal et al. [211] and Avraamidou et al. [212].

Bi-level optimization problems are hierarchical programs that model the decision-making process in the presence of multiple decision makers with one optimization problem constrained by other optimization problems. An example of such a hierarchical structure is presented in Eq. (5.8), where the upper-level problem considers decisions $\boldsymbol{x}$ of continuous nature and constrained with inequality $\boldsymbol{G}(\boldsymbol{x}, \boldsymbol{y})$. This problem is also constrained by another optimization problem, the lower-level problem. The lower-level problem contains both continuous and integer decisions $\boldsymbol{y}$ with inequality constraints $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y})$ and equality constraints $\boldsymbol{h}(\boldsymbol{y})$.

$$\begin{aligned} \min_{x} \quad & F(\boldsymbol{x}, \boldsymbol{y}) \\ s.t. \quad & \boldsymbol{G}(\boldsymbol{x}, \boldsymbol{y}) \leq \boldsymbol{0} \\ & \boldsymbol{y} \in \arg\min_{y} \left\{ f(\boldsymbol{x}, \boldsymbol{y}) : g(\boldsymbol{x}, \boldsymbol{y}) \leq \boldsymbol{0}, \boldsymbol{h}(\boldsymbol{y}) = \boldsymbol{0} \right\} \\ & [x_1, \ldots, x_c] \in \mathbb{R}^c \\ & [y_1, \ldots, y_k] \in \mathbb{R}^k, [y_{k+1}, \ldots, y_i] \in \mathbb{Z}^{i-k} \end{aligned} \tag{5.8}$$

Here, we will be solving a bi-level optimization problem of the form provided in Eq. (5.8) with DOMINO, a data-driven optimization framework that reduces bi-level optimization problems to a single-level problem using a smart sampling approach and solves the resulting data-driven optimization problem to guaranteed feasibility [211,212]. DOMINO is a flexible framework that allows for the integration of various data-driven optimization solvers within its structure. For this purpose, we selected ARGONAUT as our data-driven optimization solver and demonstrate the algorithmic steps of DOMINO–ARGONAUT with the following motivating example retrieved from Sahin and Ciric [299].

- *Step 1: Create a Latin hypercube design for the upper-level decision variables*

$$\begin{aligned} \min_{x_1, x_2} \quad & \left(0.4x_1^2x_2 - 4x_2^2\right)y_1y_2 - \left(-x_2^3 + 3x_1^2x_2\right)(1 - y_1)y_2 - \left(2x_2^2 - x_1\right)(1 - y_2) \\ \text{s.t.} \quad & \min_{y_1, y_2} \quad -\left(x_1^2x_2^2 + 8x_2^3 - 14x_1^2 - 5x_1\right)y_1y_2 - \left(-x_1x_2^2 + 5x_1x_2 + 4x_2\right)(1 - y_1)y_2 \\ & \qquad + 8x_1y_1(1 - y_2) \\ & \text{s.t.} \quad 1 - y_1 - y_2 \leq 0 \\ & \boldsymbol{x} \in [0, 10]^2, \boldsymbol{y} \in \{0, 1\} \end{aligned} \tag{5.9}$$

The motivating example (Eq. 5.9) is a nonlinear program (NLP) constrained with an integer nonlinear program (INLP) containing two bounded continuous decision variables at the upper level, two binary variables at the lower level, and a constraint at the lower-level problem.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

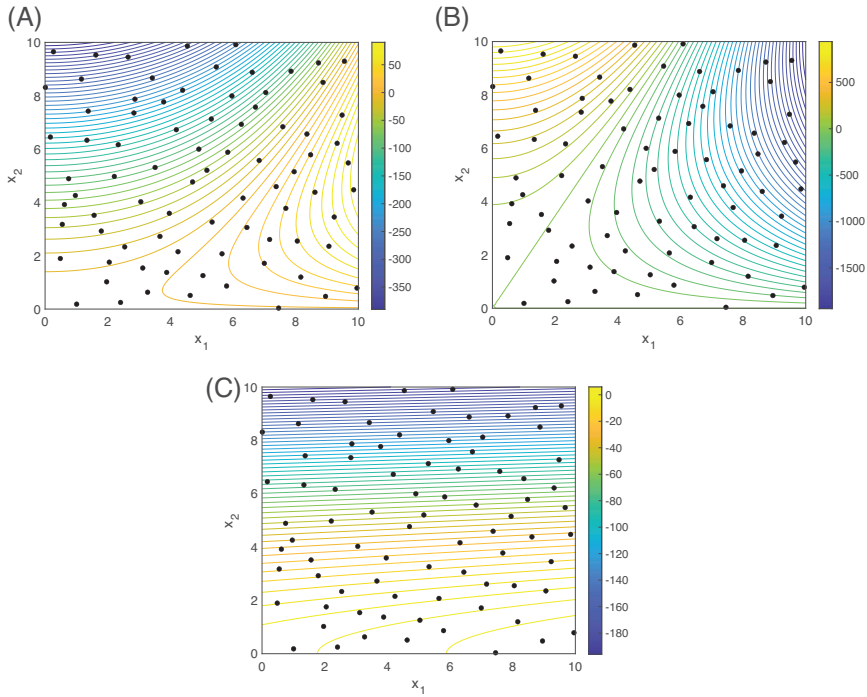


Figure 5.7 Plots of the upper-level objective function with different combinations of the lower-level decision variables: (A) $y_1 = 1, y_2 = 1$; (B) $y_1 = 0, y_2 = 1$; and (C) $y_1 = 0, y_2 = 0$ or $y_1 = 1, y_2 = 0$. The dots represent the Latin hypercube sampling design generated for the upper-level decision variables in the motivating example.

The first step of the framework is to create an initial sampling design only in the upper-level decision variable space. Depending on the type of the embedded data-driven optimizer, the initial sampling design may not be required and an initial starting point would suffice to start the algorithm. In this case, DOMINO benefits from the algorithmic capabilities of ARGONAUT and creates a maximin Latin hypercube sampling design with 81 samples, following the rule-of-thumb presented in Section 5.4.1 (Fig. 5.7).

- *Step 2: Solve the lower-level problem to deterministic global optimality at every sampling point*

  Once the initial design is constructed, the next step is to collect the corresponding output information for surrogate modeling. This step is not as straightforward as it was in the multi-objective motivating example. As seen in Eq. (5.9), the variables of both levels appear in the upper-

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Table 5.4 First five rows of the input–output data collected to optimize the bi-level optimization problem in Eq. (5.9).

x_1	x_2	obj
2.835	7.353	–192.608
0.967	4.259	–70.955
6.715	7.572	–92.792
3.881	1.374	0.106
9.322	6.216	61.499

and lower-level objective functions. As the lower-level variables are not sampled, one needs to retrieve this information to be able to collect any output data. To achieve this, every set of x_1 and x_2 values in the sampling design is fed to the lower-level problem, and the resulting single-level optimization problem is solved to global optimality using a deterministic global optimization solver. To demonstrate this, let us observe the first sample obtained from Step 1 which is $x_1 = 2.835$ and $x_2 = 7.353$. These numbers are fixed at the lower level, which results in the following optimization problem at the lower level:

$$\begin{aligned} \min_{y_1, y_2} \quad & 19.638y_2 - 22.68y_1 - 3485.222y_1y_2 \\ \text{s.t.} \quad & 1 - y_1 - y_2 \leq 0 \\ & \boldsymbol{y} \in \{0, 1\} \end{aligned} \tag{5.10}$$

By fixing the values of x_1 and x_2, the optimization problem is reduced to a single-level problem with variables y_1 and y_2 being the sole decision variables. Now, the optimization problem shown in Eq. (5.10) can be solved using an appropriate deterministic global optimizer to retrieve the optimal solution of $y_1^* = 1$ and $y_2^* = 1$.

- *Step 3: Retrieve the output data for surrogate modeling*
  The optimal values of y_1^* and y_2^* are then used to enumerate the upper-level objective function value by substituting the values of $x_1 = 2.835$, $x_2 = 7.353$, $y_1^* = 1$, and $y_2^* = 1$ which results in $obj = -192.608$. For the given set of input sampling points, the corresponding output information is now retrieved. To generate the full input–output data set, the full set of samples is evaluated following this procedure and their output information is collected (Table 5.4). It is important to note that in this motivating example, the output information entailed only the value of the upper-level objective function. However, constraints of any level

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

that contain at least one upper-level variable will be treated as a gray-box constraint within DOMINO, and their output information will be collected using the described recipe. For the motivating example, only one constraint is present at the lower level which does not contain upper-level decision variables. Hence, this constraint is automatically handled when the lower-level problem is solved to global optimality and its violation is not retrieved as output information for the data-driven optimization.

- *Step 4: Cross-validated surrogate identification and parameter estimation with ARGONAUT*
  As the DOMINO framework is benefiting from the algorithmic capabilities of ARGONAUT, the input–output data are passed to the cross-validated parameter estimation step to generate the surrogate model for the unknown objective function. The exact procedure described in Step 3 of Section 5.4.1 is followed to create a highly accurate surrogate model, which for the motivating example is a radial basis function in the first iteration. When compared to other surrogate models available in ARGONAUT's library, we observe that a linear surrogate yields a CVRMSE of 0.11, a quadratic surrogate yields a CVRMSE of 0.034, and a kriging surrogate yields a CVRMSE of 0.026. However, the fitted radial basis surrogate yields a CVRMSE of 0.007, which is the lowest among all. As a result, ARGONAUT selects the radial basis function with a thin plate spline basis to represent the unknown objective function. It is important to note that the selection of the surrogate model can change across different iterations of the algorithm. This is a natural outcome, as more samples are collected, the underlying patterns in the data set become more apparent and another surrogate model may represent the data set with better accuracy. In the motivating example, we observe that the radial basis function is selected for the first two iterations, whereas a quadratic surrogate is selected in the final iteration with a CVRMSE of 0.002 and the following functional form in the unscaled space:

$$\hat{obj} = 3.452x_1^2 - 3.950x_2^2 + 1.921x_1x_2 - 16.603x_1 - 2.376x_2 + 16.804 \tag{5.11}$$

- *Step 5: Solving the gray-box optimization problem and convergence of the data-driven algorithm*
  Next, the constructed surrogate model is embedded in a gray-box optimization problem to recover the new sampling points to be explored for the minimization of the original optimization problem. Note that

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

this optimization problem is a single-level problem that is constructed using the bi-level information of the original optimization problem. By solving this problem to multiple local optima and the global optimum, the unique solutions are retrieved as new sampling points, and the algorithmic steps of ARGONAUT, which are outlined in Section 5.4.1, are followed to recover the optimal solution of the data-driven system. At the end of ARGONAUT's algorithmic iterations, an optimal solution of $x_1^* = 0, x_2^* = 10$ with an objective function value of -400 is reported for the motivating bi-level example.

- *Step 6: Checking the gray-box constraints for violations*
  Once the optimal solution for the system is recovered by the data-driven optimizer, DOMINO proceeds with two checks to guarantee feasibility. The first check of the system is to ensure that the data-driven algorithm converged to a feasible solution. If any of the gray-box constraints are violated, the solution is considered infeasible and the DOMINO is restarted with a new sampling design. In the case of our motivating bi-level example, there are no gray-box constraints to be handled; hence, the final solution is feasible for the reduced single-level data-driven optimization problem.
- *Step 7: Checking the optimality status of the lower-level problem*
  Although the gray-box constraints are satisfied, the overall bi-level optimization problem can still be infeasible if the lower-level optimization problem is not globally optimal at the final reported solution. To ensure bi-level feasibility, the optimal solution of the upper-level problem x_1^* and x_2^* is fixed at the lower level and the optimality status of the problem is retrieved. In the motivating example, the final reported solution $x_1^* = 0$, $x_2^* = 10$, $y_1^* = 1$, $y_2^* = 1$ is globally optimal at the lower level and feasible at the upper level, which means that this solution is guaranteed feasible for the overall bi-level problem. Also, when we compare this result with the known global solution [300], we observe that the error is 0% and the globally optimal solution is identified by DOMINO. This shows that DOMINO can accurately retrieve guaranteed feasible, optimal, or near-optimal solutions for bi-level optimization problems. It is also important to note that the algorithmic capabilities of DOMINO are not limited to NLP–INLP formulations. Beykal et al. [211] have previously shown that DOMINO can be used to address LP–LP, LP–QP, LP–NLP, QP–QP, QP–NLP, NLP–QP, NLP–NLP, LP–ILP, LP–MILP, LP–MIQP, QP–MILP, QP–IQP, QP–MIQP, NLP–MILP, and NLP–MIQP type problems.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

5.5 Remarks

- Data-driven optimization has been an emerging field of sciences, engineering, and applied mathematics where many algorithmic advances are presented throughout the years.
- Algorithmic advances to date address unconstrained, box-constrained, and general constrained linear, mixed-integer linear, quadratic, and nonlinear optimization problems.
- Recent trends in data-driven optimization show that the current algorithmic advances mainly focus on addressing constrained mixed-integer nonlinear programming formulations using model-based approaches.
- Data-driven frameworks are also developed to solve high-dimensional and tailored classes of problems such as multi-objective optimization and bi-level optimization.
- More complex surrogate models, including ANNs and ensemble models, are becoming a part of data-driven optimization frameworks due to their flexibility and ability to handle mixed variable information.
- In model-based approaches, the meticulous selection and the construction of the surrogate model in discontinuous spaces without assuming the continuity of the unknown functions are yet to be addressed.
- Other open issues in data-driven optimization include handling high-dimensional problems with dimensions extending from 50 to 200 variables or more, handling stochastic and dynamic optimization problems, and addressing uncertainty in surrogate model parameters.
- The role of data-driven optimization in multi-scale systems has not been fully explored. Especially, bridging the gap across different temporal and spatial scales with surrogate modeling is important.
- Future directions of data-driven optimization algorithms will focus on hybrid modeling applications and will address the fundamental question of how to create models that contain both explicitly known components and components that are purely provided in the form of data.

5.6 Conclusion

Data-driven optimization is gaining momentum in smart manufacturing applications for the optimization of computationally expensive digital twins. As manufacturing systems are becoming hybrid through a combination of first-principles expressions and equation-free data, mathematical

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

optimization becomes a challenge due to the lack of explicit forms. Data-driven optimization emerged as an alternative to traditional optimization practices by developing efficient algorithms that use data to explore the best solution of a given problem. A plethora of algorithmic approaches exist for data-driven optimization, including sample-based direct search algorithms, hybrid techniques that balance exploration and exploitation, and model-based approaches that rely on surrogate models. These techniques address unconstrained, box-constrained, or general constrained problems with continuous, nonlinear, and/or mixed-integer formulations in local or global optimization contexts. The development of data-driven algorithms is not limited to single-level or single-objective optimization problems. Recent advances have also shown that data-driven optimization can also address other classes of problems, such as multi-objective optimization and bi-level optimization.

References

[1] M. Liu, S. Fang, H. Dong, C. Xu, Review of digital twin about concepts, technologies, and industrial applications, J. Manuf. Syst. 58 (2021) 346–361.
[2] L.M. Rios, N.V. Sahinidis, Derivative-free optimization: a review of algorithms and comparison of software implementations, J. Global Optim. 56 (3) (2013) 1247–1293.
[3] F. Boukouvala, R. Misener, C.A. Floudas, Global optimization advances in mixed-integer nonlinear programming, MINLP, and constrained derivative-free optimization, CDFO, Eur. J. Oper. Res. 252 (3) (2016) 701–727.
[4] A. Bhosekar, M. Ierapetritou, Advances in surrogate based modeling, feasibility analysis, and optimization: a review, Comput. Chem. Eng. 108 (2018) 250–267.
[5] A.R. Conn, K. Scheinberg, L.N. Vicente, Introduction to Derivative-Free Optimization, SIAM, 2009.
[6] C. Audet, W. Hare, Derivative-Free and Blackbox Optimization, Springer, 2017.
[7] M.J. Powell, Direct search algorithms for optimization calculations, Acta Numer. 7 (1998) 287–336.
[8] R.M. Lewis, V. Torczon, M.W. Trosset, Direct search methods: then and now, J. Comput. Appl. Math. 124 (1-2) (2000) 191–207.
[9] T.G. Kolda, R.M. Lewis, V. Torczon, Optimization by direct search: new perspectives on some classical and modern methods, SIAM Rev. 45 (3) (2003) 385–482.
[10] N.V. Queipo, R.T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, P.K. Tucker, Surrogate-based analysis and optimization, Prog. Aerosp. Sci. 41 (1) (2005) 1–28.
[11] A.I. Forrester, A.J. Keane, Recent advances in surrogate-based optimization, Prog. Aerosp. Sci. 45 (1-3) (2009) 50–79.
[12] S. Koziel, D.E. Ciaurri, L. Leifsson, Surrogate-based methods, Computational Optimization, Methods and Algorithms, Springer, 2011, pp. 33–59.
[13] K.K. Vu, C. d'Ambrosio, Y. Hamadi, L. Liberti, Surrogate-based methods for black-box optimization, Int. Trans. Oper. Res. 24 (3) (2017) 393–424.
[14] J. Larson, M. Menickelly, S.M. Wild, Derivative-free optimization methods, Acta Numer. 28 (2019) 287–404.
[15] K. McBride, K. Sundmacher, Overview of surrogate modeling in chemical process engineering, Chem. Ing. Tech. 91 (3) (2019) 228–239.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[16] M. Xi, W. Sun, J. Chen, Survey of derivative-free optimization, Numer. Algebra Control Optim 10 (4) (2020) 537.
[17] J.A. Nelder, R. Mead, A simplex method for function minimization, Comput. J. 7 (4) (1965) 308–313.
[18] W. Spendley, G.R. Hext, F.R. Himsworth, Sequential application of simplex designs in optimisation and evolutionary operation, Technometrics 4 (4) (1962) 441–461.
[19] K.I. McKinnon, Convergence of the Nelder–Mead simplex method to a nonstationary point, SIAM J. Optim. 9 (1) (1998) 148–158.
[20] J.C. Lagarias, J.A. Reeds, M.H. Wright, P.E. Wright, Convergence properties of the Nelder–Mead simplex method in low dimensions, SIAM J. Optim. 9 (1) (1998) 112–147.
[21] P.C. Wang, T.E. Shoup, Parameter sensitivity study of the Nelder–Mead simplex method, Adv. Eng. Software 42 (7) (2011) 529–533.
[22] P. Tseng, Fortified-descent simplicial search method: a general approach, SIAM J. Optim. 10 (1) (1999) 269–288.
[23] R.R. Barton, J.S. Ivey Jr, Modifications of the Nelder-Mead Simplex Method for Stochastic Simulation Response Optimization, Tech. Rep., Institute of Electrical and Electronics Engineers (IEEE), 1991.
[24] J.J. Tomick, S.F. Arnold, R.R. Barton, Sample size selection for improved Nelder-Mead performance, in: Winter Simulation Conference Proceedings, 1995, IEEE, 1995, pp. 341–345.
[25] J.J. Tomick, On Convergence of the Nelder-Mead Simplex Algorithm for Unconstrained Stochastic Optimization, Tech. Rep., Air Force Inst of Tech Wright-Patterson Afb OH, 1995.
[26] R.R. Barton, J.S. Ivey Jr, Nelder-Mead simplex modifications for simulation optimization, Manag. Sci. 42 (7) (1996) 954–973.
[27] K.-H. Chang, Stochastic Nelder–Mead simplex method: a new globally convergent direct search method for simulation optimization, Eur. J. Oper. Res. 220 (3) (2012) 684–694.
[28] A. Moraglio, C.G. Johnson, Geometric generalization of the Nelder-Mead algorithm, in: European Conference on Evolutionary Computation in Combinatorial Optimization, Springer, 2010, pp. 190–201.
[29] E. Brea, An extension of Nelder-Mead method to nonlinear mixed-integer optimization problems, Rev. Int. Métodos Numér. Cálc. Diseño Ing. 29 (3) 163–174.
[30] D. Lee, M. Wiswall, A parallel implementation of the simplex function minimization routine, Comput. Econ. 30 (2) (2007) 171–187.
[31] K. Klein, J. Neira, Nelder-Mead simplex optimization routine for large-scale problems: a distributed memory implementation, Comput. Econ. 43 (4) (2014) 447–461.
[32] M.A. Luersen, R.Le Riche, Globalized Nelder–Mead method for engineering optimization, Comput. Struct. 82 (23-26) (2004) 2251–2260.
[33] M.A. Luersen, R. Le Riche, F. Guyon, A constrained, globalized, and bounded Nelder–Mead method for engineering optimization, Struct. Multidiscip. Optim. 27 (1-2) (2004) 43–54.
[34] S. Singer, S. Singer, Complexity analysis of Nelder-Mead search iterations, in: Proceedings of the Conference on Applied Mathematics and Computation, 1999, pp. 185–196.
[35] S. Wessing, Proper initialization is crucial for the Nelder–Mead simplex search, Optim. Lett. 13 (4) (2019) 847–856.
[36] L. Han, M. Neumann, Effect of dimensionality on the Nelder–Mead simplex method, Optim. Methods Softw. 21 (1) (2006) 1–16.
[37] F. Gao, L. Han, Implementing the Nelder-Mead simplex algorithm with adaptive parameters, Comput. Optim. Appl. 51 (1) (2012) 259–277.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[38] I. Fajfar, Á. Bürmen, J. Puhan, The Nelder–Mead simplex algorithm with perturbed centroid for high-dimensional function optimization, Optim. Lett. 13 (5) (2019) 1011–1025.
[39] C.T. Kelley, Detection and remediation of stagnation in the Nelder–Mead algorithm using a sufficient decrease condition, SIAM J. Optim. 10 (1) (1999) 43–55.
[40] C.J. Price, I.D. Coope, D. Byatt, A convergent variant of the Nelder–Mead algorithm, J. Optim. Theory Appl. 113 (1) (2002) 5–19.
[41] L. Nazareth, P. Tseng, Gilding the lily: a variant of the Nelder-Mead algorithm based on golden-section search, Comput. Optim. Appl. 22 (1) (2002) 133–144.
[42] Á. Bu″rmen, J. Puhan, T. Tuma, Grid restrained Nelder-Mead algorithm, Comput. Optim. Appl. 34 (3) (2006) 359–375.
[43] Á. Bu″rmen, T. Tuma, Unconstrained derivative-free optimization by successive approximation, J. Comput. Appl. Math. 223 (1) (2009) 62–74.
[44] L. L´ocsi, A hyperbolic variant of the Nelder–Mead simplex method in low dimensions, Acta Univ. Sapient. Math. 5 (2) (2013) 169–183.
[45] V. Mehta, Improved Nelder–Mead algorithm in high dimensions with adaptive parameters based on Chebyshev spacing points, Eng. Optim. 52 (10) (2020) 1814–1828.
[46] R. Hooke, T.A. Jeeves, "Direct search" solution of numerical and statistical problems, J. ACM (JACM) 8 (2) (1961) 212–229.
[47] W.H. Swann, Direct search methods, in numerical methods for unconstrained optimization, W. Murray, Ed., Academic Press, London, New York, 1972, pp. 13–28.
[48] J.E. Dennis Jr, V. Torczon, Direct search methods on parallel machines, SIAM J. Optim. 1 (4) (1991) 448–474.
[49] V. Torczon, On the convergence of pattern search algorithms, SIAM J. Optim. 7 (1) (1997) 1–25.
[50] M.A. Abramson, Pattern Search Algorithms for Mixed Variable General Constrained Optimization Problems, Rice University, 2003.
[51] R.M. Lewis, V. Torczon, Rank ordering and positive bases in pattern search algorithms., Tech. Rep., Institute for Computer Applications in Science and Engineering Hampton VA, 1996.
[52] C. Bogani, M.G. Gasparo, A. Papini, Pattern search method for discrete L1–approximation, J. Optim. Theory Appl. 134 (1) (2007) 47–59.
[53] C. Bogani, M. Gasparo, A. Papini, Generalized pattern search methods for a class of nonsmooth optimization problems with structure, J. Comput. Appl. Math. 229 (1) (2009) 283–293.
[54] R.M. Lewis, V. Torczon, Pattern search algorithms for bound constrained minimization, SIAM J. Optim. 9 (4) (1999) 1082–1099.
[55] R.M. Lewis, V. Torczon, Pattern search methods for linearly constrained minimization, SIAM J. Optim. 10 (3) (2000) 917–941.
[56] R.M. Lewis, V. Torczon, A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds, SIAM J. Optim. 12 (4) (2002) 1075–1089.
[57] C. Audet, J.E. Dennis Jr, A pattern search filter method for nonlinear programming without derivatives, SIAM J. Optim. 14 (4) (2004) 980–1010.
[58] C. Audet, J.E. Dennis Jr, Analysis of generalized pattern searches, SIAM J. Optim. 13 (3) (2002) 889–903.
[59] C. Audet, Convergence results for generalized pattern search algorithms are tight, Optim. Eng. 5 (2) (2004) 101–122.
[60] M.A. Abramson, C. Audet, J.E. Dennis, Generalized pattern searches with derivative information, Math. Program. 100 (1) (2004) 3–25.
[61] I.D. Coope, C.J. Price, A direct search conjugate directions algorithm for unconstrained minimization, ANZIAM J. 42 (2000) C478–C498.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[62] I.D. Coope, C.J. Price, On the convergence of grid-based methods for unconstrained optimization, SIAM J. Optim. 11 (4) (2001) 859–869.
[63] U.M. García-Palomares, J.F. Rodríguez, New sequential and parallel derivative-free algorithms for unconstrained minimization, SIAM J. Optim. 13 (1) (2002) 79–96.
[64] S. Lucidi, M. Sciandrone, On the global convergence of derivative-free methods for unconstrained optimization, SIAM J. Optim. 13 (1) (2002) 97–116.
[65] C. Bogani, M.G. Gasparo, A. Papini, Generating set search methods for piecewise smooth problems, SIAM J. Optim. 20 (1) (2009) 321–335.
[66] T.G. Kolda, R.M. Lewis, V. Torczon, Stationarity results for generating set search for linearly constrained optimization, SIAM J. Optim. 17 (4) (2007) 943–968.
[67] R.M. Lewis, A. Shepherd, V. Torczon, Implementing generating set search methods for linearly constrained minimization, SIAM J. Sci. Comput. 29 (6) (2007) 2507–2530.
[68] J.D. Griffin, T.G. Kolda, R.M. Lewis, Asynchronous parallel generating set search for linearly constrained optimization, SIAM J. Sci. Comput. 30 (4) (2008) 1892–1924.
[69] M.A. Abramson, L. Frimannslund, T. Steihaug, A subclass of generating set search with convergence to second-order stationary points, Optim. Methods Softw. 29 (5) (2014) 900–918.
[70] T.G. Kolda, R.M. Lewis, V. Torczon, et al., A Generating Set Direct Search Augmented Lagrangian Algorithm for Optimization With a Combination of General and Linear Constraints, Tech. Rep., Technical Report SAND2006-5315, Sandia National Laboratories (2006).
[71] C. Audet, J.E. Dennis Jr, Mesh adaptive direct search algorithms for constrained optimization, SIAM J. Optim. 17 (1) (2006) 188–217.
[72] C. Audet, J.E. Dennis Jr, A progressive barrier for derivative-free nonlinear programming, SIAM J. Optim. 20 (1) (2009) 445–472.
[73] M.A. Abramson, C. Audet, J.E. Dennis Jr, S.L. Digabel, OrthoMADS: a deterministic MADS instance with orthogonal directions, SIAM J. Optim. 20 (2) (2009) 948–966.
[74] S. Le Digabel, Algorithm 909: NOMAD: Nonlinear optimization with the MADS algorithm, ACM Trans. Math. Softw. (TOMS) 37 (4) (2011) 1–15.
[75] A.R. Conn, S.L. Digabel, Use of quadratic models with mesh-adaptive direct search for constrained black box optimization, Optim. Methods Softw. 28 (1) (2013) 139–158.
[76] R.B. Gramacy, S.L. Digabel, The mesh adaptive direct search algorithm with treed Gaussian process surrogates, Pacific J. Optim. 11 (3) (2015) 419–447.
[77] C. Audet, S.L. Digabel, C. Tribes, The mesh adaptive direct search algorithm for granular and discrete variables, SIAM J. Optim. 29 (2) (2019) 1164–1189.
[78] M.-K. Seo, T.-Y. Lee, J.-W. Kim, Y.-J. Kim, S.-Y. Jung, Principal component optimization with mesh adaptive direct search for optimal design of IPMSM, IEEE Trans. Magn. 53 (6) (2017) 1–4.
[79] A.L. Custódio, J.A. Madeira, GLODS: global and local optimization using direct search, J. Global Optim. 62 (1) (2015) 1–28.
[80] A.L. Custódio, J.A. Madeira, Multiglods: global and local multiobjective optimization using direct search, J. Global Optim. 72 (2) (2018) 323–345.
[81] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-International Conference on Neural Networks, 4, IEEE, 1995, pp. 1942–1948.
[82] M. Clerc, Particle Swarm Optimization, 93, John Wiley & Sons, 2010.
[83] A.E. Olsson, Particle Swarm Optimization: Theory, Techniques and Applications, Nova Science Publishers, Inc., 2010.
[84] J. Sun, C.-H. Lai, X.-J. Wu, Particle Swarm Optimisation: Classical and Quantum Perspectives, CRC Press, 2012.
[85] J.C. Bansal, P.K. Singh, N.R. Pal, Evolutionary and Swarm Intelligence Algorithms, Springer, 2019.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[86] Y. Shi, et al., Particle swarm optimization: developments, applications and resources, in: Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546), 1, IEEE, 2001, pp. 81–86.
[87] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization, Swarm Intell. 1 (1) (2007) 33–57.
[88] M.R. Bonyadi, Z. Michalewicz, Particle swarm optimization for single objective continuous space problems: a review, Evol. Comput. 25 (1) (2017) 1–54.
[89] E.H. Houssein, A.G. Gad, K. Hussain, P.N. Suganthan, Major advances in particle swarm optimization: theory, analysis, and application, Swarm Evol. Comput. 63 (2021) 100868.
[90] N. Sorek, E. Gildin, F. Boukouvala, B. Beykal, C.A. Floudas, Dimensionality reduction for production optimization using polynomial approximations, Comput. Geosci. 21 (2) (2017) 247–266.
[91] K. Bi, B. Beykal, S. Avraamidou, I. Pappas, E.N. Pistikopoulos, T. Qiu, Integrated modeling of transfer learning and intelligent heuristic optimization for a steam cracking process, Ind. Eng. Chem. Res. 59 (37) (2020) 16357–16367.
[92] L. dos Santos Coelho, An efficient particle swarm approach for mixed-integer programming in reliability–redundancy optimization applications, Reliab. Eng. Syst. Saf. 94 (4) (2009) 830–837.
[93] C. Jia, Y. Zhang, Y. Zeng, C. Yuan, An improved particle swarm optimization algorithm for solving mixed integer programming problems, in: 2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics, 2, IEEE, 2015, pp. 472–475.
[94] F. Wang, H. Zhang, A. Zhou, A particle swarm optimization algorithm for mixed-variable optimization problems, Swarm Evol. Comput. 60 (2021) 100808.
[95] L.-B. Zhang, C.-G. Zhou, X. Liu, Z. Ma, M. Ma, Y. Liang, Solving multi objective optimization problems using particle swarm optimization, in: The 2003 Congress on Evolutionary Computation, 4, IEEE, 2003, pp. 2400–2405.
[96] J.J. Liang, B.-Y. Qu, P.N. Suganthan, B. Niu, Dynamic multi-swarm particle swarm optimization for multi-objective optimization problems, in: 2012 IEEE Congress on Evolutionary Computation, IEEE, 2012, pp. 1–8.
[97] X. Zhang, H. Liu, L. Tu, A modified particle swarm optimization for multimodal multi-objective optimization, Eng. Appl. Artif. Intell. 95 (2020) 103905.
[98] R. Kuo, C. Huang, Application of particle swarm optimization algorithm for solving bi-level linear programming problem, Comput. Math. Appl. 58 (4) (2009) 678–685.
[99] Y. Gao, G. Zhang, J. Lu, H.-M. Wee, Particle swarm optimization for bi-level pricing problems in supply chains, J. Global Optim. 51 (2) (2011) 245–254.
[100] W. Du, B. Li, Multi-strategy ensemble particle swarm optimization for dynamic optimization, Inf. Sci. 178 (15) (2008) 3096–3109.
[101] C. Li, S. Yang, A clustering particle swarm optimizer for dynamic optimization, in: 2009 IEEE Congress on Evolutionary Computation, IEEE, 2009, pp. 439–446.
[102] S. Dennis, A. Engelbrecht, A review and empirical analysis of particle swarm optimization algorithms for dynamic multi-modal optimization, in: 2020 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2020, pp. 1–8.
[103] D.R. Jones, C.D. Perttunen, B.E. Stuckman, Lipschitzian optimization without the Lipschitz constant, J. Optim. Theory Appl. 79 (1) (1993) 157–181.
[104] D.R. Jones, J.R. Martins, The DIRECT algorithm: 25 years later, J. Global Optim. 79 (3) (2021) 521–566.
[105] B.O. Shubert, A sequential method seeking the global maximum of a function, SIAM J. Numer. Anal. 9 (3) (1972) 379–388.
[106] W. Huyer, A. Neumaier, Global optimization by multilevel coordinate search, J. Global Optim. 14 (4) (1999) 331–355.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[107] J.M. Gablonsky, C.T. Kelley, A locally-biased form of the direct algorithm, J. Global Optim. 21 (1) (2001) 27–37.
[108] L. Stripinis, R. Paulavičius, J. Žilinskas, Improved scheme for selection of potentially optimal hyper-rectangles in direct, Optim. Lett. 12 (7) (2018) 1699–1712.
[109] D.R. Jones, Direct global optimization algorithm, in: C.A. Floudas, P.M. Pardalos (Eds.), Encyclopedia of Optimization, Springer US, Boston, MA, 2001, pp. 431–440.
[110] G. Di Pillo, S. Lucidi, F. Rinaldi, An approach to constrained global optimization based on exact penalty functions, J. Global Optim. 54 (2) (2012) 251–260.
[111] G. Di Pillo, S. Lucidi, F. Rinaldi, A derivative-free algorithm for constrained global optimization based on exact penalty functions, J. Optim. Theory Appl. 164 (3) (2015) 862–882.
[112] H. Liu, S. Xu, X. Chen, X. Wang, Q. Ma, Constrained global optimization via a direct-type constraint-handling technique and an adaptive metamodeling strategy, Struct. Multidiscip. Optim. 55 (1) (2017) 155–177.
[113] M.F.P. Costa, A.M.A. Rocha, E.M. Fernandes, Filter-based DIRECT method for constrained global optimization, J. Global Optim. 71 (3) (2018) 517–536.
[114] Y.D. Sergeyev, D.E. Kvasov, Global search based on efficient diagonal partitions and a set of Lipschitz constants, SIAM J. Optim. 16 (3) (2006) 910–937.
[115] R. Paulavičius, Y.D. Sergeyev, D.E. Kvasov, J. Žilinskas, Globally-biased DISIMPL algorithm for expensive global optimization, J. Global Optim. 59 (2-3) (2014) 545–567.
[116] Q. Liu, W. Cheng, A modified direct algorithm with bilevel partition, J. Global Optim. 60 (3) (2014) 483–499.
[117] H. Liu, S. Xu, X. Wang, J. Wu, Y. Song, A global optimization algorithm for simulation-based problems via the extended direct scheme, Eng. Optim. 47 (11) (2015) 1441–1458.
[118] R. Paulavičius, L. Chiter, J. Žilinskas, Global optimization based on bisection of rectangles, function values at diagonals, and a set of Lipschitz constants, J. Global Optim. 71 (1) (2018) 5–20.
[119] R. Paulavičius, Y.D. Sergeyev, D.E. Kvasov, J. Žilinskas, Globally-biased BIRECT algorithm with local accelerators for expensive global optimization, Expert Syst. Appl. 144 (2020) 113052.
[120] Y.-S. Lai, J.D. Achenbach, Direct search optimization method, J. Struct. Div., Am. Soc. Civ. Eng. 99 (1) (1973) 19–31.
[121] M. Box, A new method of constrained optimization and a comparison with other methods, Comput. J. 8 (1) (1965) 42–52.
[122] N. Durand, J.-M. Alliot, A combined Nelder-Mead simplex and genetic algorithm, in: GECCO99: Proc. Genetic and Evol. Comp. Conf, Citeseer, 1999, pp. 1–7.
[123] A.-R. Hedar, M. Fukushima, Hybrid simulated annealing and direct search method for nonlinear unconstrained global optimization, Optim. Methods Softw. 17 (5) (2002) 891–912.
[124] R. Chelouah, P. Siarry, Genetic and Nelder–Mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions, Eur. J. Oper. Res. 148 (2) (2003) 335–348.
[125] R. Chelouah, P. Siarry, A hybrid method combining continuous Tabu search and Nelder– Mead simplex algorithms for the global optimization of multiminima functions, Eur. J. Oper. Res. 161 (3) (2005) 636–654.
[126] F. Wang, Y. Qiu, Empirical study of hybrid particle swarm optimizers with the simplex method operator, in: 5th International Conference on Intelligent Systems Design and Applications, IEEE, 2005, pp. 308–313.
[127] S.-K.S. Fan, Y.-C. Liang, E. Zahara, A genetic algorithm and a particle swarm optimizer hybridized with Nelder–Mead simplex search, Comput. Ind. Eng. 50 (4) (2006) 401–425.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[128] S.-K.S. Fan, E. Zahara, A hybrid simplex search and particle swarm optimization for unconstrained optimization, Eur. J. Oper. Res. 181 (2) (2007) 527–548.
[129] C. Luo, B. Yu, Low dimensional simplex evolution: a hybrid heuristic for global optimization, in: Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007), 2, IEEE, 2007, pp. 470–474.
[130] A.F. Ali, M.A. Tawhid, A hybrid cuckoo search algorithm with Nelder–Mead method for solving global optimization problems, Springerplus 5 (1) (2016) 1–22.
[131] F. Barzinpour, R. Noorossana, S.T.A. Niaki, M.J. Ershadi, A hybrid Nelder–Mead simplex and PSO approach on economic and economic-statistical designs of MEWMA control charts, Int. J. Adv. Manuf. Technol. 65 (9-12) (2013) 1339–1348.
[132] M.F. Cardoso, R.L. Salcedo, S.F. De Azevedo, The simplex-simulated annealing approach to continuous non-linear optimization, Comput. Chem. Eng. 20 (9) (1996) 1065–1080.
[133] E. Zahara, Y.-T. Kao, Hybrid Nelder–Mead simplex search and particle swarm optimization for constrained engineering design problems, Expert Syst. Appl. 36 (2) (2009) 3880–3886.
[134] A. Menchaca-Mendez, C.A.C. Coello, A new proposal to hybridize the Nelder-Mead method to a differential evolution algorithm for constrained optimization, in: 2009 IEEE Congress on Evolutionary Computation, IEEE, 2009, pp. 2598–2605.
[135] A.L. Custódio, L.N. Vicente, Using sampling and simplex derivatives in pattern search methods, SIAM J. Optim. 18 (2) (2007) 537–555.
[136] S. Dedoncker, W. Desmet, F. Naets, Generating set search using simplex gradients for bound-constrained black-box optimization, Comput. Optim. Appl. 79 (1) (2021) 35–65.
[137] A.I.F. Vaz, L.N. Vicente, A particle swarm pattern search method for bound constrained global optimization, J. Global Optim. 39 (2) (2007) 197–219.
[138] C. Audet, V. Béchard, S.Le Digabel, Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search, J. Global Optim. 41 (2) (2008) 299–318.
[139] J.H. Lee, J.-W. Kim, J.-Y. Song, Y.-J. Kim, S.-Y. Jung, A novel memetic algorithm using modified particle swarm optimization and mesh adaptive direct search for PMSM design, IEEE Trans. Magn. 52 (3) (2015) 1–4.
[140] H. Chen, Q. Feng, X. Zhang, S. Wang, Z. Ma, W. Zhou, C. Liu, A meta-optimized hybrid global and local algorithm for well placement optimization, Comput. Chem. Eng. 117 (2018) 209–220.
[141] A.R. Conn, N.I. Gould, P.L. Toint, Trust Region Methods, SIAM, 2000.
[142] Y.-x. Yuan, A review of trust region algorithms for optimization, in: Proceedings of the 4th International Congress on Industrial & Applied Mathematics (ICIAM 99), Edinburgh, 2000, pp. 271–282.
[143] Y.-x. Yuan, Recent advances in trust region algorithms, Math. Program. 151 (1) (2015) 249–281.
[144] M.J. Powell, A direct search optimization method that models the objective and constraint functions by linear interpolation, Advances in Optimization and Numerical Analysis, Springer, 1994, pp. 51–67.
[145] M.J. Powell, UOBYQA: unconstrained optimization by quadratic approximation, Math. Program. 92 (3) (2002) 555–582.
[146] M.J. Powell, The NEWUOA software for unconstrained optimization without derivatives, in: Large-Scale Nonlinear Optimization, Springer, 2006, pp. 255–297.
[147] M.J. Powell, The BOBYQA Algorithm for Bound Constrained Optimization Without Derivatives, Cambridge NA Report NA2009/06, University of Cambridge, Cambridge, 2009, pp. 26–46.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[148] S.M. Wild, R.G. Regis, C.A. Shoemaker, ORBIT: optimization by radial basis function interpolation in trust-regions, SIAM J. Sci. Comput. 30 (6) (2008) 3197–3219.
[149] R.G. Regis, S.M. Wild, CONORBIT: constrained optimization by radial basis function interpolation in trust regions, Optim. Methods Softw. 32 (3) (2017) 552–580.
[150] E.A. Gumma, M. Hashim, M.M. Ali, A derivative-free algorithm for linearly constrained optimization problems, Comput. Optim. Appl. 57 (3) (2014) 599–621.
[151] M. Aroux´et, N. Echebest, E.A. Pilotta, et al., Active-set strategy in Powell's method for optimization without derivatives, Comput. Appl. Math. 30 (2011) 171–196.
[152] P. Conejo, E.W. Karas, L.G. Pedroso, A trust-region derivative-free algorithm for constrained optimization, Optim. Methods Softw. 30 (6) (2015) 1126–1145.
[153] E. Newby, M.M. Ali, A trust-region-based derivative free algorithm for mixed integer programming, Comput. Optim. Appl. 60 (1) (2015) 199–229.
[154] A. Conn, K. Scheinberg, P. Toint, A derivative free optimization algorithm in practice, in: 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 1998, p. 4718.
[155] N.I. Gould, C. Sainvitu, P.L. Toint, A filter-trust-region method for unconstrained optimization, SIAM J. Optim. 16 (2) (2005) 341–357.
[156] C. Audet, A.R. Conn, S.Le Digabel, M. Peyrega, A progressive barrier derivative-free trust-region algorithm for constrained optimization, Comput. Optim. Appl. 71 (2) (2018) 307–329.
[157] N. Echebest, M.L. Schuverdt, R.P. Vignau, An inexact restoration derivative-free filter method for nonlinear programming, Comput. Appl. Math. 36 (1) (2017) 693–718.
[158] I. Bajaj, S.S. Iyer, M.F. Hasan, A trust region-based two phase algorithm for constrained black-box and grey-box optimization with infeasible initial point, Comput. Chem. Eng. 116 (2018) 306–321.
[159] P.R. Sampaio, P.L. Toint, A derivative-free trust-funnel method for equality constrained nonlinear optimization, Comput. Optim. Appl. 61 (1) (2015) 25–49.
[160] J.P. Eason, L.T. Biegler, A trust region filter method for glass box/black box optimization, AlChE J. 62 (9) (2016) 3124–3136.
[161] J.P. Eason, L.T. Biegler, Advanced trust region optimization strategies for glass box/black box models, AlChE J. 64 (11) (2018) 3934–3943.
[162] F. Augustin, Y.M. Marzouk, NOWPAC: a provably convergent derivative-free nonlinear optimizer with path-augmented constraints, arXiv preprint arXiv:1403.1931.
[163] P. Conejo, E.W. Karas, L.G. Pedroso, A.A. Ribeiro, M. Sachine, Global convergence of trust-region algorithms for convex constrained minimization without derivatives, Appl. Math. Comput. 220 (2013) 324–330.
[164] F.A. Fernandes, Optimization of Fischer-Tropsch synthesis using neural networks, Chem. Eng. Technol. 29 (4) (2006) 449–453.
[165] C.A. Henao, C.T. Maravelias, Surrogate-based process synthesis, Comput. Aided Chem. Eng. 28 (2010) 1129–1134.
[166] C.A. Henao, C.T. Maravelias, Surrogate-based superstructure optimization framework, AlChE J. 57 (5) (2011) 1216–1232.
[167] S.C Fahmi, Process synthesis of biodiesel production plant using artificial neural networks as the surrogate models, Comput. Chem. Eng. 46 (2012) 105–123.
[168] A. Nuchitprasittichai, S. Cremaschi, An algorithm to determine sample sizes for optimization with artificial neural networks, AlChE J. 59 (3) (2013) 805–812.
[169] J. Eason, S. Cremaschi, Adaptive sequential sampling for surrogate model generation with artificial neural networks, Comput. Chem. Eng. 68 (2014) 220–232.
[170] D.R. Jones, M. Schonlau, W.J. Welch, Efficient global optimization of expensive black-box functions, J. Global Optim. 13 (4) (1998) 455–492.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[171] J. Knowles, ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems, IEEE Trans. Evol. Comput. 10 (1) (2006) 50–66.
[172] A. Basudhar, C. Dribusch, S. Lacaze, S. Missoum, Constrained efficient global optimization with support vector machines, Struct. Multidiscip. Optim. 46 (2) (2012) 201–221.
[173] J. Qian, Y. Cheng, J. Zhang, J. Liu, D. Zhan, A parallel constrained efficient global optimization algorithm for expensive constrained optimization problems, Eng. Optim. 53 (2) (2021) 300–320.
[174] D. Huang, T.T. Allen, W.I. Notz, N. Zeng, Global optimization of stochastic black-box systems via sequential kriging meta-models, J. Global Optim. 34 (3) (2006) 441–466.
[175] M.J. Sasena, P. Papalambros, P. Goovaerts, Exploration of metamodeling sampling criteria for constrained global optimization, Eng. Optim. 34 (3) (2002) 263–278.
[176] F.A. Viana, R.T. Haftka, L.T. Watson, Efficient global optimization algorithm assisted by multiple surrogate techniques, J. Global Optim. 56 (2) (2013) 669–689.
[177] K. Hamza, M. Shalaby, A framework for parallelized efficient global optimization with application to vehicle crashworthiness optimization, Eng. Optim. 46 (9) (2014) 1200–1221.
[178] J.P. Kleijnen, W. Van Beers, I. Van Nieuwenhuyse, Expected improvement in efficient global optimization through bootstrapped kriging, J. Global Optim. 54 (1) (2012) 59–73.
[179] J.A. Caballero, I.E. Grossmann, An algorithm for the use of surrogate models in modular flowsheet optimization, AlChE J. 54 (10) (2008) 2633–2650.
[180] E. Davis, M. Ierapetritou, A kriging method for the solution of nonlinear programs with black-box functions, AlChE J. 53 (8) (2007) 2001–2012.
[181] F. Boukouvala, M.G. Ierapetritou, Derivative-free optimization for expensive constrained problems using a novel expected improvement objective function, AlChE J. 60 (7) (2014) 2462–2474.
[182] R.R. Carpio, R.C. Giordano, A.R. Secchi, Enhanced surrogate assisted framework for constrained global optimization of expensive black-box functions, Comput. Chem. Eng. 118 (2018) 91–102.
[183] J.A. Paulson, C. Lu, Cobalt: COnstrained Bayesian optimizAtion of computationaLly expensive grey-box models exploiting derivaTive information, arXiv preprint arXiv:2105.04114.
[184] A.M. Schweidtmann, D. Bongartz, D. Grothe, T. Kerkenhoff, X. Lin, J. Najman, A. Mitsos, Deterministic global optimization with Gaussian processes embedded, Math. Program. Comput. 3 (2021) 553–581.
[185] H.-M. Gutmann, A radial basis function method for global optimization, J. Global Optim. 19 (3) (2001) 201–227.
[186] M.J. Powell, The theory of radial basis function approximation in 1990, Adv. Numer. Anal. (1992) 105–210.
[187] M. Björkman, K. Holmström, Global optimization of costly nonconvex functions using radial basis functions, Optim. Eng. 1 (4) (2000) 373–397.
[188] R.G. Regis, C.A. Shoemaker, Constrained global optimization of expensive black box functions using radial basis functions, J. Global Optim. 31 (1) (2005) 153–171.
[189] S. Jakobsson, M. Patriksson, J. Rudholm, A. Wojciechowski, A method for simulation based optimization using radial basis functions, Optim. Eng. 11 (4) (2010) 501–532.
[190] R.G. Regis, C.A. Shoemaker, Improved strategies for radial basis function methods for global optimization, J. Global Optim. 37 (1) (2007) 113–135.
[191] R.G. Regis, C.A. Shoemaker, Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization, Eng. Optim. 45 (5) (2013) 529–555.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[192] S.S. Garud, N. Mariappan, I.A. Karimi, Surrogate-based black-box optimisation via domain exploration and smart placement, Comput. Chem. Eng. 130 (2019) 106567.
[193] K. Palmer, M. Realff, Metamodeling approach to optimization of steady-state flowsheet simulations: model generation, Chem. Eng. Res. Des. 80 (7) (2002) 760–772.
[194] K. Palmer, M. Realff, Optimization and validation of steady-state flowsheet simulation metamodels, Chem. Eng. Res. Des. 80 (7) (2002) 773–782.
[195] K.S. Won, T. Ray, A framework for design optimization using surrogates, Eng. Optim. 37 (7) (2005) 685–703.
[196] T. Goel, R.T. Haftka, W. Shyy, N.V. Queipo, Ensemble of surrogates, Struct. Multidiscip. Optim. 33 (3) (2007) 199–216.
[197] J. Müller, R. Piché, Mixture surrogate models based on Dempster-Shafer theory for global optimization problems, J. Global Optim. 51 (1) (2011) 79–104.
[198] A. Cozad, N.V. Sahinidis, D.C. Miller, Learning surrogate models for simulation-based optimization, AlChE J. 60 (6) (2014) 2211–2227.
[199] C.A. Kieslich, F. Boukouvala, C.A. Floudas, Optimization of black-box problems using Smolyak grids and polynomial approximations, J. Global Optim. 71 (4) (2018) 845–869.
[200] A.M. Schweidtmann, A. Mitsos, Deterministic global optimization with artificial neural networks embedded, J. Optim. Theory Appl. 180 (3) (2019) 925–948.
[201] C.A.O. Nascimento, R. Giudici, R. Guardani, Neural network based approach for optimization of industrial chemical processes, Comput. Chem. Eng. 24 (9-10) (2000) 2303–2314.
[202] J. Katz, I. Pappas, S. Avraamidou, E.N. Pistikopoulos, Integrating deep learning models and multiparametric programming, Comput. Chem. Eng. 136 (2020) 106801.
[203] D. Kenefake, I. Pappas, S. Avraamidou, B. Beykal, H.S. Ganesh, Y. Cao, Y. Wang, J. Otashu, S. Leyland, J. Flores-Cerrillo, et al., A smart manufacturing strategy for multi-parametric model predictive control in air separation systems, J. Adv. Manuf. Process. (2022) e10120.
[204] F. Boukouvala, M.F. Hasan, C.A. Floudas, Global optimization of general constrained grey-box models: new method and its application to constrained PDEs for pressure swing adsorption, J. Global Optim. 67 (1-2) (2017) 3–42.
[205] F. Boukouvala, C.A. Floudas, Argonaut: algorithms for global optimization of constrained grey-box computational problems, Optim. Lett. 11 (5) (2017) 895–913.
[206] B. Beykal, F. Boukouvala, C.A. Floudas, N. Sorek, H. Zalavadia, E. Gildin, Global optimization of grey-box computational systems using surrogate functions and application to highly constrained oil-field operations, Comput. Chem. Eng. 114 (2018) 99–110.
[207] B. Beykal, M. Onel, O. Onel, E.N. Pistikopoulos, A data-driven optimization algorithm for differential algebraic equations with numerical infeasibilities, AlChE J. 66 (10) (2020) e16657.
[208] B. Beykal, Z. Aghayev, O. Onel, M. Onel, E.N. Pistikopoulos, Data-driven stochastic optimization of numerically infeasible differential algebraic equations: an application to the steam cracking process, Comput. Aided Chem. Eng. 49 (2022) 1579–1584.
[209] B. Beykal, N.A. Diangelakis, E.N. Pistikopoulos, Continuous-time surrogate models for data-driven dynamic optimization, Comput. Aided Chem. Eng. 51 (2022) 205–210.
[210] B. Beykal, F. Boukouvala, C.A. Floudas, E.N. Pistikopoulos, Optimal design of energy systems using constrained grey-box multi-objective optimization, Comput. Chem. Eng. 116 (2018) 488–502.
[211] B. Beykal, S. Avraamidou, I.P. Pistikopoulos, M. Onel, E.N. Pistikopoulos, DOMINO: data-driven optimization of bi-level mixed-integer nonlinear problems, J. Global Optim. (2020) 1–36.
[212] S. Avraamidou, B. Beykal, I.P. Pistikopoulos, E.N. Pistikopoulos, A hierarchical food-energy-water nexus (few-n) decision-making approach for land use optimization, Comput. Aided Chem. Eng. 44 (2018) 1885–1890.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[213] E. Davis, M. Ierapetritou, A kriging-based approach to MINLP containing black-box models and noise, Ind. Eng. Chem. Res. 47 (16) (2008) 6101–6125.
[214] E. Davis, M. Ierapetritou, A kriging based method for the solution of mixed-integer nonlinear programs containing black-box functions, J. Global Optim. 43 (2) (2009) 191–205.
[215] K. Holmström, N.-H. Quttineh, M.M. Edvall, An adaptive radial basis algorithm (ARBF) for expensive black-box mixed-integer constrained global optimization, Optim. Eng. 9 (4) (2008) 311–339.
[216] A. Costa, G. Nannicini, RBFOpt: an open-source library for black-box optimization with costly function evaluations, Math. Progr. Comput. 10 (4) (2018) 597–629.
[217] K. Rashid, S. Ambani, E. Cetinkaya, An adaptive multiquadric radial basis function method for expensive black-box mixed-integer nonlinear constrained optimization, Eng. Optim. 45 (2) (2013) 185–206.
[218] J. Müller, C.A. Shoemaker, R. Piché, SO-MI: a surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems, Comput. Oper. Res. 40 (5) (2013) 1383–1400.
[219] J. Müller, C.A. Shoemaker, R. Piché, SO-I: a surrogate model algorithm for expensive nonlinear integer programming problems including global optimization applications, J. Global Optim. 59 (4) (2014) 865–889.
[220] J. Müller, MISO: mixed-integer surrogate optimization framework, Optim. Eng. 17 (1) (2016) 177–203.
[221] R.G. Regis, Large-scale discrete constrained black-box optimization using radial basis functions, in: 2020 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2020, pp. 2924–2931.
[222] S.H. Kim, F. Boukouvala, Surrogate-based optimization for mixed-integer nonlinear problems, Comput. Chem. Eng. 140 (2020) 106847.
[223] A. Thebelt, J. Kronqvist, M. Mistry, R.M. Lee, N. Sudermann-Merx, R. Misener, ENTMOOT: a framework for optimization over ensemble tree models, Comput. Chem. Eng. 151 (2021) 107343.
[224] F. Ceccon, J. Jalving, J. Haddad, A. Thebelt, C. Tsay, C.D. Laird, R. Misener, OMLT: Optimization & machine learning toolkit, arXiv preprint arXiv:2202.02414.
[225] L. Bliek, A. Guijt, S. Verwer, M. De Weerdt, Black-box mixed-variable optimisation using a surrogate model that satisfies integer constraints, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2021, pp. 1851–1859.
[226] N. Ploskas, N.V. Sahinidis, Review and comparison of algorithms and software for mixed-integer derivative-free optimization, J. Global Optim. (2021) 1–30.
[227] The Optimization Firm, ALAMO modeling tool, https://minlp.com/alamo-modeling-tool.
[228] TOMLAB Optimization, TOMLAB /CGO toolbox, https://tomopt.com/tomlab/products/cgo/
[229] Z. Zhang, Software by late Professor M. J. D. Powell and PDFO, https://www.zhangzk.net/software.html.
[230] S.G. Johnson, The NLopt nonlinear-optimization package, https://nlopt.readthedocs.io/en/latest/.
[231] J. Paulson, The COBALT algorithm for constrained grey-box optimization of computationally expensive models, https://github.com/joelpaulson/COBALT.
[232] P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S.J. van der Walt, M. Brett, J. Wilson, K.J. Millman, N. Mayorov, A.R.J. Nelson, E. Jones, R. Kern, E. Larson, C.J. Carey, I˙. Polat, Y. Feng, E.W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E.A. Quintero, C.R. Harris, A.M. Archibald, A.H. Ribeiro, F. Pedregosa, P. van Mulbregt, SciPy 1.0 Contributors, SciPy 1.0: fundamental algorithms for scientific computing in Python, Nat. Methods 17 (2020) 261–272.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[233] TOMLAB Optimization, TOMLAB/glbDirect, https://tomopt.com/tomlab/products/base/solvers/glbDirect.php.
[234] M.A. Bouhlel, J.T. Hwang, N. Bartoli, R. Lafage, J. Morlier, J.R.R.A. Martins, SMT: surrogate modeling toolbox, https://smt.readthedocs.io/en/latest/_src_docs/applications/ego.html#.
[235] Computational Optimisation Group at Imperial College London, Ensemble tree model optimization tool, https://github.com/cog-imperial/entmoot.
[236] Lawrence Berkeley National Laboratory, MISO mixed integer surrogate optimization framework, https://optimization.lbl.gov/downloads#h.p_BjSaeAORU9gm.
[237] L. Bliek, MVRSM, https://github.com/lbliek/MVRSM.
[238] GERAD, NOMAD: a blackbox optimization software, https://www.gerad.ca/en/software/nomad/.
[239] F. Augustin, F. Menhorn, (S)NOWPAC: (Stochastic) nonlinear optimization with path-augmented constraints, https://github.com/snowpac/snowpac.
[240] L.J.V. Miranda, PySwarms a research toolkit for particle swarm optimization (PSO) in Python, https://github.com/ljvmiranda921/pyswarms.
[241] A.I.F. Vaz, PSwarm, http://www.norg.uminho.pt/aivaz/pswarm/.
[242] NEOS Server, PSwarm, https://neos-server.org/neos/solvers/go:PSwarm/AMPL.html.
[243] L. Jiang, L.T. Biegler, V.G. Fox, Simulation and optimization of pressure-swing adsorption systems for air separation, AIChE J. 49 (5) (2003) 1140–1157.
[244] S.R.R. Vetukuri, L.T. Biegler, A. Walther, An inexact trust-region algorithm for the optimization of periodic adsorption processes, Ind. Eng. Chem. Res. 49 (23) (2010) 12004–12013.
[245] N. Zhang, P. Bénard, R. Chahine, T. Yang, J. Xiao, Optimization of pressure swing adsorption for hydrogen purification based on box-Behnken design method, Int. J. Hydrogen Energy 46 (7) (2021) 5403–5417.
[246] J. Uebbing, L.T. Biegler, L. Rihko-Struckmann, S. Sager, K. Sundmacher, Optimization of pressure swing adsorption via a trust-region filter algorithm and equilibrium theory, Comput. Chem. Eng. 151 (2021) 107340.
[247] L.E. Andersson, J. Schilling, L. Riboldi, A. Bardow, R. Anantharaman, Bayesian optimization for techno-economic analysis of pressure swing adsorption processes, Comput. Aided Chem. Eng. 51 (2022) 1441–1446.
[248] F. Boukouvala, M.G. Ierapetritou, Surrogate-based optimization of expensive flowsheet modeling for continuous pharmaceutical manufacturing, J. Pharm. Innov. 8 (2) (2013) 131–145.
[249] Z. Wang, M. Ierapetritou, A novel surrogate-based optimization method for black-box simulation with heteroscedastic noise, Ind. Eng. Chem. Res. 56 (38) (2017) 10720–10732.
[250] Z. Wang, M. Ierapetritou, Constrained optimization of black-box stochastic systems using a novel feasibility enhanced kriging-based method, Comput. Chem. Eng. 118 (2018) 210–223.
[251] J.-X. Zhang, Y.-Z. Chen, Z.-N. Wu, W.-R. Liao, Optimize the preparation process of erigeron breviscapus sustained-release pellets based on artificial neural network and particle swarm optimization algorithm, Zhong Yao Cai 35 (1) (2012) 127–133.
[252] D.E. Ciaurri, T. Mukerji, L.J. Durlofsky, Derivative-free optimization for oil field operations, Computational Optimization and Applications in Engineering and Industry, Springer, 2011, pp. 19–55.
[253] O.J. Isebor, D.E. Ciaurri, L.J. Durlofsky, Generalized field-development optimization with derivative-free procedures, SPE J. 19 (05) (2014) 891–908.
[254] D. Janiga, R. Czarnota, J. Stopa, P. Wojnarowski, P. Kosowski, Performance of nature inspired optimization algorithms for polymer enhanced oil recovery process, J. Pet. Sci. Eng. 154 (2017) 354–366.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[255] M. Siavashi, M.H. Doranehgard, Particle swarm optimization of thermal enhanced oil recovery from oilfields with temperature control, Appl. Therm. Eng. 123 (2017) 658–669.
[256] N.V. Queipo, J.V. Goicochea, S. Pintos, Surrogate modeling-based optimization of SAGD processes, J. Pet. Sci. Eng. 35 (1-2) (2002) 83–93.
[257] B. Horowitz, L.J. do Nascimento Guimaraes, V. Dantas, S.M.B. Afonso, A concurrent efficient global optimization algorithm applied to polymer injection strategies, J. Pet. Sci. Eng. 71 (3-4) (2010) 195–204.
[258] B. Beykal, Advances in Data-Driven Modeling and Global Optimization of Constrained Grey-Box Computational Systems Ph.D. Thesis, Texas A&M University, 2020.
[259] J.F. Granjo, D.S. Nunes, B.P. Duarte, N.M. Oliveira, A comparison of process alternatives for energy-efficient bioethanol downstream processing, Sep. Purif. Technol. 238 (2020) 116414.
[260] W. Zhong, C. Qiao, X. Peng, Z. Li, C. Fan, F. Qian, Operation optimization of hydrocracking process based on kriging surrogate model, Control Eng. Pract. 85 (2019) 34–40.
[261] E.G. Pardo, J. Blanco-Linares, D. Vel´azquez, F. Serradilla, Optimization of a steam reforming plant modeled with artificial neural networks, Electronics 9 (11) (2020) 1923.
[262] J. Straus, J.A. Ouassou, B.R. Knudsen, R. Anantharaman, Constrained adaptive sampling for domain reduction in surrogate model generation: applications to hydrogen production, AlChE J. 67 (11) (2021) e17357.
[263] Y. Cao, J. Flores-Cerrillo, C.L. Swartz, Practical optimization for cost reduction of a liquefier in an industrial air separation plant, Comput. Chem. Eng. 99 (2017) 13–20.
[264] B. Wang, S. Shi, S. Wang, L. Qiu, X. Zhang, Optimal design for cryogenic structured packing column using particle swarm optimization algorithm, Cryogenics 103 (2019) 102976.
[265] J. Jerald, P. Asokan, G. Prabaharan, R. Saravanan, Scheduling optimisation of flexible manufacturing systems using particle swarm optimisation algorithm, Int. J. Adv. Manuf. Technol. 25 (9) (2005) 964–971.
[266] P. Hansen, B. Jaumard, G. Savard, New branch-and-bound rules for linear bilevel programming, SIAM J. 13 (5) (1992) 1194–1217.
[267] V. Visweswaran, C. Floudas, M. Ierapetritou, E. Pistikopoulos, A decomposition-based global optimization approach for solving bilevel linear and quadratic programs, State of the Art in Global Optimization, Springer, 1996, pp. 139–162.
[268] H. Tamaki, H. Kita, S. Kobayashi, Multi-objective optimization by genetic algorithms: a review, in: Proceedings of IEEE international conference on evolutionary computation, IEEE, 1996, pp. 517–522.
[269] C.A. Coello Coello, A comprehensive survey of evolutionary-based multiobjective optimization techniques, Knowl. Inf. Syst. 1 (3) (1999) 269–308.
[270] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, Wiley, 2001.
[271] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.
[272] C.A.C. Coello, G.B. Lamont, D.A. Van Veldhuizen, et al., Evolutionary Algorithms for Solving Multi-Objective Problems, 5, Springer, 2007.
[273] L. Wang, A.H. Ng, K. Deb, Multi-Objective Evolutionary Optimisation for Product Design and Manufacturing, Springer, 2011.
[274] G.P. Rangaiah, A. Bonilla-Petriciolet, Multi-Objective Optimization in Chemical Engineering: Developments and Applications, John Wiley & Sons, 2013.
[275] G.P. Rangaiah, Z. Feng, A.F. Hoadley, Multi-objective optimization applications in chemical process engineering: tutorial and review, Processes 8 (5) (2020) 508.
[276] G. Fiandaca, E.S. Fraga, S. Brandani, A multi-objective genetic algorithm for the design of pressure swing adsorption, Eng. Optim. 41 (9) (2009) 833–854.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[277] J. Beck, D. Friedrich, S. Brandani, E.S. Fraga, Multi-objective optimisation using surrogate models for the design of VPSA systems, Comput. Chem. Eng. 82 (2015) 318–329.
[278] S.G. Subraveti, Z. Li, V. Prasad, A. Rajendran, Machine learning-based multiobjective optimization of pressure swing adsorption, Ind. Eng. Chem. Res. 58 (44) (2019) 20412–20422.
[279] K. Alkebsi, W. Du, Surrogate-assisted multi-objective particle swarm optimization for the operation of CO_2 capture using VPSA, Energy 224 (2021) 120078.
[280] L. Tong, P. B´enard, Y. Zong, R. Chahine, K. Liu, J. Xiao, Artificial neural network based optimization of a six-step two-bed pressure swing adsorption system for hydrogen purification, Energy AI 5 (2021) 100075.
[281] X. Yu, Y. Shen, Z. Guan, D. Zhang, Z. Tang, W. Li, Multi-objective optimization of ANN-based PSA model for hydrogen purification from steam-methane reforming gas, Int. J. Hydrogen Energy 46 (21) (2021) 11740–11755.
[282] B.V. Piguave, S.D. Salas, D. De Cecchis, J.A. Romagnoli, Modular framework for simulation-based multi-objective optimization of a cryogenic air separation unit, ACS Omega 7 (14) (2022) 11696–11709.
[283] A. Tarafder, B.C. Lee, A.K. Ray, G. Rangaiah, Multiobjective optimization of an industrial ethylene reactor using a nondominated sorting genetic algorithm, Ind. Eng. Chem. Res. 44 (1) (2005) 124–141.
[284] G. Golkarnarenji, M. Naebe, K. Badii, A.S. Milani, A. Jamali, A. Bab-Hadiashar, R.N. Jazar, H. Khayyam, Multi-objective optimization of manufacturing process in carbon fiber industry using artificial intelligence techniques, IEEE Access 7 (2019) 67576–67588.
[285] K.N. Pai, V. Prasad, A. Rajendran, Generalized, adsorbent-agnostic, artificial neural network framework for rapid simulation, optimization, and adsorbent screening of adsorption processes, Ind. Eng. Chem. Res. 59 (38) (2020) 16730–16740.
[286] J. Lu, Q. Wang, Z. Zhang, J. Tang, M. Cui, X. Chen, Q. Liu, Z. Fei, X. Qiao, Surrogate modeling-based multi-objective optimization for the integrated distillation processes, Chem. Eng. Process. 159 (2021) 108224.
[287] A. Sinha, P. Malo, K. Deb, A review on bilevel optimization: from classical to evolutionary approaches and applications, IEEE Trans. Evol. Comput. 22 (2) (2017) 276–295.
[288] G. Liu, J. Han, S. Wang, A trust region algorithm for bilevel programing problems, Chin. Sci. Bull. 43 (10) (1998) 820–824.
[289] B. Colson, P. Marcotte, G. Savard, A trust-region method for nonlinear bilevel programming: algorithm and computational experience, Comput. Optim. Appl. 30 (3) (2005) 211–227.
[290] S.D. Handoko, L.H. Chuin, A. Gupta, O.Y. Soon, H.C. Kim, T.P. Siew, Solving multi-vehicle profitable tour problem via knowledge adoption in evolutionary bi-level programming, in: 2015 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2015, pp. 2713–2720.
[291] I. Nishizaki, M. Sakawa, Computational methods through genetic algorithms for obtaining Stackelberg solutions to two-level integer programming problems, Cybern. Syst. 36 (6) (2005) 565–579.
[292] L. Hecheng, W. Yuping, Exponential distribution-based genetic algorithm for solving mixed-integer bilevel programming problems, J. Syst. Eng. Electron. 19 (6) (2008) 1157–1164.
[293] J.M. Arroyo, F.J. Fern´andez, A genetic algorithm approach for the analysis of electric grid interdiction with line switching, in: 2009 15th International Conference on Intelligent System Applications to Power Systems, IEEE, 2009, pp. 1–6.
[294] B. Beykal, S. Avraamidou, E.N. Pistikopoulos, Bi-level mixed-integer data-driven optimization of integrated planning and scheduling problems, Comput. Aided Chem. Eng. 50 (2021) 1707–1713.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[295] B. Beykal, S. Avraamidou, E.N. Pistikopoulos, Data-driven optimization of mixed-integer bi-level multi-follower integrated planning and scheduling problems under demand uncertainty, Comput. Chem. Eng. 156 (2022) 107551.
[296] A. Sinha, P. Korhonen, J. Wallenius, K. Deb, An interactive evolutionary multi-objective optimization algorithm with a limited number of decision maker calls, Eur. J. Oper. Res. 233 (3) (2014) 674–688.
[297] T.T. Binh, U. Korn, MOBES: a multiobjective evolution strategy for constrained optimization problems, in: The Third International Conference on Genetic Algorithms (Mendel 97), 25, Citeseer, 1997, p. 27.
[298] I. Pappas, S. Avraamidou, J. Katz, B. Burnak, B. Beykal, M. Turkay, E.N. Pistikopoulos, Multiobjective optimization of mixed-integer linear programming problems: a multi-parametric optimization approach, Ind. Eng. Chem. Res. 60 (23) (2021) 8493–8503.
[299] K.H. Sahin, A.R. Ciric, A dual temperature simulated annealing approach for solving bilevel programming problems, Comput. Chem. Eng. 23 (1) (1998) 11–25.
[300] A. Mitsos, Global solution of nonlinear mixed-integer bilevel programs, J. Global Optim. 47 (4) (2010) 557–582.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

CHAPTER SIX

Machine learning for control of (bio)chemical manufacturing systems

Andreas Himmel[a], Janine Matschek[a], Rudolph Kok (Louis)[b], Bruno Morabito[b], Hoang Hai Nguyen[a] and Rolf Findeisen[a]
[a]Technical University of Darmstadt, Control and Cyber-Physical Systems, Darmstadt, Germany
[b]Otto von Guericke University Magdeburg, Systems Theory and Automatic Control, Magdeburg, Germany

6.1 Introduction

The chemical and biochemical industry is one of the critical and core elements in the manufacturing sector, covering processes for the production of batteries, biochemicals, chemicals, fertilizers, food and beverages, petrochemicals, pharmaceuticals, etc. [1–4]. Control, automation, and monitoring of the involved physical, chemical, and biochemical processes are crucial to obtaining high-quality products, ensuring safety, meeting regulation requirements, providing flexibility to adjust to changes, and achieving and improving economics and sustainability. However, flexible, yet optimal control and monitoring of some of the processes are challenging [5]. Most biochemical and chemical processes are complex, multiscale, high-dimensional systems that exhibit large nonlinearity and are affected by uncertainties [4,6,7]. The control strategy must take all these aspects into account and often needs to achieve a balance between different requirements. One way to consider these effects is the use of (detailed) mathematical models that describe the plant dynamics and to employ model-based control and optimization [8,9] techniques. These models are traditionally built based on first-principle knowledge, such as thermodynamics, energy, and mass conservation laws, which can be challenging, expensive, or not possible at all.

Recently, significant progress in the field of artificial intelligence and machine learning has been achieved, driven by the ever-increasing amount of data and by breakthroughs in areas such as deep learning, allowing the applications of machine learning models. Machine learning is successfully applied in many fields, such as robotics [10], autonomous driving [11], medicine [221], or transportation [13], to name just a few. While machine

Artificial Intelligence in Manufacturing: Concepts and Methods.
DOI: https://doi.org/10.1016/B978-0-323-99134-6.00009-8
Copyright © 2024 Elsevier Inc.
All rights are reserved, including those for text and data mining, AI training, and similar technologies.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY
Copyright Elsevier 2024

learning is used in chemical and biochemical engineering and process systems engineering [14–16], for example, for process and materials discovery [17,18], and process data analytics [19,20], the application of machine learning to (bio)chemical processes control is yet limited and often challenging [15,19]. Safety requirements and lack of reliable data are some of the most critical bottlenecks for machine learning applications in the field of control, especially for (bio)chemical process operation. Historical data collected do not represent the current plant operation due to, for example, design changes or modifications, equipment degradation due to fouling or aging, changes in the operating regimes, etc. Hence, experiments and instrumentation must be carried out for new data acquisition. Since the dynamics of these processes are often slower than, for example, the dynamics of robots, these experiments require a considerable amount of time, which can, in turn, cause a loss of production time. These losses might not be affordable, especially for the production of bulk chemicals, where plants run with tight economic margins. However, new measurement technologies, like spectroscopic methods, are becoming available, which make easy-to-use inline measurements possible. Furthermore, machine learning algorithms are developed that can deal with limited amounts of data. Approaches like physics-informed learning [21] and hybrid modeling [22] allow to incorporate of prior knowledge to enhance the extrapolation capabilities of models in regions with little or no data.

When data are available, they almost always contain some uncertainty, which has to be taken into account. Additionally, malfunctioning of a plant, during the experiment or during production, can have severe repercussions in terms of safety and product quality. To gain the trust of the operators, machine learning models must be equipped with some degree of robustness and safety guarantees and the control algorithms should exploit those [23,24]. Hence, machine learning algorithms should be developed to support the control of (bio)chemical plants such that control performance can be enhanced while guaranteeing satisfaction of all safety regulations. Often, various control strategies are applied on multiple hierarchical levels, spanning from classical PID controllers to optimization-based control such as model predictive control. Each of these levels interacts with the process at a different level of granularity, hence deals with different timescales and parts of the plant. Machine learning applications should be able to integrate with such hierarchical structure.

In this chapter, we want to outline how machine learning can address the mentioned challenges and support the control and controller design specifically for the (bio)chemical industry. Fig. 6.1 depicts possibilities how

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

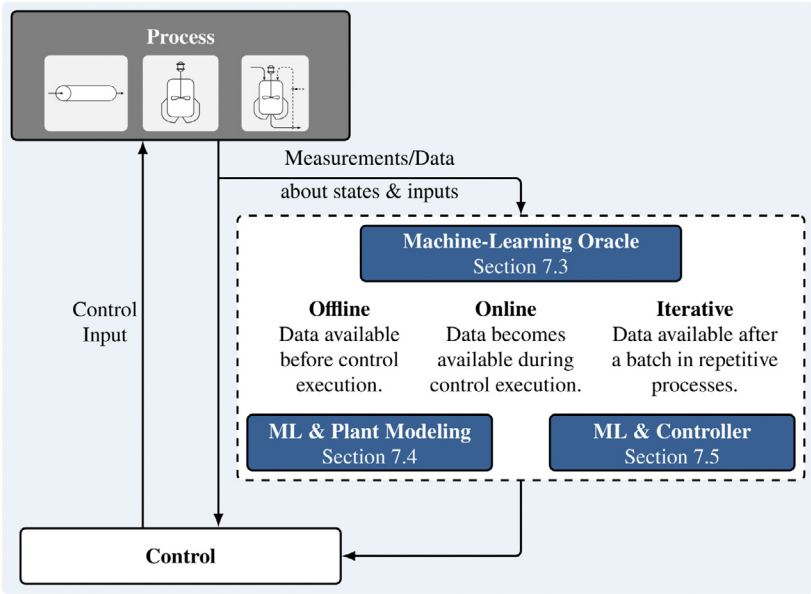


Figure 6.1 This book chapter outlines how machine learning can be used to support the control of (bio)chemical manufacturing processes.

machine learning can be used to do so. Basically, the machine learning–Oracle (ML–Oracle) will use data from the plant to design or adapt models or parts of the control algorithm. The learned relations will be provided to the controller to improve performance, robustness, or safety. The generation of the machine learning components can be performed offline (before control execution), online (during controller runtime), or iteratively (in between controller executions). In terms on machine learning algorithms, we focus especially on neural networks and Gaussian processes to represent commonly used machine learning techniques [1,2,25]. We present a generic framework how machine learning can be embedded in process control, operation and monitoring, allowing for other machine learning methods. Mainly, the support of control technology by machine learning will be addressed in this chapter from two perspectives (see also Fig. 6.1):

1. Machine learning can help to derive models of the (bio)chemical plant, but also for the cost and constraints, for analysis, simulation-based controller design, as well as model-based estimation and control.
2. Machine learning can be used to learn or replace controllers and control laws directly from data.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

The first item is related to *machine learning-supported control* via data-based or hybrid representation of the system dynamics, state and parameter estimation, or monitoring. Generic reviews on how machine learning can be used for modeling of (bio)chemical plants can be found in references [22,26,27]. In contrast, we focus on modeling which is explicitly done to perform some notion of control or controller design, that is, learning-supported simulation-, optimization-, and model-based controller design. The second item relates to *learning controllers*, that is, controller parametrization, controller approximation, learning-based decision-making, adaptive control, reinforcement learning, as well as imitation learning. Here, the controller is learned/approximated via machine learning techniques.

With respect to control strategies used in combination with machine learning, we put special focus on model predictive control techniques [28–30], which have a high potential to improve control operation in combination with machine learning techniques [23,24,31].

Our main contribution is to present a structured overview how different machine learning methods can be applied to support the control of (bio)chemical manufacturing processes. To do so, we outline a generic framework which covers the most common tasks and algorithms of machine learning in control. Based on this framework, we present a vast literature review to describe the usage of machine learning in manufacturing processes.

The remainder of this chapter is structured as follows: First, we outline what the classes of chemical processes considered in Section 6.2 from which we derive tasks for machine learning to support the control of these processes. These tasks span from modeling and estimation for monitoring and control, learning disturbances and reference signals, to a data-based controller design. Section 6.3 presents the mathematical notation of the machine leaning components in an abstract/generic form and describes the training procedure of these components via a generic ML–Oracle. Based on this generic description, Sections 6.4 and 6.5 give an overview of the applications of machine learning for the identified subtasks, see also Fig. 6.1. Section 6.4 focuses on learning-based modeling of the dynamics of (bio)process plants. It also describes how these models can be used for monitoring, simulation, and model-based control. Section 6.5 outlines the use of machine learning to learn a control law based on data or to approximate a difficult control law via machine learning. Here, the machine learning algorithms are directly used to represent the controller of the system instead of the dynamical system model. Section 6.6 summarizes this overview article and presents future research directions.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

6.2 (Bio)chemical processes

This section briefly discusses the modeling and operation of (bio)chemical processes, typically process operations, such as continuous or repeated batch operation, and the frequently encountered hierarchical control structures. We only touch on topics relevant to the subsequent sections. For an in-depth introduction and review to control and process systems engineering appropriate modeling and control, we refer to references [32–36].

Chemical and biochemical processes produce from input materials (feeds) products. They typically consist of different subcomponents, such as reactors, separators, and mixers. They are controlled by manipulating input variables, such as flow rates, temperatures, and pressures, while the process is monitored by measurements, such as concentrations, temperatures, and humidity.

6.2.1 Plant modeling

Plant dynamics are often described by suitable mathematical models. These models provide insight, enable improved plant design, and are crucial for process optimization, control, and monitoring [32–34]. Process models can be obtained from first principles based on fundamental engineering, physical, and chemical principles and/or experimental data [22,32,37]. Deriving suitable models is often challenging and might involve many states, unknown parameters, inputs/manipulated variables, and outputs.

For simplicity of presentation, we assume that the process or element considered does not require the use of distributed parameters and states, such as concentrations and temperatures varying over the geometry. We take a rather general model structure, where to apply machine learning techniques, we consider a hybrid composition of model components based on first-principle insight and data and learning-based components of the form:

$$\dot{\mathrm{x}}(t) = f(\mathrm{x}(t), \mathrm{z}(t), \mathrm{u}(t), p) + \mathrm{F}(\mathrm{x}(t), \mathrm{z}(t), p) \tag{6.1a}$$

$$0 = g(\mathrm{x}(t), \mathrm{z}(t), \mathrm{u}(t), p) + \mathrm{G}(\mathrm{x}(t), \mathrm{z}(t), p) \tag{6.1b}$$

$$\mathrm{y}(t) = h(x(t), \mathrm{z}(t)). \tag{6.1c}$$

Here, $\mathrm{x}(t) \in \mathcal{X} \subseteq \mathbb{R}^{n_\mathrm{x}}$ and $\mathrm{z}(t) \in \mathcal{Z} \subseteq \mathbb{R}^{n_\mathrm{z}}$ are dynamical and algebraic states, $\mathrm{u}(t) \in \mathcal{U} \subseteq \mathbb{R}^{n_\mathrm{u}}$ denotes the manipulating variables/inputs, and $p \in \mathcal{P} \subseteq \mathbb{R}^{n_\mathrm{p}}$ represents model parameters. The variable $\mathrm{y}(t) \in \mathcal{Y} \subseteq \mathbb{R}^{n_y}$ is the measurement/output of the process $h : \mathcal{X} \times \mathcal{Z} \to \mathcal{Y}$. The dynamics of the

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

process is described by a first-principle component $f : \mathcal{X} \times \mathcal{Z} \times \mathcal{U} \times \mathcal{P} \to \mathbb{R}^{n_x}$ and a (learned) component based on data $\mathrm{F} : \mathcal{X} \times \mathcal{Z} \times \mathcal{P} \to \mathbb{R}^{n_x}$. Similarly, $g : \mathcal{X} \times \mathcal{Z} \times \mathcal{U} \times \mathcal{P} \to \mathbb{R}^{n_z}$ and $\mathrm{G} : \mathcal{X} \times \mathcal{Z} \times \mathcal{P} \to \mathbb{R}^{n_z}$ describe the coupling between the algebraic states and other variables, where we assume that g is based on physical considerations, while G might be learned based on measurements.

Note that this choice of the model structure is somewhat arbitrary and is taken to stress how machine learning components can be integrated into the modeling, control, and monitoring parts.

Remark 6.2.1. In many cases, Eq. (6.1b) is given explicitly by:

$$\mathrm{z}(t) = g_{\mathrm{ex}}(\mathrm{x}(t), \mathrm{u}(t), p) + \mathrm{G}_{\mathrm{ex}}(\mathrm{x}(t), p)$$

which allows simplifying the model Eqs. (6.1) into an ordinary differential equation.

Often the system behavior is represented in discrete time, leading to a discrete-time model. The discrete-time model is then given by integrating Eqs. (6.1a) and (6.1b), for example, assuming a piece-wise constant input (i.e., $\mathrm{u}(t) = u_k$, $t \in [t_k, t_{k+1})$) and the sampling time $T := t_{k+1} - t_k$. We assume an equidistant time grid. Note that the further explanations can also be extended to variable grid sizes. The resulting model equations are given by

$$(\mathrm{x}(t_{k+1}), \mathrm{z}(t_{k+1})) = I\big(\mathrm{x}(t_k), \mathrm{u}(t_k), p\big) \tag{6.2a}$$

$$\mathrm{y}(t_k) = h(x(t_k), \mathrm{z}(t_k)) \tag{6.2b}$$

where, $I : \mathcal{X} \times \mathcal{U} \times \mathcal{P} \to \mathcal{X} \times \mathcal{Z}$ is the mapping resulting from the integration. In principle, I can be decomposed similarly to the continuous time case into learned, a first-principle part, and algebraic subcomponents.

Remark 6.2.2. In the following, we shortly use $x_k := \mathrm{x}(t_k)$, $z_k := \mathrm{z}(t_k)$ and $u_k := \mathrm{u}(t_k)$.

First-principle models

First-principle modeling is typically based on laws of physics, such as the conservation of energy or mass or thermodynamic relations. Doing so often allows for describing the physical process with a high degree of spatial and temporal resolution [38] if the process is sufficiently well understood. First-principle models can, under suitable assumptions, have a high extrapolate quality and might be applicable over a wide range of operating conditions

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

while providing physical insight [33,39]. However, the widespread use of first-principle models is hindered by several reasons. The performance of first-principle models is limited by the assumptions made [4]. Furthermore, deep process knowledge and understanding are required to develop first-principle models capable of describing the underlying relations [39]. This renders the development of process models for large-scale systems with many integrated processes challenging, expensive sometimes impractical [40]. Also, the maintenance of complex first-principle models is often not economically feasible, and a trade-off between sustainability and rigor is required [32,40]. First-principle models can also suffer from high computational complexity due to the high dimensionality, nonlinearities, and wide span of time and space scales of chemical and biochemical process dynamics [4,39,41,42].

Data-based models

Data-based models or model parts (represented F and G in Eqs. (6.1) can be used to address some of the mentioned drawbacks of first-principle models. Machine learning algorithms can describe relationships between input and output data based on general structures without knowing the process in advance [4,40]. Thus, machine learning techniques can approximate unknown phenomena and parameter effects, reducing experts' manual effort, design time, and development costs [43]. Additionally, the flexibility in production can be increased since machine learning methods are adaptable and easy to retrain on new data when process changes occur [37]. In particular, machine learning methods allow to deal with noisy data, model uncertainties, large search spaces, nonlinearities, ill-defined subproblems, or tasks that require fast and real-time feasible solutions [6,16,39]. Some techniques, such as Gaussian processes, can quantify approximation errors or model uncertainties allowing to obtain reliable models with corresponding error bounds [27,31,38,44]. Furthermore, machine learning can help to develop process models for complex systems with reduced computational burden, enabling online applications [4,6,45].

Hybrid models

Machine learning models can be fused with first-principle models, forming so-called hybrid models [22,46]. Focus has recently been given to data-driven modeling, which encloses or embeds specific properties of the underlying process principles to promote process understanding and reduce dataset size [16,47,48]. Hybrid models often have better extrapolation capabilities than pure data- based models due to incorporating first principles. They also

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

provide physical significance to the modeling structure and parameters while bounding the model predictions to the physical constraints of the system [4,47]. Hybrid models have been used, for example, in reaction systems, thermodynamics, biochemical processes, and fluid dynamics to estimate difficult-to-measure parameters and states using process data [4,47,49]. Further insights can be found in a recent review article [48]. In biotechnology, the interest in machine learning, particularly hybrid models, has increased in the last 20 years. A recent review [27] covers machine learning applications in biotechnology of more than 200 papers published in the last 20 years. For example, in metabolic engineering, hybrid models have been used to model intracellular dynamics (e.g., metabolism, enzyme regulation), simplifying model identification, control, and optimization [46,49,50]. For example, they have been used for modeling algae [51,52], bacteria [22,53], and for signaling pathways of animal cells [54–56].

6.2.2 Processes operating modes

(Bio)chemical processes are frequently divided into continuous, batch, and fed-batch processes, cf. Fig. 6.2. While dynamical system models of the form Eqs. (6.1) can be used to describe all three operating modes similarly, the control objectives are different. In the continuously operating processes, the goal is often to either stabilize the system states at specified steady-state conditions concerning disturbances while process optimizing efficiency or to follow a transition trajectory to reach desired operating conditions, cf. Fig. 6.2A [57]. These desired set points and trajectories are usually dictated by a higher-level controller/optimizer (see Section 6.2.3). The desired products are produced continuously without interruption, unlike (fed)batch operations. Changes in the inputs (feed) to the process result in a dynamical or transitional phase in the output states (production).

However, the residence time of the reactors might not be large enough to convert all substances into the desired products. Hence, recycling is often utilized, cf. the dashed lines in Fig. 6.2A. Recycling process streams can also come from downstream units, as indicated by the dashed arrow in Fig. 6.2A. Plant models can be generated using the dynamical data, online or offline, obtained from the process during operation.

In contrast, batch and fed-batch processes usually start with the same initial conditions, terminate in a finite time, and are operated repeatedly, cf. Fig. 6.2B [33,57]. Fed-batch processes have a continuous feed similar to continuous operation. However, the product is only extracted after a finite time, similar to the batch operating mode, cf. dashed line in Fig. 6.2B. The

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

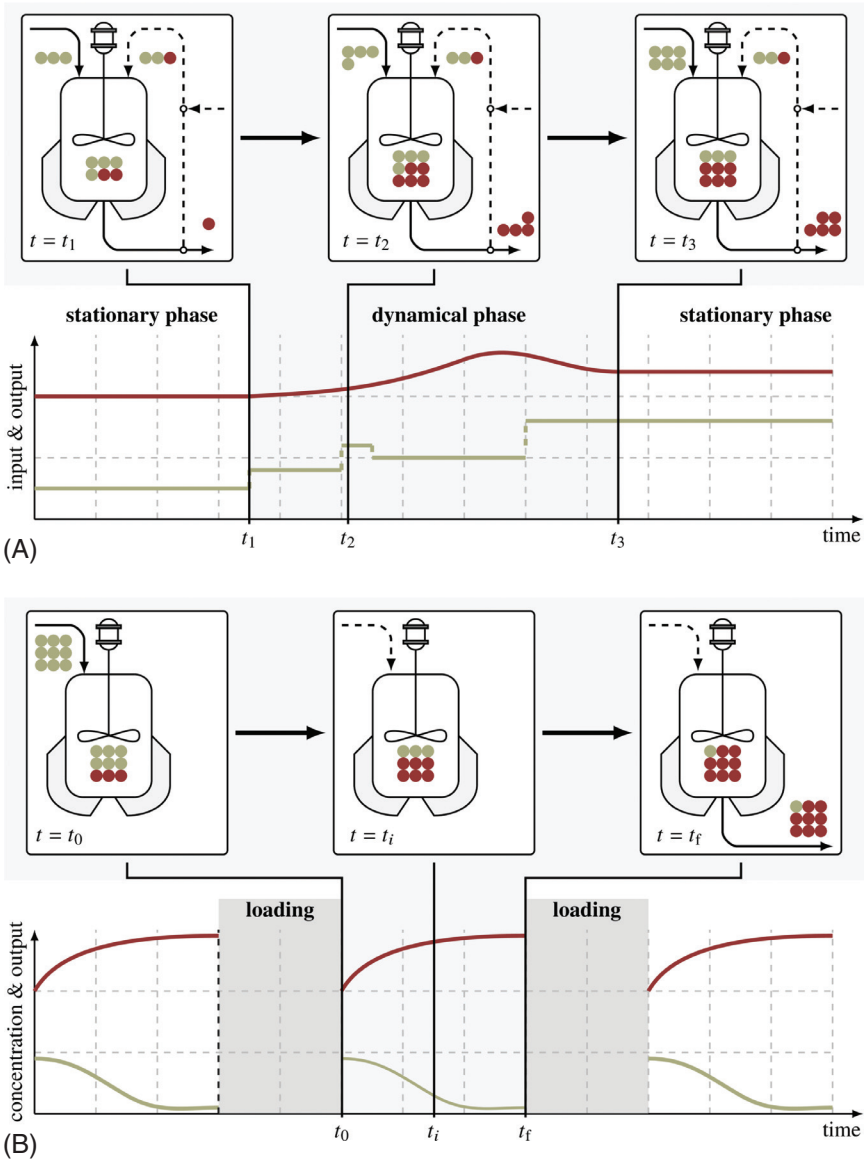


Figure 6.2 Illustration of the two typical process operating modes. The dashed lines represent an optional recycle (A) or an optional feed (B) to cover the fed-batch case.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

fed-batch operation ensures a supply of raw materials; however, larger reactor volumes are typically required [33]. Plant models can be generated using the dynamical data, online or offline, obtained from each batch.

Asymptotic properties such as stability are often of secondary importance for controller design for (fed)batch processes. While the safety of reactors that can show, for example, thermal runaway is essential, following a specified time-varying profile/recipe, reaching a given value at the end of the batch, minimizing batch time, or optimizing overall efficiency are of core interest. Controlling and optimizing (fed)batch processes is especially hard due to the significant state variation, which requires the consideration of the nonlinear behavior of the controlled process. While linearizations of the nonlinear process behavior can often be used for control, estimation, and modeling of continuous processes that are operated in a steady state, this is often inadequate for (fed)batch processes. However, the repetitive nature of (fed)batch processes can be utilized to tune and improve control performance over batches, cf. Fig. 6.2B [58,59]. Iterative learning control (see Section 6.5.2.3) and run-to-run (or batch-to-batch) learning and optimization are suited to these operation modes but should be able to cope with changing batch initial conditions.

The operating modes also influence how machine learning is integrated into the process. For example, for (fed)batch processes, some data might be available only at the end of the batch. In this case, the machine learning model can be trained once the batch is completed, resulting in a batch-to-batch learning approach, that is, where the models are updated only in between batches. For continuous time processes, the models can be updated with a different frequency depending on the case.

6.2.3 Control architecture

Manufacturing processes require a suitable control architecture that implements the mentioned objectives for continuous, batch, or fed-batch processes. The control architecture is responsible for achieving specified production quantity and quality operating points. Another important aspect is the immediate and guaranteed disturbance rejection and noise compensation, thus ensuring the safe and stable operation of the plant.

Modern chemical processes usually possess a hierarchical control structure to realize an economically optimal and safe operational process, cf. [33,60–62]. Fig. 6.3 shows a simplified representation of the different control layers. The actions/responses of each layer are executed in different time horizons. At the upper control level, optimal economic operation

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

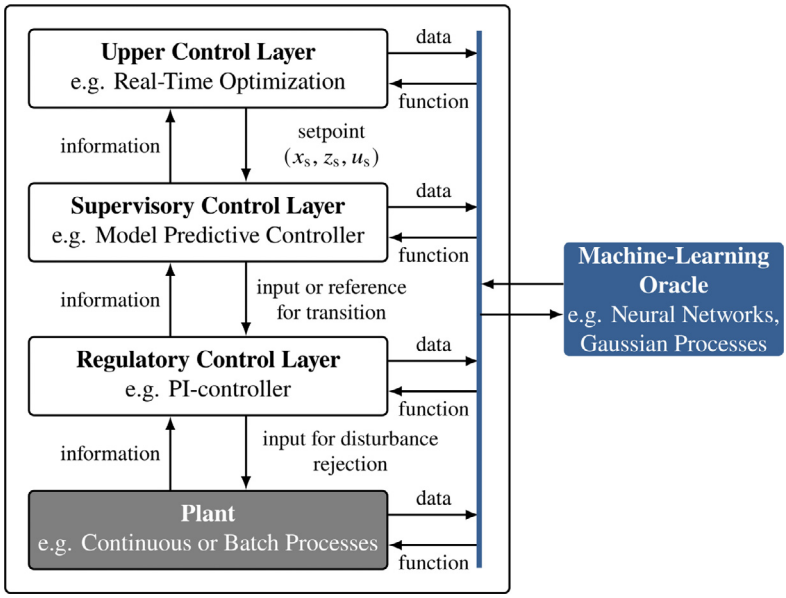


Figure 6.3 Typically encountered control architecture for (bio)chemical processes.

points are determined using the external specifications set at the planning and scheduling level for long periods and measurements from the plant. Depending on the application, these economic decisions about current production levels' set points are executed in hours or a few days. For this purpose, often a static optimization problem using an economic objective function, $J_e : \mathcal{X} \times \mathcal{Z} \times \mathcal{U} \times \mathbb{R}^{n_{eco}} \to \mathbb{R}$ (e.g., maximizing yield or profit, etc.), and the steady-state plant model, is computed. The corresponding problem is referred to as real-time optimization (RTO) [60,63]. For a given economic parameter, $p_{eco} \in \mathbb{R}^{n_{eco}}$ (e.g., describing the raw material or product prices), the general real-time optimization control law becomes

$$\mathrm{K_{RTO}} : \mathbb{R}^{n_{eco}} \to \mathcal{X} \times \mathcal{Z} \times \mathcal{U}, \left(p_{eco}\right) \mapsto \mathrm{K_{RTO}}\left(p_{eco}\right) := (x_s, z_s, u_s)$$

where the economically optimal set point (x_s, z_s, u_s) is obtained by solving:

$$(x_s, z_s, u_s) := \underset{(x,\, z,\, u)}{\arg\min} \quad J_{eco}\left(x, z, u, p_{eco}\right) \tag{6.3a}$$

$$\text{subject to} \quad 0 = f(x, z, u, p) + \mathrm{F}(x, z, p), \tag{6.3b}$$

$$0 = g(x, z, u, p) + \mathrm{G}(x, z, p) \tag{6.3c}$$

$$(x, z, u) \in \mathcal{X} \times \mathcal{Z} \times \mathcal{U} \tag{6.3d}$$

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Subsequently, the set point is sent to the control layer below in order to implement it (Fig. 6.3).

The underlying supervisory control level implements this set point, often utilizing a model predictive controller (MPC). The main idea of an MPC is based on the repeated solution of an optimal control problem on a moving horizon while incorporating new measurements of the plant state. In this way, the input of the system is continuously updated by comparing actual measurements with the predicted system behavior, considering various constraints. For general information on MPC, we refer to references [28,29,65,152]. There are many MPC formulations. We only sketch a basic set point tracking formulation, where the control law, given by:

$$\mathrm{K}_{\mathrm{MPC}} : \mathcal{X} \to \mathcal{U},\ (x_k) \mapsto \mathrm{K}_{\mathrm{MPC}}(x_k) = u_k := \mathrm{pr}_1\big(\mathbf{u}^*\big)$$

provides, for a current state x_k, the input u_k to be applied to the plant. This input value is given by solving the nonlinear program

$$\mathbf{u}^* := \underset{u_0,\ldots,u_{N-1}}{\arg\min} \sum_{l=1}^{N} \|\hat{x}_l - x_{\mathrm{s}}\|_Q + \|\hat{z}_l - z_{\mathrm{s}}\|_R + \|u_{l-1} - u_{\mathrm{s}}\|_S \tag{6.4a}$$

$$\text{subject to} \quad (\hat{x}_{l+1}, \hat{z}_{l+1}) = I\big(\hat{x}_l, u_l, p\big) \qquad l = 0, \ldots, N-1, \tag{6.4b}$$

$$0 = \hat{x}_0 - x_k, \tag{6.4c}$$

$$(\hat{x}_l, \hat{z}_l, u_l) \in \mathcal{X} \times \mathcal{Z} \times \mathcal{U} \quad l = 0, \ldots, N-1, \tag{6.4d}$$

$$(\hat{x}_N, \hat{z}_N) \in \mathcal{X}_{\mathrm{f}} \times \mathcal{Z}_{\mathrm{f}}. \tag{6.4e}$$

Here for simplicity of presentation, a quadratic cost function is used, where Q, R, and S are suitable penalty terms. The dynamical plant model is replaced by a discrete-time model. In addition to the path constraints (Eq. 6.4d), we can also define additional terminal constraints (Eq. 6.4e) that must be satisfied for the states at the end of the prediction horizon N. The idea of feedback comes from the fact that the current state value x_k of the system is used as the initial value for prediction in Eq. (6.4c). The current input value is then transmitted to the regulatory layer where plant information is processed over a short time horizon.

The regulatory layer ensures directly on the plant side that the decisions of the supervisory layer are implemented within seconds on the process elements. Furthermore, this layer supplies the rejection of high-frequency disturbances ensuring a stable process operation. Typical control laws used here are PID controllers or linear quadratic regulators.

Each of these control layers can be supported by machine learning algorithms. The blue box on the right side of Fig. 6.3 visualizes the machine

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

learning generation block, which supplies mappings or functions to each control layer while using data. These functions represent system models, such as static or dynamic and linear or nonlinear state space models. Moreover, the ML–Oracle can also provide control laws to replace components of the hierarchical control structure, cf. Section 6.5. For example, it can provide set points or feedback laws to replace the real-time optimization or a model predictive controller, respectively, using machine learning.

6.2.4 Monitoring

To enable closed-loop control, the control units must provide the current state information. It can be estimated or extracted from online measurements where noise should be filtered, and outliers and faults should be detected. Parameter estimation can be necessary to obtain reliable models for these tasks. In general, this is tackled by a monitoring unit. It often relies on using the plant's model Eqs. (6.1). However, data-based algorithms can also be used for monitoring purposes without needing a plant model. In both cases, that is, model-based and data-based monitoring, machine learning can help increase the monitoring or estimation quality. In the first case, machine learning can be used to model the system, while the (partially) learned model can be used in model-based estimation, such as moving horizon estimators [66]. In the second case, state, parameter, and fault estimation can be performed directly on the data using machine learning algorithms. In both cases, the ML–Oracle block builds the basis to design and train the machine learning components, outlined in Subsections 6.3.1 and 6.4.1.

6.3 The machine learning Oracle and machine learning approaches in a nutshell

As machine learning can support process control, modeling, monitoring, and operation, we introduce the concept of the so-called ML–Oracle. The proposed abstraction of the ML–Oracle provides a way to introduce a generic framework of how machine learning methods can be embedded in the control, modeling, monitoring, and optimization task. Furthermore, we briefly review some relevant machine learning algorithms, focusing on feed-forward and recurrent neural networks (RNN), Gaussian processes, and physics-informed learning.

6.3.1 The machine learning–Oracle

Machine learning can support different tasks in control, modeling, and monitoring. To abstract from the specific learning method used, we denote

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

the support provided by the learning as ML–Oracle. The Oracle obtains information—data and measurements—from the process (Fig. 6.3). It exploits the data and provides support in the form of a "function" learned. This function could be, in principle, any part of the control, modeling, or monitoring component, for example, a surrogate model, a part of a hybrid model, an element of the cost function used for predictive and optimal control, or the set of constraints considered.

In an abstract sense, the ML–Oracle $\mathcal{G}$ provides a map:

$$\mathcal{D}_{\text{tot}} \xrightarrow{\mathcal{G}} \text{S}, \tag{6.5}$$

from a dataset D_{tot} (and further process information), a function $\text{S} : \mathcal{F} \rightarrow \mathcal{L}$. This function provides a correlation between the feature set $\mathcal{F}$ and the label set $\mathcal{L}$ based on the data provided. Hence, the ML–Oracle uses machine learning algorithms to train data-based functions that perform either regression or classification tasks to support the modeling, control, or monitoring. The dataset D_{tot} provided by the system can contain various information, such as measured (time-series) data points, current parameter configurations of the plant, or measurement signals that are exploited in ML–Oracle for the training of the function S.

According to the specific application and design of the ML–Oracle, S can represent (parts of the) dynamic plant model or a control formulation. In the first case, the plant model is created or supported by the ML–Oracle using data-based techniques. The function can, for example, be provided F and/or G in Eqs. (6.1). In the second case, the ML–Oracle might provide parts of the control law. In this way, classical controller design procedures can be extended by using continuously updated data from the plant used to improve the control by learning. This allows to adapt to effects that are difficult to model in advance. Furthermore, the ML–Oracle can be used to approximate or learn the control law itself, for example, if the evaluation of the control law is expensive on embedded hardware, or if the control law itself is adjusted by learned parts.

The ML–Oracle, in general, uses data containing measured or observed information from the plant or the controllers. It can incorporate additional information and requirements if available. For instance, the ML–Oracle-generated plant model's interpretability, differentiability, and real-time capabilities might be important to increase acceptance or enable efficient gradient-based optimizations using the learned component.

Often, quantification of uncertainty of the learned part, such as the model uncertainty, might be essential, for example, to estimate the useful operating

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

area of the model provided. Suppose S provides a control law or a part of it. In that case, it often is intended to maintain closed-loop stability or ensure robustness in the presence of unpredictable disturbances or model uncertainties. These requirements should be considered in the ML–Oracle by using appropriate priors, approaches, or constraints in the design of the desired functions. In the following, we provide a mathematical abstraction of the ML–Oracle, including prior information, and elaborate on two specific algorithms: neural networks and Gaussian processes. Note that we focus on machine learning-based regression, rather than the classification.

6.3.2 Abstraction of the machine learning–Oracle

We introduced the ML–Oracle as a map that uses a dataset $\mathcal{D}_{\text{tot}}$ to learn a function:

$$\mathrm{S} \in \mathcal{M} := \{\xi : \mathcal{F} \to \mathcal{L}\}$$

between the elements of a feature space $\mathcal{F}$ and a space $\mathcal{L}$ of the labeled data. Fig. 6.4A illustrates this ML–Oracle and describes its basic structure. Based on the dataset $\mathcal{D}_{\text{tot}}$, the ML–Oracle produces the map S involving a coordination/planning step (lower blue box) and a training step (upper blue box).

6.3.2.1 The dataset

We assume that the total dataset $\mathcal{D}_{\text{tot}} = \{\mathcal{D}, \mathcal{D}_{\text{add}}\}$ used to generate S contains two subsets $\mathcal{D}$ and $\mathcal{D}_{\text{add}}$ encoding different information about the process. The dataset:

$$\mathcal{D} := \left\{\left(f_{\text{pre},1}, \ell_{\text{post},1}\right), \ldots \left(f_{\text{pre},n}, \ell_{\text{post},n}\right) \mid f_{\text{pre},i} \in \mathcal{F}_{\text{pre}}, \ell_{\text{post},i} \in \mathcal{L}_{\text{post}}\right\}$$

is a collection of measurements that indirectly represent the features and the labels to learn S. To obtain the features $f_i \in \mathcal{F}$ and labels $\ell_i \in \mathcal{L}$ needed to generate S, processing of the data $\mathcal{D}$ might be required. This processing is performed by the training block, cf. Fig. 6.4 and Section 6.3.2.1.

Remark 6.3.1. The dataset $\mathcal{D}$ is assumed to consist of labeled features, indicating supervised learning.

The labels in the dataset $\mathcal{D}$ can be either discrete or a continuous values. If a continuous set is considered, the learning task is called regression, otherwise it is called classification.

Additionally to the measured input–output data $\mathcal{D}$, the ML–Oracle might need to consider additional parameters. For example, in physics-informed learning, the training of a machine learning algorithm takes into account a first-principle model that contains certain parameters or

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

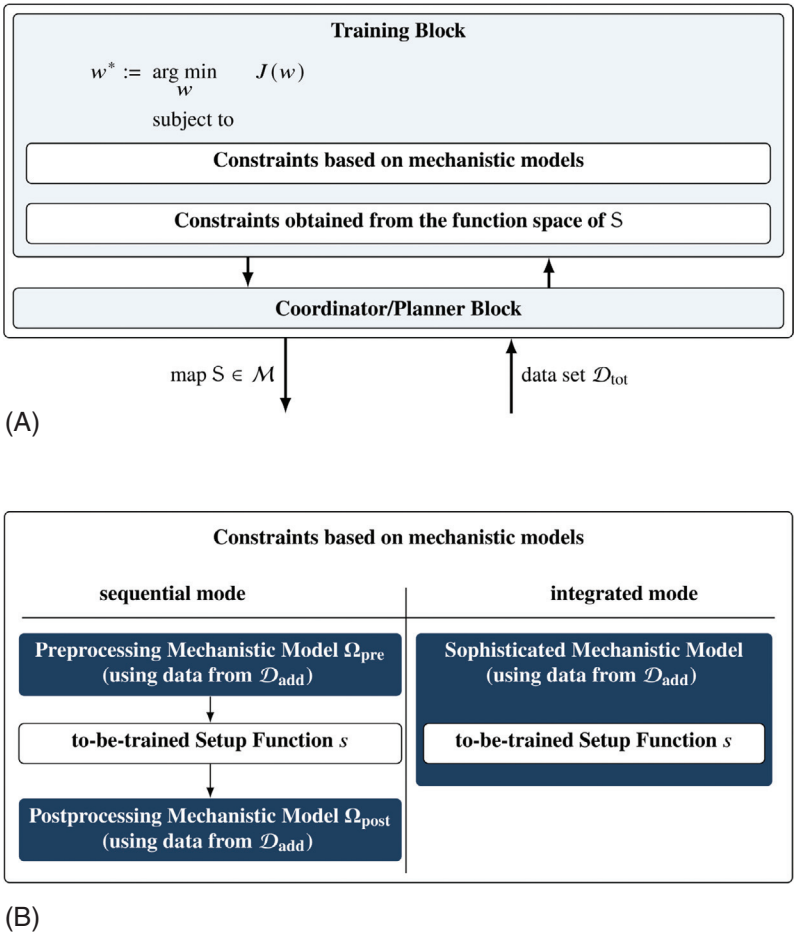


Figure 6.4 General structure of the machine learning (ML)–Oracle: (A) Generic structure of the ML–Oracle. (B) Possible constraints used in the ML–Oracle.

should satisfy certain constraints. The additional information about physical assumptions, knowledge, or restrictions which should be considered during the training process is stored in $\mathcal{D}_{\text{add}}$. The total dataset $\mathcal{D}_{\text{tot}} = \{\mathcal{D}, \mathcal{D}_{\text{add}}\}$ is passed to the coordination/planning block responsible for executing the training block and evaluating the trained model.

The coordinator

Commonly, the dataset $\mathcal{D}$ is divided into a training, test, and validation dataset. The coordinator/planner block is responsible for selecting these divisions and evaluating the performance of the trained algorithms. The

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

training data are used for learning, the test dataset is used for evaluating the performance on unseen examples, and the validation dataset is used to compare different machine learning models, to select the best-performing model. To speed up the training, the training dataset can be further divided into batches. These are a subset of the training dataset passed to the machine learning model before its parameters or weights are updated. The number of times the training dataset has been used for training is called the *epoch*.

The coordinator uses the dataset $\mathcal{D}_{\text{tot}}$ and performs the following tasks:

1. Selecting and forwarding of training data and additional parameters to the training block
2. Executing the training block
3. Evaluating the quality of trained function

Algorithm 6.1 formalizes this task, where $\mathcal{P}$ $(\mathcal{D})$ denotes the power set of the dataset $\mathcal{D}$.

Algorithm 6.1 Coordination of the ML–Oracle designing the data-based function.

Input: Data set $\mathcal{D}_{\text{tot}}$
Result: function $S \in \mathcal{M}$ trained on the data set $\mathcal{D}_{\text{tot}}$
while *true* **do**
  selecting subsets D_{train} and D_{test} from the data set $\mathcal{P}$ $(\mathcal{D})$
  running the training block using D_{train} and $\mathcal{D}_{\text{add}}$
  evaluating the function S using D_{test} in terms of e.g. inter- and extrapolation quality or constraint satisfaction
  if *Function S satisfies all conditions* **then**
    break
  end
end

Setup functions

In order to find the function S, the ML–Oracle might use a set of to-be-trained setup function

$$\mathrm{s} : \mathcal{F} \times \mathcal{W} \times \mathcal{P}(\mathcal{D}) \to \mathcal{L}$$

where $\mathcal{W}$ is the set of parameters and $\mathcal{P}$ $(\mathcal{D})$ denotes the power set of the incoming data $\mathcal{D}$. The to-be-trained setup function is chosen externally in accordance to the particular application. For a neural network, for instance, one chooses the number of layers and the way these layers are connected, as well as the activation function. The weights of the network and possibly the parameters of the activation function are not yet determined, whereas in a Gaussian process, one chooses the prior mean as well as the kernel function,

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

which is used to build the covariance matrix from a subset of the data. The hyperparameters of the kernel function are not yet specified and must still be trained. Once the parameters in have been determined, S is obtained by fixing the parameters and training data such that $\mathrm{S}(\bullet) = \mathrm{s}(\bullet, w, D)$, that is, S is the evaluation of s for a given set of parameters and data.

In general, two types of to-be-trained setup functions can be distinguished:

Directly data-based setup functions: The function s depends directly on a subset D of the original set $\mathcal{D}$ of data points, that is, $\mathcal{P}(\mathcal{D}) \ni D \neq \emptyset$. Machine learning algorithms that use these type of setup functions do not only use data during training, that is, to determine the parameters in $\mathcal{W}$, but they also use data for evaluation or prediction during training. Typical examples are Gaussian process or kernel-based interpolation.

Indirectly data-based setup functions: The function s is not directly dependent on the data points. Hence, the selected element D from the power set of the data is empty, that is, $D \neq \emptyset$. Instead of using data directly, the elements of $\mathcal{D}$ are used to determine parameters in $\mathcal{W}$. The prediction or evaluation of the trained algorithm then only relies on $\mathcal{F}$ and $\mathcal{W}$. Typical examples of machine learning algorithms that use this type of setup functions are neural networks or linear regression techniques.

The chosen to-be-trained setup function is used inside the training block to parameterize the machine learning algorithm, which is described in the following section.

The training block

In most machine learning approaches, the training of a function is formulated in terms of an optimization problem in order to find the best model. Depending on the particular application, the objective function can be classified into two main categories.

Error-based objectives: Here, the aim is to minimize the deviation between the measured label data and the predicted label data based on. The objective function

$$J_{\mathrm{er}} : \mathcal{L}_{\mathrm{post}} \times \mathcal{L}_{\mathrm{post}} \to \mathbb{R} \tag{6.6}$$

maps from a preselected set of data points $D \in \mathcal{P}(\mathcal{D})$ and the corresponding predicted information using the to-be-trained setup function on the real numbers. Typical examples are functions for mean absolute

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

error (MAE), mean squared error (MSE), binary cross entropy (BCE), or cross entropy loss, etc.

Evidence-based objectives: This type of objective aims to describe the capability of the distribution of the parameter $w \in \mathcal{W}$ to explain the observed data. The objective function

$$J_{\text{ev}} : \mathcal{P}(\mathcal{D}) \times \mathcal{W} \to \mathbb{R}, \tag{6.7}$$

maps from a preselected set of data points $D \in \mathcal{P}(\mathcal{D})$ and the parameters required to define a to-be-trained setup function to the real numbers.

These types of objectives relate to directly data-based setup functions. An example for this type of objective functions is the likelihood function.

In addition to these two types of objective functions, regularization functions are often used balance the influence of the parameters and to avoid overfitting. In this way, it is possible to modify the structure of the to-be-trained setup function by weighting different terms or components within such a function.

The goal of the training block is to identify the optimal parameter $w^* \in \mathcal{W}$. Once the parameter is found, S can be constructed based on s evaluated at w^*. In the case of indirectly data-based setup functions maps only from features (and not parameters) for further applications. In case of directly data-based setup functions, the training data $D \in \mathcal{P}(\mathcal{D})$ set in must also be fixed to obtain.

Physics-informed learning

For many applications, besides the main task of minimizing the objectives Eqs. (6.6) or (6.7) (in conjunction with regularization terms), one should also ensure that certain (physical) constraints, laws, or known relationships are satisfied. Embedding these relations into the learning is commonly denoted as physically informed machine learning.

One could, for example, require that the learned function satisfies properties such as continuous differentiability, monotonicity, or periodicity, that is, the ML–Oracle outcome should be an element of "the restricted function space":

$$\mathcal{M}_{\text{r}} := \left\{\xi : \mathcal{F} \to \mathcal{L} \mid \xi \text{ has to satisfy additional conditions}\right\}.$$

For instance, in regression, one might be only interested in continuously differentiable functions defined by

$$C^k(\mathcal{F}, \mathcal{L}) := \left\{\xi : \mathcal{F} \to \mathcal{L} \mid \xi \text{ is } k\text{−times continuously differentiable}\right\}.$$

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Other restrictions could be to permit only monotonous functions, positive/negative definite functions, or bounded once. These constraints are considered in the training block of the ML–Oracle, compare Fig. 6.4A.

A second type of constraint arises from mechanistic models as illustrated in Fig. 6.4B. Mechanistic models are used when the sets $\mathcal{F}_{\text{pre}}$ and $\mathcal{L}_{\text{post}}$ associated with the data points in $\mathcal{D}$ do not directly correspond to the feature and label space, that is, $\mathcal{F}$ and $\mathcal{L}$. In other words, the learning of the desired map from $\mathcal{F}$ to $\mathcal{L}$ can only use data from $\mathcal{F}_{\text{pre}} \neq \mathcal{F}$ and $\mathcal{L}_{\text{post}} \neq \mathcal{L}$. In such cases, it is necessary to translate between these sets, which can be done by means of additional models. These models usually have the form of implicitly or explicitly given algebraic or dynamic systems of equations. For instance, it is necessary to convert between different physical variables (e.g., temperature, pressure, concentrations, chemical potentials, etc.) when not all of them can be measured directly. Depending on the way these physics-informed mechanistic models are applied, one can distinguish between sequential and integrated methods. These methods use additional parameters $d \in \mathcal{D}_{\text{add}}$ that must be provided to the ML–Oracle in addition to the data points.

Sequential methods: Sequential methods use physics-informed models:

$$0 = \tilde{\Omega}_{\text{pre}}\big(f_{\text{pre}}, f\big), \quad \text{and} \quad 0 = \tilde{\Omega}_{\text{post}}\big(\ell, \ell_{\text{post}}\big),$$

to convert the measured data so that the setup function can process it. Typical examples might include thermodynamic equations that describe the relationship between different state variables (e.g., temperature, pressure, volume, etc.), thermodynamic potentials (e.g., enthalpy, entropy, etc.) or material properties (e.g., densities, capacities, etc.). If these models are (locally) given explicitly, that is, in the form of mappings

$$\Omega_{\text{pre}} : \mathcal{F}_{\text{pre}} \to \mathcal{F}, \quad \text{and} \quad \Omega_{\text{post}} : \mathcal{L} \to \mathcal{L}_{\text{post}}$$

It is possible to integrate these mappings directly into the setup function. The result is a hybrid model. In general, sequential methods of physics-informed machine learning allow to outsource the transformation of the data in $\mathcal{D}$. More precisely, the conversion from f_{pre} to f and from ℓ_{post} to ℓ can be performed offline and outside of the training block.

Table 6.1 shows two examples of a training procedure where the data is converted sequentially. It provides the mathematical formulation of the training of a given setup function s to obtain optimal parameters w^*. In both cases presented in Table 6.1, additional constraints enter the optimization, which reflect the mechanistic model. In Case 1, this

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Table 6.1 Example for training a map G_{ex} using sequential methods. The system in both cases is given as an algebraic model.

Setup function:	indirectly data-based setup function to be trained
Objective function:	error-based to minimize deviation between simulated & measured data
Case 1:	$z_1 = g_{ex}(u), z_2 = G_{ex}(z_1), y = h(z_2)$ with $z = (z_1, z_2)$
Data set:	$\mathcal{D} := \{(u_k, y_k) \mid u_k \in \mathcal{F}_{pre}, y_k \in \mathcal{L}_{post}, k = 1, \ldots, N\}$
Physical model:	define $\Omega_{pre} := g_{ex}, \Omega_{post} := h$
Training:	$w^* := \arg\min_w \sum_k \lVert y_k - \hat{y}_k \rVert_Q$ subject to $\hat{y}_k = h \circ s(g_{ex}(u_k), w) \quad k = 1, \ldots, N$
ml-oracle output:	$G_{ex}(z_1) \approx S(z_1) := s(z_1, w^*)$
Case 2:	$0 = g(z_1, u), z_2 = G_{ex}(z_1), y = h(z_2)$ with $z = (z_1, z_2)$
Data set:	$\mathcal{D} := \{(u_k, y_k) \mid u_k \in \mathcal{F}_{pre}, y_k \in \mathcal{L}_{post}, k = 1, \ldots, N\}$
Physical model:	define $\tilde{\Omega}_{pre} := g, \Omega_{post} := h$
Training:	$w^* := \arg\min_w \sum_k \lVert y_k - \hat{y}_k \rVert_Q$ subject to $0 = g(z_{1,k}, u_k) \quad k = 1, \ldots, N,$ $\hat{y}_k = h \circ s(z_{1,k}, w) \quad k = 1, \ldots, N$
ml-oracle output:	$G_{ex}(z_1) \approx S(z_1) := s(z_1, w^*)$

mechanistic model is given explicitly; Case 2 uses implicit mechanistic constraints. Even though these constraints are stated in Table 6.1 as a part of the training optimization, the related transformations could be performed outside of the optimizations, since Ω_α or $\tilde{\Omega}_\alpha$, $\alpha \in \{\text{pre}, \text{post}\}$ do not depend on the parameters w.

Integrated methods: Integrated methods deal with physics-informed models that are mutually coupled with the setup function. In this case, it is not possible to calculate the features f and labels ℓ from $\mathcal{D}$ without evaluating the model using parameters w which are still unknown and need to be trained. One common example of such models is dynamical system, where the setup function takes the role of algebraic equations within the system description. For instance, the setup function can represent the kinetic equation in a dynamic model of a reactor. However, often the reaction rates are not directly measurable and the measurement data are composed of the concentration profiles. Learning the reaction rate from these data includes the solution of a dynamical system equation, which must be performed in each training step. Table 6.2 illustrates the training procedure, where the data-based model is part of an

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Table 6.2 Example for training a map G_{ex} using integrated methods. The systems are either given by an algebraic or a dynamical model.

Setup function:	indirectly data-based setup function to be trained
Objective function:	error-based to minimize deviation between simulated & measured data
Case 1:	$\mathbf{0} = \boldsymbol{g}(\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{u}), \boldsymbol{z}_2 = \mathrm{G}_{\mathbf{ex}}(\boldsymbol{z}_1), \boldsymbol{y} = \boldsymbol{h}(\boldsymbol{z}_1, \boldsymbol{z}_2)$ **with** $\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2)$
Data set:	$\mathcal{D} := \left\{(u_k, y_k) \mid u_k \in \mathcal{F}_{\text{pre}}, y_k \in \mathcal{L}_{\text{post}}, k = 1, \ldots, N\right\}$
Physical model:	$0 = g\big(z_1, \mathrm{s}(z_1, w), u\big) =: \tilde{g}(z_1, w, u)$ and define $\tilde{h}(z_1, w) := h\big(z_1, \mathrm{s}(z_1, w)\big)$ using s
Training:	$w^* := \arg\min_w \sum_k \lVert y_k - \hat{y}_k \rVert_Q$ subject to $0 = \tilde{g}(z_{1,k}, w, u_k) \quad k = 1, \ldots, N,$ $\hat{y}_k = \tilde{h}(z_{1,k}, w) \quad k = 1, \ldots, N$
ml-oracle output:	$\mathrm{G}_{\text{ex}}(z_1) \approx \mathrm{S}(z_1) := \mathrm{s}(z_1, w^*)$
Case 2:	$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{u}), \boldsymbol{z} = \mathrm{G}_{\mathbf{ex}}(\boldsymbol{x}), \boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x})$
Data set:	$\mathcal{D} := \left\{(t_k, y_k) \mid t_k \in \mathcal{F}_{\text{pre}}, y_k \in \mathcal{L}_{\text{post}}, k = 1, \ldots, N\right\}$
Additional data:	$\mathcal{D}_{\text{add}} := \{x_0, (u_k)_{k \in \mathbb{T}}\}$ with $\mathbb{T} := \{0, 1, \ldots, N-1\}$
Physical model:	one-step integrator $(x_{k+1}, z_{k+1}) = I(x_k, u_k, w)$ using s
Training:	$w^* := \arg\min_w \sum_k \lVert y_k - \hat{y}_k \rVert_Q$ subject to $(\hat{x}_{k+1}, z_{k+1}) = I(\hat{x}_k, u_k, w) \quad k = 0, \ldots, N-1,$ $\hat{x}_0 = x_0,$ $\hat{y}_k = h(\hat{x}_k) \quad k = 1, \ldots, N$
ml-oracle output:	$\mathrm{G}_{\text{ex}}(x) \approx \mathrm{S}(x) := \mathrm{s}(x, w^*)$

algebraic model (Case 1) and a dynamical model (Case 2). Here, the system must be solved during training to obtain data that can be compared with the measured information. The constraints in the optimization describe the system equations and directly depend on the parameters w. Hence, there is no possibility to precalculate features and labels detached from the optimization to simplify training.

In summary, physics-informed knowledge can be integrated into the design process of the setup function and also into the training of the parameters in various ways. In this way, physical models can be used to convert between measured data and variables needed for training. Also the choice of a setup function to be trained can be motivated by physical and technical aspects. For instance, the initial network topology of a neural network (e.g., number of hidden layers and choice of the activation function) or the kernel function of a Gaussian process can be crucial in approximating the data. In particular, the choice of setup function determines how efficiently

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

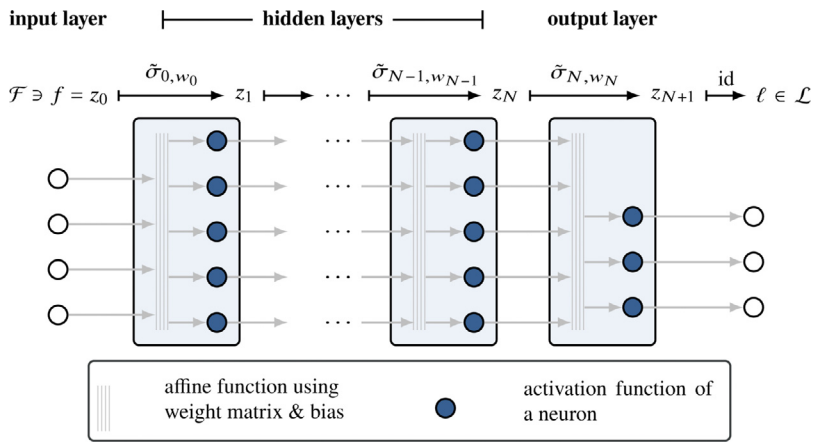


Figure 6.5 Representation of a feed-forward neural network.

additional constraints on the function space $\mathcal{M}_r$ can be incorporated into the optimization. Indeed, some setup functions satisfy certain properties (e.g., periodicity) directly by their structure. Also, the choice of initial parameters or boundaries results from prior knowledge of experienced engineers or from technical restrictions, see reference [38].

6.3.3 Machine learning Oracle examples

We now outline two specific examples for ML–Oracle s, namely neural networks and Gaussian processes. For this, we assume that the features and labels are measured directly and no conversion between $\mathcal{F}_{\text{pre}}/\mathcal{F}$ or $\mathcal{L}_{\text{post}}/\mathcal{L}$ is required via physical models.

6.3.3.1 Feed-forward artificial neural networks

Artificial neural networks are one of the most flexible and wide-spread machine learning approaches. They are inspired by the neural networks of the brain [67]. Artificial neural networks are formed by a set of neurons that can be organized in layers (Fig. 6.5) or in more complex structures [68,69], where each neuron is connected to other neurons in the network. We begin with feed-forward neural networks (FNNs) (cf. Fig. 6.5), that is, each neuron in a layer is connected to the neurons in the next layer without self-connections or loops. Each of these connections is weighted with a parameter that indicates the importance of the signal to the next neuron. Information is entered in the input layer and transported and transformed through the network to obtain the corresponding output. The

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

transformation is performed in the neurons by applying nonlinear activation functions to the weighted sum of all inputs and a bias to each neuron.

The feature $f =: z_0$ is the input to the overall network. Training a network aims at finding parameters, such as the weights, to produce the network output $z_{N+1} \in \mathcal{L}$ that mimics the label data.

Neural networks belong to the group of machine learning techniques that use indirectly data-based setup functions. This means that the measured data are only used to train the parameters and not during inference. The to-be-trained setup function S_{FNN} is composed of elementary functions

$$z_{l+1} = \tilde{\sigma}_{l,w_l}(z_l) := \sigma_l(W_l z_l + b_l), \tag{6.8}$$

which describe the processing in each layer. Here, $\sigma_l : \mathbb{R}^{n_{l+1}} \to \mathbb{R}^{n_{l+1}}$ is the activation function, and $w_l = (W_l, b_l) \in W_l := \mathbb{R}^{n_{l+1},n_l} \times \mathbb{R}^{n_{l+1}}$ is the parameter of the layer l with the weight matrix W_l and bias b_l. The output information $z_l \in \mathbb{R}^{n_l}$ of layer l builds the input to layer $l+1$, where n_l indicates the number of neurons in the layer l. The numbers $n_0 = n_{\mathrm{f}}$ and $n_{N+1} = n_\ell$ are identical to the dimension of the feature space and the label space. The parameters of a neural network, that is, the elements of $\mathcal{W}$, consist of the entries in the weighting matrices and the bias terms.

By repeating this operation for all the N layers, we obtain the to-be-trained setup function

$$\mathrm{S}_{\mathrm{FNN}} : \mathcal{F} \times \mathcal{W} \to \mathcal{L}, (f, w) \mapsto \mathrm{s}_{\mathrm{FNN}}(f, w) := \tilde{\sigma}_{N,w_N} \circ \ldots \circ \tilde{\sigma}_{0,w_0}(f)$$

The individual parameters w_l form the entire parameter $w = (w_0, \ldots, w_N) \in \mathcal{W} := \mathcal{W}_0 \times \ldots \mathcal{W}_N$. The weights W_i and bias b_i of every layer are optimized during the training procedure using error-based objectives (Eq. 6.6). In other words, the error between the predicted and measured labels is minimized [70]. In practice, gradient-based optimization algorithms are often used to tackle the training of artificial neural networks. Based on the specific structure of deep neural networks (networks with many hidden layers), special subtypes of gradient-based techniques are used for the training procedure. One of them is back propagation, which allows iterative updating of parameters while preventing the network from getting stuck in local minima due to vanishing gradients. For more detailed discussion on vanishing gradient and back propagation, readers are referred to [71] and [72].

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

6.3.3.2 Recurrent neural network

Feed-forward neural networks are static maps between inputs and outputs. In some cases, it is useful to express the current output not only as a function of the current input but also of the previous ones. This is the case where the order of the measured features is important to predict the labels, for example, in time series data, text generation, or text prediction. Recurrent neural networks are neural networks that serve this purpose by providing the network with a *memory effect* which is often useful when modeling dynamical systems. To integrate this memory effect into the network, one uses so-called *hidden states*, which allow to transfer the information from previous evaluations into further ones via information feedback. For the sake of simplicity, we assume only one processing layer to describe the principle of an RNN. More precisely, we consider a network with only one other layer besides the input and output layer. In general, there are different options within such a layer and thus different architectures to incorporate the memory effect into the network. In the following, we focus only on a single RNN cell, further structures are, for example, gated recurrent unit (GRU) [73] or long short-term memory (LSTM) [74].

An RNN cell is a kind of hidden layer formed by two sublayers of neurons (Fig. 6.6). The first layer is used to calculate the hidden states, while the second layer defines how the labels are derived from these states. To bring the RNN cell architecture into our framework, the following two assumptions are used:

1. The feature and label space are of the form

$$\mathcal{F} = \bar{\mathcal{F}} \times \mathcal{H} \quad \text{and} \quad \mathcal{L} = \bar{\mathcal{L}} \times \mathcal{H}. \tag{6.9}$$

Here, the set $\mathcal{H}$ contains the so-called hidden states allowing to describe a parameter-dependent mapping. The sets $\bar{\mathcal{F}}$ and $\bar{\mathcal{L}}$ represent the original domain and the codomain if one would use an FNN to describe the setup function.

2. The part of the labels containing the internal state h is fed back to the input of the neuron (Fig. 6.6).

The feature and label space is artificially extended in assumption 1 by the hidden states to allow to feeding them back as a part of the cell output and input. The second assumption gives us the ability to store and reuse information once it has been computed. In other words, the neuron is equipped with a memory. The elementary functions of the individual

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

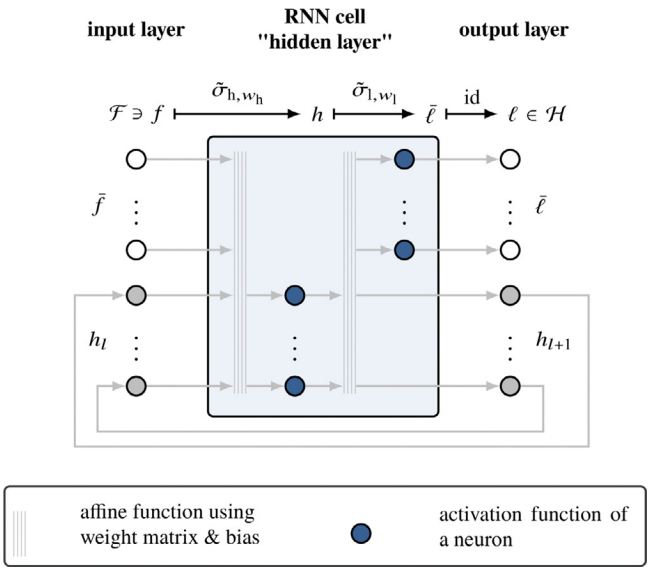


Figure 6.6 Representation of a recurrent neural networks (RNN). Note that we used the subscript *l* to highlight the memory effect of the network due to the feedback.

sublayers within an RNN cell are defined as follows

$$\begin{aligned} 1^{\mathrm{st}} \text{ layer}: \quad & h_{l+1} = \tilde{\sigma}_{\mathrm{h},w_\mathrm{h}}\big(f, h_l\big) := \sigma_\mathrm{h}\big(W_\mathrm{f} f + W_\mathrm{h} h_l + b_\mathrm{h}\big), \\ 2^{\mathrm{nd}} \text{ layer}: \quad & \ell = \tilde{\sigma}_{1,w_1}(h_{l+1}) \ := \sigma_1(W_1 h_{l+1} + b_1) \end{aligned}$$

Here, the maps $\sigma_\mathrm{h} : \mathbb{R}^{n_\mathrm{h}} \to \mathbb{R}^{n_\mathrm{h}}$ and $\sigma_1 : \mathbb{R}^{n_\mathrm{h}} \to \mathbb{R}^{n_\ell}$ are the activation functions.

The parameters $w_\mathrm{h} = (W_\mathrm{f}, W_\mathrm{h}, b_\mathrm{h}) \in W_\mathrm{h} := \mathbb{R}^{n_\mathrm{h}, n_\mathrm{f}} \times \mathbb{R}^{n_\mathrm{h}, n_\mathrm{h}} \times \mathbb{R}^{n_\mathrm{h}}$ and $w_1 = (W_1, b_1) \in W_1 := \mathbb{R}^{n_\mathrm{h}}, n_\mathrm{h} \times \mathbb{R}^{n_\mathrm{h}}$ consist of the weight matrices (W_f,W_h. and .W_1) and bias values (b_h and .b_1) that have to be trained. Using Eq. (6.9), the setup function of a single RNN cell can be defined as:

$$\mathrm{s}_\mathrm{RNN} : \bar{\mathcal{F}} \times \mathcal{H} \times \mathcal{W} \to \bar{\mathcal{L}} \times \mathcal{H}, \big(f, h_l, w\big) \mapsto \mathrm{s}_\mathrm{RNN}\big(f, h_l, w\big) := (\ell, h_{l+1}),$$

where $w := (w_\mathrm{h}, w_1) \in \mathcal{W} := \mathcal{W}_\mathrm{h} \times \mathcal{W}_1$.

6.3.3.3 Gaussian process

Learning via Gaussian processes (GPs) differs from that of neural networks in terms of the to-be-trained setup function, the objective during training, and the type of data processing. The basic idea is that the labels are interpreted as normally distributed random variables, which depend on the features. To describe the similarity of data points, one uses a kernel function σ :

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

$\mathcal{F} \times \mathcal{F} \times \mathcal{W} \to \mathbb{R}_0^+$, assigning a positive real number to two features. The kernel specific parameter $\mathrm{h} \in \mathcal{H}$ is either assumed to be given or trained using data and an evidence-based objective function. The kernel function can be used to evaluate neighborhoods or relationships between two points in the feature space. If the value of σ is greater for two points than for two other points, it can be concluded that there is a stronger coupling between them. This in turn affects the evaluation of the labels among each other. In general, the kernel function must satisfy the following two properties:

Symmetric: For a fixed parameter $\mathrm{h} \in \mathcal{H}$, the kernel function is symmetric, that is, $\sigma(f_1, f_2, \mathrm{h}) = \sigma(f_2, f_1, \mathrm{h})$

Upper bounded: For a fixed parameter $\mathrm{h} \in \mathcal{H}$ and a fixed feature $f \in \mathcal{F}$, the function $\tilde{\sigma} : \mathcal{F} \to \mathbb{R}_0^+, \tilde{\sigma}\left(\tilde{f}\right) := \sigma\left(f, \tilde{f}, \mathrm{h}\right)$, is upper bounded by $\tilde{\sigma}\left(f\right)$.

The symmetry is important because two arbitrary points in the feature space $\mathcal{F}$ are in the same relationship no matter which point is used for which argument of σ. The second property ensures that the similarity between an arbitrary point $\tilde{f}$ in the feature space $\mathcal{F}$ and a base point f is not greater than the similarity of f to itself. A common example of a kernel function used in the literature is the squared exponential function

$$\sigma_{\mathrm{se}}\left(f_1, f_2, \mathrm{h}\right) := \mathrm{h}_1 \exp\left(\frac{\sum_\alpha \left(f_1^\alpha - f_2^\alpha\right)^2}{\mathrm{h}_2}\right)$$

Where, f_i^α denotes the αth component of f_i and $\mathrm{h}_1, \mathrm{h}_2 \in \mathbb{R}_+$are the parameters.

By means of the kernel function, a directly data-based setup function

$$\mathrm{s}_{\mathrm{GP}} : \mathcal{F} \times \mathcal{W} \times \mathcal{P}(\mathcal{D}) \to \mathcal{L}$$
$$(\tilde{f}, w, D) \mapsto \mathrm{s}_{\mathrm{GP}}(\tilde{f}, w, D) := \sum_{\alpha,\beta=1}^{n} \left(K(w, D)^{-1}\right)^{\alpha\beta} \ell_\beta \sigma\left(f_\alpha, \tilde{f}, \mathrm{h}\right)$$

can be designed using the incoming data $D = \left\{\left(f_1, \ell_1\right), \ldots, \left(f_n, \ell_n\right)\right\} \in \mathcal{P}(\mathcal{D})$ and the hyperparameter $w := (\mathrm{h}, \nu) \in \mathcal{W} := \mathcal{H} \times \mathbb{R}_+$. Here, h denotes the kernel specific parameter and $\nu \in \mathbb{R}_+$the variance. The matrix-valued function

$$K : \mathcal{W} \times \mathcal{P}(\mathcal{D}) \to \mathbb{R}^{n,n},$$
$$(w, D) \mapsto K(w, D) := \begin{pmatrix} \sigma\left(f_1, f_1, \mathrm{h}\right) & \ldots & \sigma\left(f_1, f_n, \mathrm{h}\right) \\ \vdots & \ddots & \vdots \\ \sigma\left(f_n, f_1, \mathrm{h}\right) & \ldots & \sigma\left(f_n, f_n, \mathrm{h}\right) \end{pmatrix} + \nu\mathbb{I},$$

yields the covariance matrix and thus allows to describe the relationship of the features to each other. Due to the symmetry property of the kernel

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

function, it follows that $K(w, D)$ is a symmetric matrix. In the literature, the function s_{GP} is referred to as a posterior mean with a prior mean of zero, see [64].

The benefit of Gaussian processes arises from the stochastic nature of this approach. In addition to the s_{GP} function, another function κ can be generated that describes a covariance around s_{GP} and thus defines a confidence interval. This posterior covariance function reads

$$\kappa : \mathcal{F} \times \mathcal{W} \times \mathcal{P}(\mathcal{D}) \to \mathbb{R}_+, (\tilde{f}, w, D) \mapsto \kappa(\tilde{f}, w, D)$$

where

$$\kappa(\tilde{f}, w, D) := \sigma(\tilde{f}, \tilde{f}, \mathrm{h}) - \sum_{\alpha,\beta=1}^{n} \left(K(w, D)^{-1}\right)^{\alpha\beta} \sigma\left(f_\alpha, \tilde{f}, \mathrm{h}\right) \sigma\left(f_\beta, \tilde{f}, \mathrm{h}\right)$$

In order to train the hyperparameters $w \in \mathcal{W}$, one chooses an evidence-based objective function to maximize the reliability of the data.

Table 6.3 Selected references for the presented machine learning methods for modeling and control. The list is incomplete and has a special focus toward machine learning methods for model predictive control.

Network type	References
Feed-forward NNs	[42,46,49,53,75–89]
Recurrent NNs	[90–114,116,210]
Gaussian process	[117–134]

6.4 Machine learning-supported modeling for monitoring and control

The increased availability of plant data as a result of cheaper data collection and storage combined with faster model development has increased the application of machine learning in the (bio)chemical industries [1]. Machine learning has been applied to a wide spectrum of fields including energy, base chemicals, petrochemicals, oil and gas, polymers, pharmaceuticals, food, beverage, biotechnology, mineral industry, and water [2,37].

In this section, we briefly outline how machine learning can be used to enable or enhance process monitoring and control providing models of the system to the estimators or controllers. Hence, data are used in the ML–Oracle to develop a plant model Eqs. (6.1) using data-based and

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

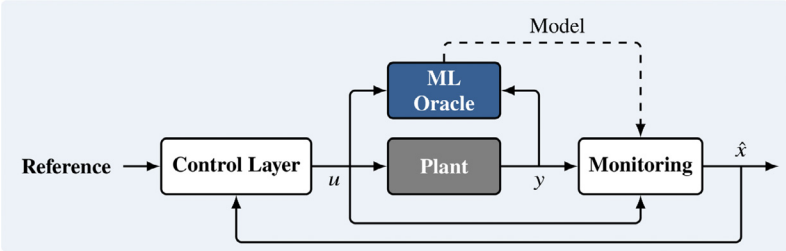


Figure 6.7 Learning system models for process monitoring.

hybrid modeling techniques. We outline how machine learning models of a system can be used for monitoring in Subsection 6.4.1 and for control in Subsection 6.4.2.

6.4.1 Learning system models for process monitoring

We summarize and review applications of data-based plant modeling to support plant monitoring (cf. Fig. 6.7), focusing on online and offline applications, covering dimensionality reduction, and parameter estimation, as well as online applications, covering state estimation and fault detection.

6.4.1.1 Dimensionality reduction with machine learning

High-fidelity plant models are valuable for engineers as they improve the process understanding, enable plant design, monitoring, and control [32–34]. To obtain accurate models, first-principle models have been developed for complex processes that capture the dynamics, hydraulics, physicochemical properties, or thermodynamics [32,34,135]. This first-principle modeling involves solving nonlinear differential equations, numerical integration, parameter fitting, and root finding methods [34,135]. Since these operations can be computationally challenging, overly complex models might not be suited for real-time applications, optimal process design, or control due to their complexity [32]. Machine learning algorithms have been used to tackle this challenge by learning reduced order models for decreased computational offline and online demands. For example, governing mechanisms have been identified using machine learning techniques to obtain low dimensional models of the essential system dynamics [34].

However, the use of machine learning models, instead of first-principles models, does not necessarily lead to increased computational speed or model order reduction. Training these machine learning models on large

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

datasets can require significant (offline) computational resources. Hence, sampling design methods used for reduced order model development have been developed, see, for instance, reference [136] for more information. In reference [136], Gaussian process and neural network surrogate models were developed using data generated by a complex thermodynamic model for liquid–liquid equilibrium phase prediction. Both reduced order models provided accurate predictions 36 times faster than the complex thermodynamic model. In reference [12], a hybrid reduced order model for fluid dynamics was developed to speed up the prediction of both velocity and pressure fields for external and internal flow problems. It was noted by Raissi et al. [12] that the approach has the benefit of providing accurate predictions regardless of the initial and boundary conditions. This implies that the model is flexible and can be used on the domain of interest, decreasing the amount of calculation and data required. The integration of first principles into the reduced order model enabled its application to model blood flow in an intracranial aneurysm, where the model was able to predict the velocity and pressure fields accurately.

Remark 6.4.1. In general, direct data-based machine learning methods suffer more from computational complexity for big datasets compared to indirect data-based algorithms such as neural networks which come with reduced computational effort required during prediction. For instance, the computational complexity of Gaussian processes scales cubically with the size of the dataset which makes online applications difficult [138]. However, algorithms such as sparse Gaussian processes and data update schemes have been developed to circumvent this drawback. For example, Fezai et al. [138] developed a Gaussian regression process method that uses only a subset of the dataset for offline training. Online training is implemented on new data to reduce the computational complexity and ensure sufficient accuracy.

6.4.1.2 Parameter and state estimation with machine learning

Certain phenomena, mechanisms, and physicochemical properties in (bio)chemical processes are sometimes unknown or only partly known [4,37,39,139]. Some chemical properties cannot be measured online and require laboratory analysis, which suffers from large time delays [47]. The time delays for online analyzers, such as gas chromatography, poses significant challenges for accurate online monitoring and control [45,140]. Moreover, industrial analyzers are expensive to acquire and maintain as some might not be able to be calibrated online [6,47,141].

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

These challenges can be addressed using the plant model as an estimator, also referred to as a soft sensor [141]. Machine learning has gained popularity in the application as estimators due to their robustness, simplicity, and flexibility. Different artificial neural network structures can be used for estimation, each with their own advantages and limitations, as summarized by Ali [45]. A guideline for estimator development and examples of data-based and hybrid estimators based on previous literature work are provided by Ali [45]. The objective is to estimate the unmeasurable variables using the measurable variables for online monitoring, control, and process optimization [47,141]. The estimator can be used to estimate the unknown parameters that are required in the plant model, referred to as parameter estimation. If the parameters remain unchanged throughout the operating window, they can be estimated offline. For example, a Gaussian process regression model with the radial basis function kernel was used by Bishnoi et al. [142] for offline parameter estimation of nine different properties for glass compositions using 37 different chemical components that can be used to form the glass composition. Online parameter estimation is required when the parameters are dependent on the current state of the process. In reference [143], the surface tension parameter for pure compounds, using a single layer feed-forward neural network, was estimated. The model was capable of estimating the surface tension for 752 different pure compounds at various temperatures using the information about the chemical functional groups present in the pure compound.

Machine-learned estimators can also be used to estimate the unmeasured states such as concentrations, heat flux, reaction rates, growth kinetics, battery life cycle [3,45,144]. For state estimation, the estimator is trained offline and applied online due to their dependency on the current plant state. The estimator's internal parameters can be updated by applying additional training using online data to improve the accuracy [45]. For instance, a RNN was used by Nikolaou and Hanagandi [145] to model a nonisothermal continuously stirred tank reactor (CSTR) that acted as an open-loop observer where the process states were fed back to a linear controller. Local stability of the observer and the controller were obtained. The proposed nonlinear controller had a better performance compared to an optimally tuned linear controller [145]. Another example is given by Psichogios and Ungar [70], where hybrid modeling was used to estimate the states of a fed-batch bioreactor. A neural network was used to model the growth rate kinetics that was used in the first-principle model to estimate the states of the bioreactor at the next sampling time. Training of the machine learning

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

model's internal parameters required the use of sensitivity equations that were derived from the objective function. The same method was used by Georgieva and de Azevedo [42] and Galvanauskas et al. [146] on a crystallization process where the machine learning model was used to model the kinetic parameters required in the first-principle models to estimate the online process states. A model predictive controller was further applied by Georgieva and de Azevedo [42] to control the process using the hybrid state estimator, which is discussed in Subsection 6.4.2.2.

6.4.1.3 Fault detection supported by machine learning

Industrial sensors can become faulty due to sensor drift, blockages, or damage to the sensor resulting in problems for accurate process control [20,140]. A small malfunction can have a large effect on a manufacturing process that can be prevented by early detection [1,47]. Hence, fault detection and diagnosis are important components in plant monitoring. They can ensure safe process operation, reliability, product quality, and decrease the costs from production loss [1,47,138]. Often, fault detection models are used to classify an unmeasured fault based on the measured variables of the process [1]. These fault classification models can be based purely on first principles, data, or a hybrid combination of the two [1].

In hybrid and data-based fault detection, the training dataset requires classification and labeling of the fault which can be challenging as some faults might be undiagnosed [147]. Labeling the data requires manual analysis, which can increase model cost and development time [38]. Data can be generated using an accurate plant model or plant simulation for normal as well as faulty conditions [138,147]. The internal parameters of the machine learning model can be updated using online plant data, if required [138]. A general guide to fault detector development for chemical processes as well as various applications are presented by Md Nor et al. [1], where issues regarding high-dimensionality, nonlinearity, complexity, and fault types are discussed.

Shohei [147] used a convolutional neural network to predict faults in a heating, ventilation, and air conditioning (HVAC) system by transforming the data into an image space. The convolutional neural network was trained using simulated data and evaluated on plant data. The model was able to accurately detect the faults on the plant data. This training approach can save model development and data cost while ensuring a high model accuracy. Gaussian process regression has been applied to fault detection problems while providing the advantages of only requiring a relatively small dataset, approximating the system uncertainty, and providing confidence intervals

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

for its predictions [138]. A Gaussian process regression model was developed by Fezai et al. [138] for the benchmark Tennessee Eastman chemical process with the objective to reduce the false alarm rate, missed detection rate, and model computational time.

6.4.1.4 Outlook: use of machine-learned models for process monitoring

Even though the discussed approaches for data-based monitoring of manufacturing systems are promising, some open challenges remain to be solved. The increased scale and complexity of industrial chemical processes result in high dimensional datasets that contain some irrelevant and redundant features which adds complexity and reduces the performance [1]. Machine learning which is capable of feature extraction without any additional knowledge should be favored [1]. Machine learning-based estimators can also be complex to implement, maintain, and to troubleshoot. Hence, simpler designs are required along with more tests on pilot and industrial plants to persuade industry to implement these techniques [45]. Also, soft sensor performance can degrade over time due to the time-varying characteristics of industrial processes [47]. Adaptive training methods that are capable of online learning and calibration to tackle this issue require further investigation [1,45,47]. Furthermore, inaccuracies in laboratory data caused by uncertain time delays, varying sampling intervals, and sampling habits should be taken into account in order to improve the quality of the training data [47]. Hence, robust approaches should be developed. At the same time, focus should be given to provide better explainable and interpretable models by incorporating physics and domain knowledge [16,45].

In particular for manufacturing systems, machine learning methods capable of detecting multiple faults occurring in the same time window should be researched to account for process interactions with other systems [1]. More research into modeling transitional processes with machine learning, such as start-ups or shutdowns, where faults are most likely to occur is required [1].

Finally, the incorporation of estimators with process control strategies should also be investigated to improve the overall estimation and control performance in real time [45].

6.4.2 Learning models for process control

Process control is used to drive the process to a reference state in a safe and efficient way by manipulating the inputs to the process [33]. Traditionally, the

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

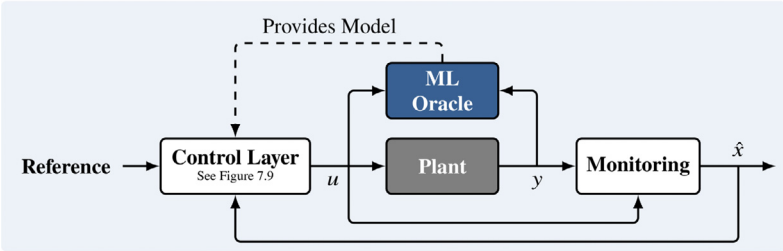


Figure 6.8 Learning system models for process control.

control structure can be organized in a hierarchy, where the process control activities are grouped into different layers operating in different time scales and with different goals (cf. Fig. 6.3).

Model-based control methods can be used in the lower control layer (e.g., linear quadratic regulators), in the supervisory control layer (e.g., model predictive control), and in the upper control layer (e.g., real-time optimization). These layers uses plant models of different granularity (e.g., static models in RTO and dynamic models in MPC) directly in the design of the controller. The model accuracy is important since it greatly influences the controller performance. Obtaining an accurate plant model can be difficult for industrial processes due to nonlinearities and changes in the process characteristics [140]. Machine learning can be used to increase the accuracy of plant models using process data, as discussed in Subsection 6.2.1. This subsection discusses the applications of using machine-learned plant models in model-based control methods such as adaptive control, model predictive control, and inverse model-based control [148,149]. Hence, we do not train a machine learning algorithm to replace or approximate an existing controller (see Section 6.5), but utilize machine learning for process modeling and supply the gathered information to the model-based control techniques (cf. Fig. 6.8).

6.4.2.1 Adaptive control using machine learning plant models

Controller parameters can require adaptation throughout the plant operation to compensate for changes in the process dynamics, process specifications, or in the environment while achieving and maintaining the desired performance [33,150].

These changes are, for example, heat exchanger fouling or catalyst deactivation, frequent or large disturbances in feed quality or composition, changes in product specifications such as quality changes, and inherent

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

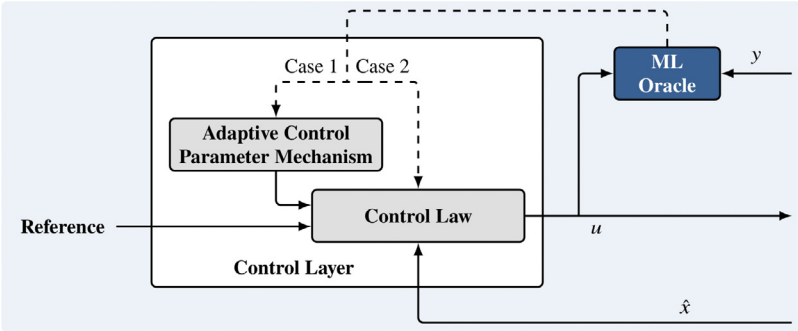


Figure 6.9 Application of the learned system model for process control.

nonlinear behavior [33,57]. Adaptive control can be used to directly adapt the control parameters when the process deviate from the desired behavior. A model-based control design can also be used to adapt the model parameters online which in turn are used to adapt the control law (indirect adaptive control).

Indirect adaptive control is also referred to as self-tuning control. The controller parameters are indirectly adapted online using updated plant models [33,148,150]. Here, the ML–Oracle is used to design and adapt the plant model online. Based on these online learned/adapted models, different adjustments of controllers can be performed. For example, the design/adaptation of PID control gains is possible based on machine learning models (cf. Fig. 6.9, Case 1). Also, the updated model can be used in model-based controller such as LQR or MPC to obtain optimal inputs (cf. Fig. 6.9, Case 2). In principle, also the parameters of an LQR or MPC (such as the weights in the cost function) could be adapted based on machine learning models regardless of the update of the prediction model.

Parlos et al. [151] used an indirect adaptive control structure to control a steam ML–Oracle. A RNN was used to generate the plant model. A weight adaption law was used to adjust the weights of the RNN, using new observations as they became available, in order to enhance the predictive capabilities of the plant model. A gain adaption law was used to adjust the PID controller gains online to ensure sufficient controller performance over the entire operating region. The advantage of this method is that it can be implemented in the industry using the existing control hardware, thereby requiring little capital investment. Applications of neural networks in adaptive control for (bio)chemical processes are summarized by Mears et al. [57] and Hussain [148].

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

6.4.2.2 Model predictive control based on machine learning models

Model predictive control (MPC) is a model-based optimal control method that repeatedly solves an optimal control problem. It uses the model of the plant to find an optimal input trajectory by predicting the system into the future. Two main advantages are: it can be used for multi-inputs multioutputs systems and it can guarantee satisfaction of input and state constraints [29,33,40,140,152]. For instance, MPC is widely used in distillation and reaction processes in (bio)chemical industries as it allows to optimize performance while operating close to the process limits [27,57,140,148]. However, the performance of the MPC control method depends on the accuracy of the plant model [33,40]. Hence, modeling and model maintenance are critical to prevent deterioration and to keep MPC controllers operational [153]. Solving the optimization problem using complex plant models can be computationally demanding and has been a challenge for industrial implementation. The plant model should be able to describe the true process unknowns to prevent unrealistic state predictions but should also be simple enough to find the solution of the control problem online [153].

For this reason, often linear plant models are used in MPC in industrial applications as this simplifies the model development as well as online optimization [32]. However, nonlinear model predictive control would allow to operate the system in wider operating ranges, as it takes the nonlinearities of the underlying processes into account [148]. This is especially important in batch processes and dynamically operated plants with changing process conditions. Still, Kano and Ogawa [140] found that nonlinear MPC has not been widely implemented due to difficulties in developing nonlinear process models.

The development of such nonlinear models can be simplified using machine learning. Machine learning has been successfully used to build the plant model required by MPC, see Table 6.3 and [33]. Here, the ML–Oracle is used to obtain the plant model Eqs. (6.1) for MPC. This plant model, which is in general nonlinear, is used inside the MPC for prediction (cf. Fig. 6.9, Case 2) [148]. Thus, the machine learning model enters the optimization problem as an equality constraint similar to classical optimal control approaches [148]. In contrast to that, Section 6.5 tackles the case when the MPC feedback law is approximated using machine leaning techniques, for example, to improve computational speed especially for large dimensional problems [88].

The performance of a machine-learned plant model in MPC was, for example, studied by Macmurray and Himmelblau [154] who compared the

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

MPC performance using a simplified first-principle model and a RNN model on a simulated packed distillation column. The RNN plant model was trained offline using data generated from a complex first-principle dynamic model. Similar performances between the two methods have been obtained by Macmurray and Himmelblau [154], concluding that machine learning techniques are adequate to develop dynamic plant models for complex or unknown processes with sufficient performances.

Another example is crystallization processes, which have complex nucleation, particle growth, and agglomeration mechanisms [42]. These mechanisms can be modeled using machine learning methods combined with first-principle models to obtain accurate hybrid plant models, as discussed in subsection. These hybrid crystallization plant models can then be further used in model-based control such as MPC, as demonstrated by Georgieva and de Azevedo [42] on a sugar crystallization process. The difference between using a first-principle model and a hybrid plant model in a Lyapunov-based MPC for a simulated continuously stirred tank reactor was investigated by Zhang et al. [89]. A neural network model was used to model the nonlinear reaction rate in the hybrid plant model. Similar performances between the first principles and hybrid models were obtained. However, the hybrid model in practice is more effective due to the challenge of determining the reaction rate of a process based on first principles. Different applications of neural networks with MPC for chemical processes are summarized by Hussain [148] and for biochemical processes by Mowbray et al. [27] and Mears et al. [57]. Robust and stochastic MPC methods for uncertainty systems have been reviewed by Saltık et al. [153].

6.4.2.3 Inverse control using machine learning plant models

Inverse model-based methods directly model the inverse process dynamics to predict the desired control actions over a prediction horizon to drive the process to the desired reference state (cf. Fig. 6.9, Case 2) [155]. The control performance is dependent on the accuracy of these inverse models. Machine learning can be used to obtain the inverse models and relies on the amount and accuracy of the data available to train [155,156]. Direct inverse control and internal model control are two common types of inverse control methods [148,156,157]. The inverse model acts as the controller that is in cascade with the system under control. If the inverse model counteracts the system dynamics perfectly, no feedback is needed and the output of the system perfectly follows the desired reference. However, additional feedback controllers such as PID controllers are often combined with direct inverse

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

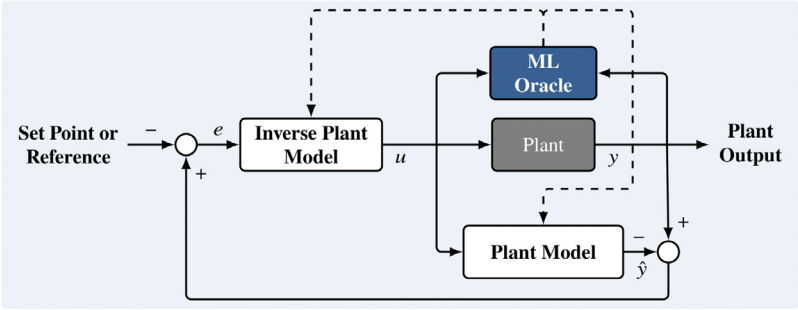


Figure 6.10 Internal model control structure.

control. The desired outputs (set points) along with the past and current process inputs and outputs are then fed into the inverse model that predicts the appropriate control inputs [148,157].

A direct inverse control structure, using neural networks, has been combined with a conventional proportional-integral (PI) controller in [155] to control the concentration and pH in a steel pickling process. The PI controller was used to lower set point offsets and prevent oscillations of the control variables. The proposed combined strategy was compared with their respective individual methods for a nominal, disturbance, and model mismatch case. The PI controller failed to reach the set point in the desired time for both the disturbance and model mismatch case while the direct inverse control method was capable of stabilizing the process with a set point offset for all three cases. The combined strategy was capable of tighter control with no set point offset. Lee et al. [149] developed a RNN to model the inverse dynamics of the large-scale pulverized coal power plant. A PID feedback controller was used to eliminate the steady-state error due to model mismatch and disturbances. They were able to shorten the stabilization time for the process and eliminate the steady-state error (set point offset). Hybrid modeling was applied by Andrášik et al. [158] with a direct inverse model-based strategy. The different error components for a PID controller were used as inputs to the neural network along with the states. Better regulatory and tracking performances were achieved with the proposed method compared to normal direct inverse control.

Internal model control adds a plant model to the direct inverse control strategy to improve the robustness of the controller (cf. Fig. 6.10) [148,155,157,159]. The plant model accounts for mismatches and irregularities that can occur due to equipment degradation and noise [148]. The inverse model acts as the controller with the plant model minimizing the

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

set point offset, even in the presence of noise and disturbances [157,159]. Stability, offset-free tracking, and a parameter-free controller are the three major advantages using internal model-based controllers [159]. The inverse model can be obtained by either inverting the plant model numerically or training the model to identify the inverse [156]. Internal model control is capable to allow trade-offs between performance and robustness unlike direct inverse control [33].

A neural network was used in an internal model-based control strategy by Hosen et al. [159]. The set point tracking and disturbance rejection ability was improved by combining it with prediction interval based modeling to add tighter upper and lower bounds to the predictions. A neural network was used to develop a direct inverse control and an internal model control strategy by Ramli et al. [157] for a simulated debutanizer distillation column. The internal model control strategy had a better performance with faster settling times compared to a conventional PID controller and the direct inverse control strategy. Different applications of inverse model-based applications using neural networks are summarized by Hussain [148].

6.4.2.4 Outlook: machine learning-based models for process control

To enable safe, reliable, and precise process control based on machine learning models, one of the main topics is availability of high-quality models. This quality refers to the accuracy of the machine learning model predictions. Besides, the future of process modeling needs to meet a variety of requirements, including explainability/interpretability, ability to handle uncertainty, and ability to use large quantities of data [1]. To increase profit and flexibility in operation, focus on adaptive mechanisms for prediction models under frequent operating condition deviations is required. Online learning during control execution can be used to do so. However, safety guarantees for these online adaptations still need to be ensured. Robust adaptive controllers or controllers that exploit models of the uncertainties seem promising to tackle these challenges (see also Subsection 6.5.1).

6.5 Control via machine learning

In the previous section, we showed some possible applications of machine learning models for estimation, prediction, fault diagnosis, and model-based control. These mainly focus on modeling the plant using machine learning techniques cf. Figs. 6.7 and 6.8. In this section, we show that machine learning can also be applied in the control layer of the manufacturing

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

process, which can be used to either improve the control performance of the main baseline controller (e.g., an MPC for controlling with a GP for learning the uncertainty), or to replace the baseline controller completely by acting as the main controller of the system. In Subsection 6.4.2, the role of machine learning focused on the plant model, while the control task is achieved by other algorithms based on the model. In this section, machine learning directly builds or adjusts the control algorithm. We discuss the adaptation of controllers via machine learning to guarantee safety and constraint satisfaction despite uncertainties as well as the substitution of controllers via machine learning.

6.5.1 Learning uncertainties for safe control

There exist different uncertainties that can affect the control performance of manufacturing systems. Uncertainties are present in plant models due to the challenge of accurately modeling real plants. Moreover, measurement noise and unmodeled external disturbances can limit the control performance. Significant uncertainties can lead to violations of constraints which, if critical, can cause a fatal failure for the complete system. We focus on robust control approaches that use constraints back-off. This approach is equipped with a representation of the model uncertainty and can provide a certain degree of safety and performance to the plant. Although uncertainty can affect all the components of the control loop, most commonly robust approaches considers model and measurement uncertainties. A common approach in constrained control is to build a *nominal* model (uncertainty free) and then back-off the constraints to such an extent that, if the *nominal* model does not violate the shrunk constraints, the real (uncertain) system will not violate the original constraints. Gaussian processes can be used to model the uncertainty, since they naturally provide not only the learned function but also its probability distribution (cf. Subsection 6.3.3.3). This information can be used to back-off the constraints and guarantee constraint satisfaction up to a certain probability. In references [44,160], this approach was used for lutein production from microalgae. In reference [137], a Monte Carlo sampling strategy for uncertainty propagation together with a power series expansion to represent the confidence interval of the process constraints was used for constraint back-off in a water treatment plant example. In reference [161], a stochastic MPC using GPs was used to back-off the constraints of a batch reactor. The previous approach provides probabilistic constraint satisfaction; hence, it cannot provide constraint satisfaction guarantees for *all* uncertainty

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

realizations. For this purpose, set-based approach can be used. These assume that the uncertainty lies with uniform probability within a finite set and therefore can provide constraint satisfaction guarantees for any uncertainty realization in that set. Some of these approaches use tube-based MPC, where a so-called *ancillary* feedback controller is used to maintain the system uncertainty in a (possibly small) bounded set and hence to avoid expansion with the prediction horizon [117,131,162]. Nevertheless, these approaches tend to be conservative, because they consider also uncertainty events with very low probability. Other approaches do not consider constraint back-off, but discourage the controller to visit areas where the uncertainty is large, for example, by defining an uncertainty measure that enters the objective function [49,53,163]. These are, in general, easier to implement in contrast to the constraint back-off method, but do not guarantee robust constraint satisfaction.

6.5.2 Substitute conventional controllers with machine learning

Machine learning methods can act as the controller in the closed loop system. To design the machine learning controllers, imitation learning, reinforcement learning, or iterative learning control can be used, which we outline in the following.

6.5.2.1 Imitation learning with neural networks

In imitation learning, there exist a baseline controller, which can deliver a desired performance, that is learned and substituted by a machine learning-based controller (Fig. 6.11). This approach can have economical and computational benefits when the learned controller requires less expensive hardware and lower computational power. For example, if a model predictive controller is used, a nonlinear optimization problem needs to be solved at each time instant, which is computationally challenging. For linear systems, explicit MPC can help to reduce the computational burden by dividing the state-space into different regions, precomputing, and storing the control law as an explicit function for each region [164,165]. This approach allows rapid and efficient evaluation without solving the optimization problem at each time step. Nevertheless, it suffers storage problems because of the requirement to store all the precomputed control laws.

A popular approach to solve both problems (fast online computations and large storage demands) is to use a neural network to learn the baseline MPC.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

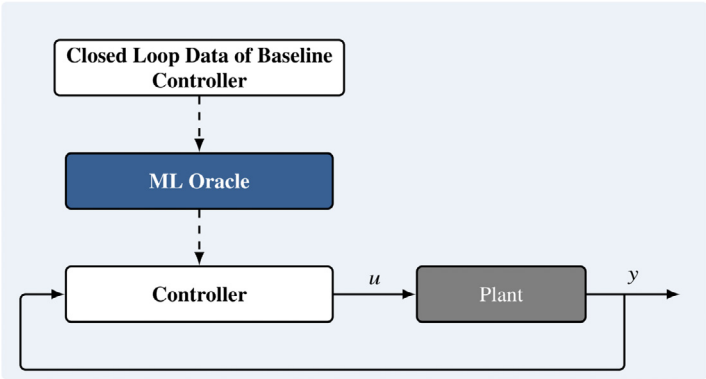


Figure 6.11 In imitation learning, the neural network, acting as the controller, is trained from input–output data of a baseline controller.

There are numerous papers that exploit this approach, for example [166–172]. Other machine learning methods can also be applied. For example, support vector machines were used to avoid the problem of storing the explicit MPC for high-dimensional systems [173]. The control inputs and closed-loop data, consisting of the states or outputs of the real or the simulated plant, are generated using the baseline controller. The data are then used to train the machine learning-based controller offline.

If we put this into the framework we have mentioned in Section 6.2, the ML–Oracle uses the states or measurements as features and the input as labels to learn an explicit control law. Hence, this approach belongs to the category of supervised learning. Since the baseline controller needs to be synthesized, at least the nominal mathematical model of the plant is often assumed to be known, which can provide useful information to guarantee safety for the closed-loop system with the machine learning-based control.

In general, guaranteeing safety (stability and constraint guarantees) for closed-loop systems with machine learning components is challenging. However, several approaches have been investigated to guarantee stability for closed-loop systems using neural networks. The first approach is to exploit the theorems by Cybenko [174] and Hornik [175] about the universal approximation ability of multilayer feed-forward neural networks. They have shown that any continuous mapping over a compact domain can be approximated as accurately as necessary by a feed-forward neural network with one hidden layer. In other words, given any (small) positive number, a neural network with sufficiently large number of nodes can guarantee that the difference between the ground truth function and the approximated

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

function is always smaller than that number for all points in the compact domain. This theorem is combined with small gain theorem to guarantee stability of the closed-loop system in reference [176]. The second approach exploits the characteristics of the nonlinear activation functions commonly used in deep neural networks. Since these functions often satisfy the sector bound conditions or can be transformed by using loop transform techniques so that the sector bound conditions are satisfied, the stability of the closed-loop system can be reformulated into the form of a *diagonal linear differential inclusion*, which can be guaranteed by available tools from robust control theory such as integral quadratic constraint (IQC) [177] or absolute stability of Lure systems [178]. This approach also allows analyzing the robustness of the system under disturbance and uncertainty.

The work by Hertneck et al. [179] uses a different approach to obtain the probabilistic guarantees based on Hoeffding's Inequality, where the neural network is used to learn a robust MPC subject to inaccurate inputs within given bounds. The neural network is used to learn this controller via offline samples. Finally, the approximated neural network controller is validated statistically by using Hoeffding's Inequality to obtain a bound on error between the baseline MPC and the learned one. More recently, by extending the work of Hertneck et al. [179], the authors in reference [88] propose a new neural network architecture to obtain offset-free closed-loop performance for industrial large-scale linear MPC, which are challenging for online QP solvers. Furthermore, the aim of this paper also focuses on set point tracking MPC, which is relevant to the manufacturing industry. The input-to-state of the system is guaranteed by using the available results in the MPC literature. The condition under which the neural network controllers are robust to state estimation errors and process disturbances is also established. Some works obtain the guarantee by exploiting a special structure of neural network to impose stability, for example [180,181].

6.5.2.2 Reinforcement learning

Reinforcement learning (RL) is a data-driven (possibly) model-free control approach, where the mathematical model of the plant is not required and it does not necessarily needs a baseline controller either (Fig. 6.12), cf. [183,184] for process control related reviews.

RL allows to find "optimal" controllers for nonlinear stochastic systems when the dynamics are unknown or affected by significant uncertainty. Although RL has a strong connection with optimal control, there exist differences in the terminology used in the artificial intelligence community and

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

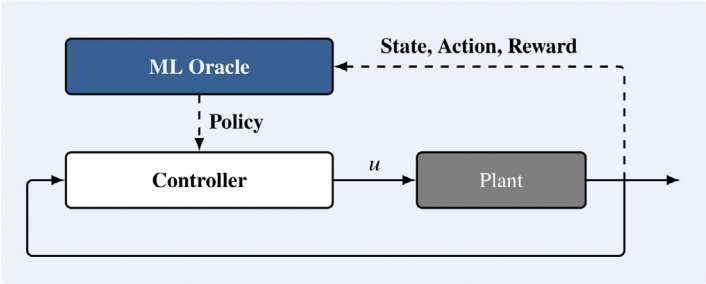


Figure 6.12 In reinforcement learning, by using information of states, actions and rewards, the machine learning (ML)–Oracle helps establish an optimal policy.

Table 6.4 Some terminology commonly used in reinforcement learning (RL), and their control counterparts [182].

Terms in RL	Control-related terms
Environment	System
Agent	Decision maker or controller
Action	Decision or control
Reward of a stage	(Opposite of) Cost of a stage
Value (or reward) function	(Opposite of) Cost function
Deep reinforcement learning	Approximate dynamic programming using value and/or policy approximation with deep neural networks
Planning	Solving a dynamic programming problem with a known mathematical model
Learning	Solving a dynamic programming problem without using an explicit mathematical model

the optimal control community [182]. Table 6.4 shows some terminology commonly used in RL and their control counterparts.

If we use the framework from Section 6.2, we can explain that in RL, the ML–Oracle learns, either offline or online, from samples of transitions and rewards to solve an optimal control problem and obtain a state feedback control law, called policy in RL literature. The ML–Oracle plays the role of the controller in the closed-loop system. In order to solve the optimal control problem, the ML–Oracle has to learn either the optimal value (or action-value) function, the optimal control policy, or both (actor–critic methods) which are impossible to be exactly derived in general. Therefore, functional approximation must be utilized. A very common type of function approximators, which is extensively used at the moment, is deep neural

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

network (DNN). The resulting method is known as deep reinforcement learning (DRL).

There are review/tutorial papers (e.g., see references [185–187]) which draw the connections between RL and two popular control methods: MPC and adaptive/approximate dynamic programming (ADP). In reference [188], a reinforcement learning approach was used for batch bio-processes and compared to NMPC, where extensive offline training was carried out using a simplified model.

Safety for RL, which currently is one of the main drawbacks, is still an open question because stability conditions and constraints cannot be guaranteed naturally during the problem formulation, compared to approaches based on first principles. For example, the constraints and stability conditions are integrated naturally during formulation of the MPC problem. However, this is harder to be integrated for RL during the process of solving for optimal policy. The basic problem for RL lies in the fact that RL is based on a finite sampling dataset, and without the absolute knowledge of the system, it is impossible to guarantee robust control satisfaction without introducing additional assumptions. Therefore, there are several works that have combined the advantages of MPC in dealing with stability and safety with RL to achieve some degree of safety (cf. [24]). One possible combination is proposed by Zanon and Gros [189], where MPC is used as a function approximator within RL to provide safety and stability guarantees. Also RL is also utilized to tune the MPC parameters, thus improving closed-loop performance in a data-driven fashion. In order to guarantee stability and safety for RL, Zhang et al. [190] propose Q-learning-based MPC for online control of nonlinear systems where an accurate mathematical model is not available. In this approach, two neural networks are implemented with an actor–critic structure. The critic network approximates the Q-function, trained by system input and state measurements, while the actor network is used to approximate a Lyapunov-based MPC to reduce time for solving optimization problem. The stability and safety is guaranteed by the Lyapunov's second method for stability and constraints integrated in solving the MPC problem. Alhazmi et al. [191] used RL to learn unknown parameters of chemical processes but an economic MPC is used to guarantee stability.

Several methods that aim at using RL while guaranteeing constraint satisfaction use a *safety filter*, that is, a backup controller (e.g., MPC) that intervenes in place of the RL when constraint violation cannot be guaranteed [192,193]. Wabersich [222] used an MPC as a *filter* that is entitled to change

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

the RL input as little as possible in such a way that constraints are satisfied. Besides the approach of combining MPC with RL for safety guarantees, other theoretical tools in control systems theory are also used, for example, Lyapunov design principles by Perkins and Barto [194], or robust control by Jin and Lavaei [195]. Furthermore, the work of Pan et al. [196] proposes a constrained Q-learning algorithm which can guarantee safety with high probability by using self-tuned constraint tightening.

6.5.2.3 Iterative learning control

Iterative learning control (ILC) is a control technique with the objective to improve the performance of processes that are executed repeatedly as the number of repetitions increases [197]. In (bio)chemical engineering, these processes can be, for example, (fed)batch processes where the performance is usually measured in terms of a tracking error of a possible time-varying reference [198].

These processes have a fixed operating time, the same initial conditions, are operated repetitively (cf. Subsection 6.2.2). Trajectory changes, disturbances, and initialization errors can also occur in these processes to a certain degree [198]. Iterative learning control has been developed for more than 50 years; hence, a large number of variations have been proposed. We only outline the main idea (refer to the review articles [197–201] for a more detailed discussion).

The idea is to learn a so-called *filter* that reduces the tracking error $(e_{ki})_{i\in\mathbb{I}}$, with $e_{ki} := r_i - y_{k_i}$, between a reference $(r_i)_{i\in\mathbb{I}}$ and the measured plant output sequence $\left(y_{k_i}\right)_{i\in\mathbb{I}}$ at every iteration k over the set $\mathbb{I} := \{0, 1, \ldots, N\}$ of sample times. This means that after each run of the process, the ML–Oracle obtains the measured output sequence as well as the input sequence $(u_{ki})_{i\in\mathbb{I}}$ that generated it. The idea behind the algorithm of the ML–Oracle is to update the sequence $(u_{ki})_{i\in\mathrm{I}}$ such that $e_{ki} \to 0$ for all sample times $i \in \mathbb{I}$ as $k \to \infty$ (cf. Fig. 6.13). Using a time–discrete dynamical system in a lifted-system representation (cf. [218,219]), the update algorithm within the ML–Oracle can also be written explicitly in general form by

$$u_{k+1} = Q(u_k + Le_k)$$

where, $Q \in \mathbb{R}^{N,N}$ and $L \in \mathbb{R}^{N,N}$ are called the *Q-filter* and the *gain matrix*. These matrices depend not only on the transfer behavior of the plant but also on the ILC tuning techniques used [197]. The open-loop control sequence $(u_{ki})_{i\in\mathrm{I}}$ is applied to the plant. Often, ILC is coupled with a feedback controller to reject nonrepeating disturbances (cf. e.g., [202]).

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

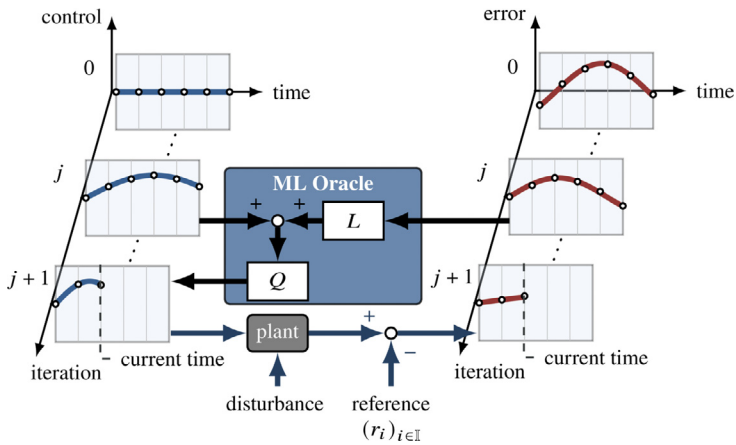


Figure 6.13 Depiction of an iterative learning controller for a time–discrete dynamical system. The matrices L and Q are the called learning function and Q-filter [197].

This basic formulation was adapted to a model-based formulation that uses model plant inversion in [207,216].

ILC is usually applied to linear or linearized models in discrete time, but extensions to continuous-time, nonlinear systems are available (cf. [115,201,203]). Asymptotic stability can be guaranteed with some conditions on the filter matrix and for some cases also in presence of non-repeating perturbations [204]. For repetitive processes, whose trajectories, dynamics, and disturbances do not change much between iterations, ILC can be a valid control strategy to use in combination with a feedback control strategy. Nonrepeating disturbances and noise can be detrimental for the performance of the ILC. For this, it is recommended to couple the ILC control algorithm with a closed-loop controller.

6.5.3 Outlook

Model-based control methods require a model of the system which are often derived from first principles. The benefit of applying ML for control, especially for the RL case, is that there is no need to build a model of the system, which is often expensive, timely, and requires a lot of effort. It also means that we do not need prior expert knowledge of the systems if we use a completely model-free or data-driven approach. However, the drawback of the approach, which is based completely on ML, is that it lacks safety guarantees, robustness analysis, and constraint satisfaction. Therefore, the future of the research in this direction is to combine both approaches so that we can benefit from the advantages of both approaches.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

6.6 Conclusion

The control of (bio)chemical processes is vital in order to obtain high-quality products and ensure safety, but challenging due to complexity, high-dimensionality, strong nonlinearity, and uncertainties. To deal with these challenges, the traditional control scheme is often build on mathematical models that are based on first-principle knowledge and using model-based control systems. However, building mathematical models is often difficult, time-consuming, and economically expensive.

In the last decade, there has been a strong, growing interest in machine learning in the (bio)chemical industry due to an increasing amount of available data as well as the breakthroughs in deep learning and reinforcement learning. These techniques are promising, as they can accelerate and facilitate the modeling and control of complex systems such as (bio)chemical processes. Nevertheless, there are several existing bottlenecks that prevent machine learning from being widely applied in this field, including safety requirements and lack of good-quality data.

This chapter has outlined how machine learning can be integrated inside the control systems to enhance the performance. We proposed a generic and unified framework for integrating machine learning techniques with closed-loop control systems which is applicable to many machine learning techniques. In this framework, we characterize the machine learning components as an ML–Oracle block, which can be described as an abstract map that uses a dataset to construct a continuous function to provide a correlation between the feature set and the label set. This abstract concept allows us to describe many machine learning techniques in a unified way.

With this framework, we have reviewed and classified machine learning methods for (bio)chemical systems from two main perspectives. The first perspective discusses machine learning methods that can be used to derive the models of the (bio)chemical plant for identification, analysis, estimation, and monitoring tasks. The obtained model can then be used for simulation-, optimization-, and model-based controller design; that is, the control design tasks are done with conventional control theory after obtaining the model from machine learning. This part covers many machine learning techniques that have been used in the control field for decades as well as recent techniques such as Gaussian process.

The second perspective discusses machine learning methods that can be used directly in control algorithms to support the control tasks partially or to replace the controllers completely. This part covers more recent techniques, such as imitation learning using neural networks or reinforcement

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Table 6.5 Selected, non-extensive, overview of works on machine-learningsupported control. Note that there is no clear-cut classification, as in some cases it is difficult to classify the methods into these categories.

	Controller Design supported by Machine Learning	**Controller Design via Machine Learning**
Real-Time Optimization	• learning modifiers of the modifier adaptation scheme via GPs [205,206,208] • hybrid modeling for RTO models [40]	• solving static optimization of RTO using RL [220]
Supervisory Control	• plant models for model-based control [42,57,89,148,153,154,160] • inverse models to provide control actions [148,149,157,159]	• set-point optimization using RL [211,213] • imitation (supervise) learning [176] • RL-based [188,191]
Regulatory Control	• observer for parameter and state estimation [42,70,142,143,145,146] • self-tuning PID controllers [151,155,158,209,214,215,217]	• RL for tuning PID controller [212]

learning. Table 6.5 summarizes a selected, nonextensive, overview of works on machine learning- supported control, which have been mentioned in this chapter, in the context of the common hierarchical control structure in modern chemical manufacturing processes. With regards to machine learning in the context of RTO, some work has proposed to learn the modifiers of a modifier adaptation scheme [63,205,206]. The technique tries to reach the real plant optimum despite model uncertainties, which is achieved by modifying the objective function and constraints of Eq. (6.3) in such a way that the necessary conditions of optimality *of the real plant* are obtained.

We did not limit ourselves in reviewing only machine learning applications of (bio)chemical processes, but we also mentioned methods that have been used in different fields. For example, to the best of our knowledge, limited work has been done on safety guarantees for machine learning-supported control and estimation for (bio)chemical processes. For this reasons, we reviewed approaches that have been developed in other fields, such as robotics. The goal is to bring awareness to the (bio)chemical community about these methods that can be used as inspiration for new (bio)chemical applications.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

We also see several research topics as essential for future developments: The main bottleneck to machine learning-supported approaches in (bio)chemical processes is data. For this reason, we believe that extensive research must be done to develop new measurement technologies. Furthermore, to gain the trust of the practitioners, future research should consider the development of robust modeling and control approaches, in other words, approaches that guarantee margins of safety, for example, on product specifications and or process constraints.

References

[1] N. Md Nor, C.R. Che Hassan, M.A. Hussain, A review of data-driven fault detection and diagnosis methods: applications in chemical process systems, Rev. Chem. Eng. 36 (4) (2020) 513–553.
[2] J. Panerati, et al., Experimental methods in chemical engineering: artificial neural networks–ANNs, Can. J. Chem. Eng. 97 (9) (2019) 2372–2382.
[3] K. Severson, et al., Data-driven prediction of battery cycle life before capacity degradation, Nat. Energy 4 (5) (2019) 383–391.
[4] S. Zendehboudi, N. Rezaei, A. Lohi, Applications of hybrid models in chemical, petroleum, and energy systems: a systematic review, Appl. Energy, 228, 2018, pp. 2539–2566.
[5] K. Severson, P. Chaiwatanodom, R. Braatz, Perspectives on process monitoring of industrial systems, Annu. Rev. Control 42 (2016) 190–200.
[6] D.M. Himmelblau, Accounts of experiences in the application of artificial neural networks in chemical engineering, Ind. Eng. Chem. Res. 47 (16) (2008) 5782–5796.
[7] J.A. Paulson, et al., Fast stochastic model predictive control of end-to-end continuous pharmaceutical manufacturing, Comput. Aided Chem. Eng. 41 (2018) 353–378.
[8] S. Lucia, et al., Predictive control, embedded cyber physical systems and systems of systems: a perspective, Annu. Rev. Control 41 (2016) 193–207.
[9] T. Binder, et al., Introduction to model based optimization of chemical processes on moving horizons, in: Online Optimization of Large Scale Systems, Springer, 2001, pp. 295–339.
[10] J. Kober, J.A. Bagnell, J. Peters, Reinforcement learning in robotics: a survey, Int. J. Rob. Res. 32 (11) (2013) 1238–1274.
[11] S. Grigorescu, et al., A survey of deep learning techniques for autonomous driving, J. Field Rob. 37 (3) (2020) 362–386.
[12] M. Raissi, A. Yazdani, G.E. Karniadakis, Hidden fluid mechanics: a Navier-Stokes informed deep learning framework for assimilating flow visualization data, arXiv (2018).
[13] F. Zantalis, et al., A review of machine learning and IoT in smart transportation, Future Internet 11 (4) (2019) 94.
[14] A.M. Schweidtmann, et al., Machine learning in chemical engineering: a perspective, Chem. Ing. Tech. 93 (12) (2021) 2029–2039.
[15] J. Lee, J. Shin, M. Realff, Machine learning: overview of the recent progresses and implications for the process systems engineering field, Comput. Chem. Eng. 114 (2018) 111–121.
[16] V. Venkatasubramanian, The promise of artificial intelligence in chemical engineering: is it here, finally? AlChE J. 65 (2) (2019) 466–478.
[17] Y. Liu, et al., Materials discovery and design using machine learning, J. Mater. 3 (3) (2017) 159–177.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[18] T. Zhou, Z. Song, K. Sundmacher, Big Data creates new opportunities for materials research: a review on methods and applications of machine learning for materials design, Engineering 5 (6) (2019) 1017–1026.
[19] S.J. Qin, Process data analytics in the era of Big Data, AlChE J. 60 (9) (2014) 3092–3100.
[20] L.H. Chiang, E.L. Russell, R.D. Braatz, Fault Detection and Diagnosis in Industrial Systems, Springer Science & Business Media, 2000.
[21] G.E. Karniadakis, et al., Physics-informed machine learning, Nat. Rev. Phys. 3 (6) (2021) 422–440.
[22] M. Von Stosch, et al., Hybrid semi-parametric modeling in process systems engineering: past, present and future, Comput. Chem. Eng. 60 (2014) 86–101.
[23] A. Mesbah, et al., Fusion of machine learning and MPC under uncertainty: what advances are on the horizon? in: 2022 American Control Conference (ACC), IEEE, 2022, pp. 342–357.
[24] L. Hewing, et al., Learning-based model predictive control: toward safe learning in control, Annu. Rev. Control Robot. Auton. Syst. 3 (1) (2020) 269–296.
[25] Z. Ge, et al., Data mining and analytics in the process industry: the role of machine learning, IEEE Access 5 (2017) 20590–20616.
[26] M.R. Dobbelaere, et al., Machine learning in chemical engineering: strengths, weaknesses, opportunities, and threats, Engineering 7 (9) (2021) 1201–1211, doi:10.1016/j.eng.2021.03.019.
[27] M. Mowbray, et al., Machine learning for biochemical engineering: a review, Biochem. Eng. J. 172 (2021) 108054.
[28] D. Mayne, et al., Constrained model predictive control: stability and optimality, Automatica 36 (6) (2000) 789–814.
[29] R. Findeisen, F. Allgöwer, An introduction to nonlinear model predictive control, in: 21st Benelux Meeting on Systems and Control, 11, Citeseer, 2002, pp. 119–141.
[30] L. Grüne, J. Pannek, Nonlinear model predictive control, in: Non-Linear Model Predictive Control, Springer, 2017, pp. 45–69.
[31] J. Matschek, et al., Constrained Gaussian process learning for model predictive control, IFAC-PapersOnLine 53 (2) (2020) 971–976.
[32] C. Pantelides, J. Renfro, The online use of first-principles models in process operations: review, current status and future needs, Comput. Chem. Eng. 51 (2013) 136–148.
[33] D.E. Seborg, et al., Process Dynamics and Control, third ed, Wiley, 2011, p. 464.
[34] R. Subramanian, R.R. Moar, S. Singh, White-box machine learning approaches to identify governing equations for overall dynamics of manufacturing systems: a case study on distillation column, Mach. Learn. Appl. 3 (2021) 100014.
[35] B.W. Bequette, Process Control: Modeling, Design, and Simulation, Prentice Hall Professional, 2003.
[36] G. Stephanopoulos, Chemical Process Control, 2, Prentice Hall, 1984.
[37] M. Pirdashti, et al., Artificial neural networks: applications in chemical engineering, Rev. Chem. Eng. 29 (4) (2013) 205–239.
[38] J. Willard, et al., Integrating physics-based modeling with machine learning: a survey, CoRR 2003 (04919) (2020) 271–278.
[39] N. Bhutani, G.P. Rangaiah, A.K. Ray, First-principles, data-based, and hybrid modeling and optimization of an industrial hydrocracking unit, Ind. Eng. Chem. Res. 45 (23) (2006) 7807–7816.
[40] D. Zhang, et al., Hybrid physics-based and data-driven modeling for bioprocess online simulation and optimization, Biotechnol. Bioeng. 116 (11) (2019) 2919–2930.
[41] I. Basheer, M. Hajmeer, Artificial neural networks: fundamentals, computing, design, and application, J. Microbiol. Methods 43 (1) (2000) 3–31.
[42] P. Georgieva, S.F. de Azevedo, Neural network-based control strategies applied to a fed-batch crystallization process, Int. J. Chem. Biol. Eng. 1 (12) (2007) 145–154.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[43] S. Curteanu, F. Leon, Hybrid neural network models applied to a free radical polymerization process, Polym. Plast. Technol. Eng. 45 (9) (2006) 1013–1023.
[44] E. Bradford, et al., Dynamic modeling and optimization of sustainable algal production with uncertainty using multivariate Gaussian processes, Comput. Chem. Eng. 118 (2018) 143–158.
[45] J.Mohd Ali, et al., Artificial intelligence techniques applied as estimator in chemical process systems: a literature survey, Expert Syst. Appl. 42 (14) (2015) 5915–5931.
[46] R. Oliveira, Combining first principles modelling and artificial neural networks: a general framework, Comput. Chem. Eng. 28 (5) (2004) 755–766.
[47] C. Shang, F. You, Data analytics and machine learning for smart process manufacturing: recent advances and perspectives in the Big Data era, Engineering 5 (6) (2019) 1010–1016.
[48] G. Karniadakis, et al., Physics-informed machine learning, Nat. Rev. Phys. 3 (2021) 422–440.
[49] B. Morabito, et al., Towards risk-aware machine learning supported model predictive control and open-loop optimization for repetitive processes, IFAC-PapersOnLine 54 (6) (2021) 321–328.
[50] A. Teixeira, et al., Hybrid semi-parametric mathematical systems: bridging the gap between systems biology and process engineering, J. Biotechnol. 132 (4) (2007) 418–425.
[51] F. García-Camacho, et al., Artificial neural network modeling for predicting the growth of the microalga Karlodinium veneficum, Algal Res. 14 (2016) 58–64.
[52] E.A. del Rio-Chanona, et al., Dynamic modeling and optimization of cyanobacterial C-phycocyanin production process by artificial neural network, Algal Res. 13 (2016) 7–15.
[53] A.P. Teixeira, et al., Bioprocess iterative batch-to-batch optimization based on hybrid parametric/nonparametric models, Biotechnol. Progr. 22 (1) (2006) 247–258.
[54] D. Henriques, et al., Data-driven reverse engineering of signaling pathways using ensembles of dynamic models, PLoS Comput. Biol. 13 (2) (2017) 1–25, doi:10.1371/journal.pcbi.1005379.
[55] D. Lee, A. Jayaraman, J. Kwon, Development of a hybrid model for a partially known intracellular signaling pathway through correction term estimation and neural network modeling, PLoS Comput. Biol. 16 (12) (2020) 1–31, doi:10.1371/journal.pcbi.1008472.
[56] D. Lee, A. Jayaraman, J.-I. Kwon, A hybrid mechanistic data-driven approach for modeling uncertain intracellular signaling pathways, in: Proceedings of the American Control Conference, 2021, 2021, pp. 1903–1908.
[57] L. Mears, et al., A review of control strategies for manipulating the feed rate in fed-batch fermentation processes, J. Biotechnol. 245 (2017) 34–46.
[58] D. Bonvin, Optimal operation of batch reactors: a personal view, J. Process Control 8 (5-6) (1998) 355–368.
[59] J.H. Lee, K.S. Lee, Iterative learning control applied to batch processes: an overview, Control Eng. Pract. 15 (10) (2007) 1306–1318.
[60] M. Darby, et al., RTO: an overview and assessment of current practice, J. Process Control 21 (6) (2011) 874–884.
[61] S. Qin, T. Badgwell, A survey of industrial model predictive control technology, Control Eng. Pract. 11 (7) (2003) 733–764.
[62] S. Skogestad, I. Postlethwaite, Multivariable Feedback Control: Analysis and Design, John Wiley & Sons, Inc., 2005.
[63] A. Marchetti, B. Chachuat, D. Bonvin, Modifier-adaptation methodology for real-time optimization, Ind. Eng. Chem. Res. 48 (13) (2009) 6022–6033.
[64] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning. Adaptive Computation and Machine Learning, MIT Press, 2006.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[65] F. Allgöwer, R. Findeisen, Z.K. Nagy, Nonlinear model predictive control: from theory to application, J. Chin. Inst. Chem. Eng. 35 (3) (2004) 299–316.
[66] P. Santos, et al., Improving operation in an industrial MDF flash dryer through physics-based NMPC, Control Eng. Pract. 94 (2020) 1–15, doi:10.1016/j.conengprac.2019.104213.
[67] W.-S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (4) (1943) 115–133.
[68] A. Kusiak, Convolutional and generative adversarial neural networks in manufacturing, Int. J. Prod. Res. 58 (5) (2020) 1594–1604.
[69] F.-Y. Zhou, L.-P. Jin, J. Dong, Review of convolutional neural network, Chin. J. Comput. 40 (6) (2017) 1229–1251.
[70] D.C. Psichogios, L.H. Ungar, A hybrid neural network-first principles approach to process modeling, AlChE J. 38 (10) (1992) 1499–1511.
[71] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[72] S. Skansi, Introduction to Deep Learning From Logical Calculus to Artificial Intelligence, Springer, 2018.
[73] J. Chung, et al., Empirical evaluation of gated recurrent neural networks on sequence modeling (2014). arXiv preprint arXiv:1412.3555.
[74] A. Sherstinsky, Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network, Phys. D 404 (2020) 132306.
[75] D. Samek, P. Dostal, MPC using adaline, in: Annals of DAAAM and Proceedings of the International DAAAM Symposium, 2005, pp. 335–336.
[76] A. Embaby, Z. Nossair, H. Badr, Adaptive nonlinear model predictive control algorithm for blood glucose regulation in type 1 diabetic patients, in: 2nd Novel Intelligent and Leading Emerging Sciences Conference, NILES 2020, 2020, pp. 109–115.
[77] A. Kheirabadi, R. Nagamune. Real-time relocation of floating offshore wind turbines for power maximization using distributed economic model predictive control. Am. Control Conf. (ACC) Proceeding In: vol. 2021, 2021, pp. 3077–3081. doi:10.23919/ACC50511.2021.9483056.
[78] S. Chen, Z. Wu, and P. Christofides. Cyber-security of decentralized and distributed control architectures with machine-learning detectors for nonlinear processes, in: Proceedings of the American Control Conference. 2021, pp. 3273–3280.
[79] Z. Shao, et al., An internal model controller for three-phase APF based on LS-extreme learning machine, Open Electr. Electron. Eng. J. 8 (2014) 717–722.
[80] D. Sarali, V. Agnes Idhaya Selvi, K. Pandiyan, An improved design for neural-network-based model predictive control of three-phase inverters, in: 2019 International Conference on Clean Energy and Energy Eflcient Electronics Circuit for Sustainable Development, INCCES 2019, 2019.
[81] T. Van Den Boom, M. Botto, P. Hoekstra, Design of an analytic constrained predictive controller using neural networks, Int. J. Syst. Sci. 36 (10) (2005) 639–650.
[82] T. Varshney, R. Varshney, S. Sheel, ANN based IMC scheme for CSTR, in: Proceedings of the International Conference on Advances in Computing, Communication and Control, ICAC3'09, 2009, pp. 543–546.
[83] F. Mjalli, Adaptive and predictive control of liquid-liquid extractors using neural-based instantaneous linearization technique, Chem. Eng. Technol. 29 (5) (2006) 539–549.
[84] J. Ou, R. Rhinehart, Grouped-neural network modeling for model predictive control, ISA Trans. 41 (2) (2002) 195–202.
[85] O. Dahunsi, J. Pedro, Neural network-based identification and approximate predictive control of a servo-hydraulic vehicle suspension system, Eng. Lett 18 (4) (2010) 1–15.
[86] S. Mohanty, Artificial neural network based system identification and model predictive control of a flotation column, J. Process Control 19 (6) (2009) 991–999.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[87] J. Bethge, et al., Modelling human driving behavior for constrained model predictive control in mixed traffic at intersections, IFAC-PapersOnLine 53 (2) (2020) 14356–14362.
[88] P. Kumar, J. Rawlings, S. Wright, Industrial, large-scale model predictive control with structured neural networks, Comput. Chem. Eng. 150 (2021) 107291.
[89] Z. Zhang, et al., Real-time optimization and control of nonlinear processes using machine learning, Mathematics 7 (10) (2019) 890.
[90] Z. Yan, X. Le, J. Wang, Tube-based robust model predictive control of nonlinear systems via collective neurodynamic optimization, IEEE Trans. Ind. Electron. 63 (7) (2016) 4377–4386.
[91] L. Zhang, M. Pan, S. Quan, Model predictive control of water management in PEMFC, J. Power Sources 180 (1) (2008) 322–329.
[92] K. Temeng, P. Schnelle, T. McAvoy, Model predictive control of an industrial packed bed reactor using neural networks, J. Process Control 5 (1) (1995) 19–27.
[93] K. Zarkogianni, et al., An insulin infusion advisory system based on autotuning nonlinear model-predictive control, IEEE Trans. Biomed. Eng. 58 (9) (2011) 2467–2477.
[94] K. Patan, Neural network-based model predictive control: fault tolerance and stability, IEEE Trans. Control Syst. Technol. 23 (3) (2015) 1147–1155.
[95] K. Dalamagkidis, K. Valavanis, L. Piegl, Nonlinear model predictive control with neural network optimization for autonomous autorotation of small unmanned helicopters, IEEE Trans. Control Syst. Technol. 19 (4) (2011) 818–831.
[96] T. Thuruthel, et al., Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators, IEEE Trans. Rob. 35 (1) (2019) 127–134.
[97] Z. Yan, J. Wang, Model predictive control of nonlinear systems with unmodeled dynamics based on feedforward and recurrent neural networks, IEEE Trans. Ind. Inf. 8 (4) (2012) 746–756.
[98] R. Al Seyab, Y. Cao, Nonlinear system identification for predictive control using continuous time recurrent neural networks and automatic differentiation, J. Process Control 18 (6) (2008) 568–581.
[99] P. Kittisupakorn, et al., Neural network based model predictive control for a steel pickling process, J. Process Control 19 (4) (2009) 579–590.
[100] C.H. Lu, C.C. Tsai, Adaptive predictive control with recurrent neural network for industrial processes: an application to temperature control of a variable-frequency oil-cooling machine, IEEE Trans. Ind. Electron. 55 (3) (2008) 1366–1375.
[101] Z. Yan, J. Wang, Model predictive control for tracking of underactuated vessels based on recurrent neural networks, IEEE J. Ocean. Eng. 37 (4) (2012) 717–726.
[102] Y. Pan, J. Wang, Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks, IEEE Trans. Ind. Electron. 59 (8) (2012) 3089–3101.
[103] L. Zhang, et al., Model predictive control for electrochemical impedance spectroscopy measurement of fuel cells based on neural network optimization, IEEE Trans. Transp. Electrif. 5 (2) (2019) 524–534.
[104] H. Huang, L. Chen, E. Hu, A hybrid model predictive control scheme for energy and cost savings in commercial buildings: simulation and experiment, in: Proceedings of the American Control Conference 2015-July, 2015, pp. 256–261.
[105] D. Pereira, F. Lopes, E. Watanabe, Nonlinear model predictive control for the energy management of fuel cell hybrid electric vehicles in real time, IEEE Trans. Ind. Electron. 68 (4) (2021) 3213–3223.
[106] S. Yang, et al., Experiment study of machine-learning-based approximate model predictive control for energy-efficient building control, Appl. Energy 288 (2021) 1–12, doi:10.1016/j.apenergy.2021.116648.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[107] Y. Pan, J. Wang, A neurodynamic optimization approach to nonlinear model predictive control, in: Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics, 2010, pp. 1597–1602.
[108] J. Atuonwu, et al., Identification and predictive control of a multistage evaporator, Control Eng. Pract. 18 (12) (2010) 1418–1428.
[109] Z. Wang, T. Hong, M. Piette, Predicting plug loads with occupant count data through a deep learning approach, Energy 181 (2019) 29–42.
[110] F. Núñez, et al., Neural network-based model predictive control of a paste thickener over an industrial internet platform, IEEE Trans. Ind. Inf. 16 (4) (2020) 2859–2867.
[111] Y. Wu, et al., A predictive energy management strategy for multi-mode plug-in hybrid electric vehicles based on multi neural networks, Energy 208 (2020) 1–17, doi:10.1016/j.energy.2020.118366.
[112] Z. Wu, D. Rincon, P. Christofides, Real-time adaptive machine- learning-based predictive control of nonlinear processes, Ind. Eng. Chem. Res. 59 (6) (2020) 2275–2290.
[113] Z. Wu, D. Rincon, P. Christofides, Process structure-based recurrent neural network modeling for model predictive control of nonlinear processes, J. Process Control 89 (2020) 74–84.
[114] Y. Pan, J. Wang, Model predictive control for nonlinear affine systems based on the simplified dual neural network, in: Proceedings of the IEEE International Conference on Control Applications, 2009, pp. 683–688.
[115] Y.Q. Chen, C. Wen, Iterative Learning Control: Convergence, Robustness and Applications, Springer, London, 1999.
[116] Y. Pan, J. Wang, Two neural network approaches to model predictive control, in: Proceedings of the American Control Conference, 2008, pp. 1685–1690.
[117] J. Bethge, et al., Multi-mode learning supported model predictive control with guarantees, IFAC-PapersOnLine 51 (20) (2018) 517–522.
[118] J. Kocijan, et al., Predictive control with Gaussian process models, in: Eurocon 2003: The International Conference on Computer as a Tool, 1, IEEE, 2003, pp. 352–356.
[119] J. Kocijan, et al., Gaussian process model based predictive control, in: Proceedings of the 2004 American Control Conference, 3, IEEE, 2004, pp. 2214–2219.
[120] R. Murray-Smith, et al., Adaptive, cautious, predictive control with Gaussian process priors, in: IFAC Proceedings Volumes, 36, 2003, pp. 1155–1160.
[121] B. Likar, J. Kocijan, Predictive control of a gas–liquid separation plant based on a Gaussian process model, Comput. Chem. Eng. 31 (3) (2007) 142–152.
[122] A. Grancharova, J. Kocijan, T.A. Johansen, Explicit stochastic nonlinear predictive control based on Gaussian process models, in: European Control Conference (ECC), IEEE, 2007, pp. 2340–2347.
[123] A. Grancharova, J. Kocijan, T.A. Johansen, Explicit stochastic predictive control of combustion plants based on Gaussian process models, Automatica 44 (6) (2008) 1621–1631.
[124] G. Cao, E. Lai, F. Alam, Gaussian process model predictive control of unknown nonlinear systems, IET Control Theory Appl. 11 (5) (2017) 703–713.
[125] G. Cao, E. Lai, F. Alam, Gaussian process based model predictive control for linear time varying systems, in: 14th International Workshop on Advanced Motion Control (AMC), IEEE, 2016, pp. 251–256.
[126] T.X. Nghiem, C.N. Jones, Data-driven demand response modeling and control of buildings with Gaussian processes, in: American Control Conference (ACC), IEEE, 2017, pp. 2919–2924.
[127] M. Maiworm, et al., Stability of Gaussian process learning based output feedback model predictive control, in: 6th IFAC Conference on Nonlinear Model Predictive Control, 2018, pp. 551–557.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[128] M. Maiworm, D. Limon, R. Findeisen, Online Gaussian process learning-based model predictive control with stability guarantees, Int. J. Robust Nonlinear Control 31 (18) (2021) 8785–8812, doi:10.1002/rnc.5361.
[129] L. Hewing, M.N. Zeilinger, Cautious model predictive control using Gaussian process regression, arXiv (2017).
[130] X. Yang, J.M. Maciejowski, Fault tolerant control using Gaussian processes and model predictive control, Int. J. Appl. Math. Comput. Sci. 25 (1) (2015) 133–148.
[131] R. Soloperto, et al., Learning-based robust model predictive control with state-dependent uncertainty, IFAC-PapersOnLine 51 (20) (2018) 442–447.
[132] E. Bradford, et al., Hybrid Gaussian process modeling applied to economic stochastic model predictive control of batch processes, in: Recent Advances in Model Predictive Control, Springer, 2021, pp. 191–218.
[133] J. Caldwell, J.A. Marshall, Towards efficient learning-based model predictive control via feedback linearization and Gaussian process regression, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2021, pp. 4306–4311.
[134] F. Li, H. Li, Y. He, Adaptive stochastic model predictive control of linear systems using Gaussian process regression, IET Control Theory Appl. 15 (5) (2021) 683–693.
[135] J. Sansana, et al., Recent trends on hybrid modeling for Industry 4.0, Comput. Chem. Eng. 151 (2021) 107365.
[136] C. Nentwich, J. Winz, S. Engell, Surrogate modeling of fugacity coefficients using adaptive sampling, Ind. Eng. Chem. Res. 58 (40) (2019) 18703–18716.
[137] M. Rafiei, L.A. Ricardez-Sandoval, Stochastic back-off approach for integration of design and control under uncertainty, Ind. Eng. Chem. Res. 57 (12) (2018) 4351–4365.
[138] R. Fezai, et al., Online reduced Gaussian process regression based generalized likelihood ratio test for fault detection, J. Process Control 85 (2020) 30–40.
[139] T. Chai, S.J. Qin, H. Wang, Optimal operational control for complex industrial processes, Annu. Rev. Control 38 (1) (2014) 81–92.
[140] M. Kano, M. Ogawa, The state of the art in advanced chemical process control in Japan, IFAC Proc. 42 (11) (2009) 10–25.
[141] Z. Ge, T. Chen, Z. Song, Quality prediction for polypropylene production process based on CLGPR model, Control Eng. Pract. 19 (5) (2011) 423–432.
[142] S. Bishnoi, et al., Scalable Gaussian processes for predicting the optical, physical, thermal, and mechanical properties of inorganic glasses with large datasets, Mater. Adv. 2 (1) (2021) 477–487.
[143] F. Gharagheizi, et al., Use of artificial neural network-group contribution method to determine surface tension of pure compounds, J. Chem. Eng. Data 56 (5) (2011) 2587–2601.
[144] P. Attia, et al., Closed-loop optimization of fast-charging protocols for batteries with machine learning, Nature 578 (7795) (2020) 397–402.
[145] M. Nikolaou, V. Hanagandi, Control of nonlinear dynamical systems modeled by recurrent neural networks, AlChE J. 39 (11) (1993) 1890–1894.
[146] V. Galvanauskas, P. Georgieva, S. de Azevedo, Dynamic optimisation of industrial sugar crystallization process based on a hybrid (mechanistic + ANN) model, in: The 2006 IEEE International Joint Conference on Neural Network Proceedings, IEEE, 2006, pp. 2728–2735.
[147] M. Shohei, et al., Fault detection and diagnosis for heat source system using convolutional neural network with imaged faulty behavior data, HVACR Res. 26 (1) (2020) 52–60.
[148] M.A. Hussain, Review of the applications of neural networks in chemical process control: Simulation and online implementation, Artif. Intell. Eng. 13 (1) (1999) 55–68.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[149] K.Y. Lee, et al., Inverse dynamic neuro-controller for superheater steam temperature control of a large-scale ultra-supercritical (USC) boiler unit, IFAC Proc. Vol. (IFAC-PapersOnLine) 42 (9) (2009) 107–112.
[150] I. Landau, Controls, adaptive systems, in: Encyclopedia of Physical Science and Technology, Elsevier, 2003, pp. 649–658.
[151] A.G. Parlos, S. Parthasarathy, A.F. Atiya, Neuro-predictive process control using on-line controller adaptation, IEEE Trans. Control Syst. Technol. 9 (5) (2001) 741–755.
[152] J. Rawlings, D. Mayne, M. Diehl, Model Predictive Control: Theory, Computation, and Design (2018).
[153] M.B. Saltık, et al., An outlook on robust model predictive control algorithms: reflections on performance and computational aspects, J. Process Control 61 (2018) 77–102.
[154] J.C. Macmurray, D.M. Himmelblau, Modeling and control of a packed distillation column using artificial neural networks, Comput. Chem. Eng. 19 (10) (1995) 1077–1088.
[155] P. Thitiyasook, P. Kittisupakorn, M.A. Hussain, Dual-mode control with neural network based inverse model for a steel pickling process, Asia Pac. J. Chem. Eng. 2 (6) (2007) 536–543.
[156] M.A. Hussain, J.Mohd Ali, M.J. Khan, Neural network inverse model control strategy: discrete-time stability analysis for relative order two systems, Abstr. Appl. Anal. 2014 (2014) 1–11, doi:10.1155/2014/645982.
[157] N.M. Ramli, M.A. Hussain, B.M. Jan, Multivariable control of a debutanizer column using equation based artificial neural network model inverse control strategies, Neurocomputing 194 (2016) 135–150.
[158] A. Andrášik, A. Mészáros, S.F. De Azevedo, On-line tuning of a neural PID controller based on plant hybrid modeling, Comput. Chem. Eng. 28 (8) (2004) 1499–1509.
[159] M.A. Hosen, et al., NN-based prediction interval for nonlinear processes controller, Int. J. Control Autom. Syst. 19 (9) (2021) 3239–3252.
[160] M. Mowbray, et al., Safe chance constrained reinforcement learning for batch process control, arXiv (2021).
[161] E. Bradford, et al., Stochastic data-driven model predictive control using Gaussian processes, Comput. Chem. Eng. 139 (2020).
[162] A. Aswani, et al., Provably safe and robust learning-based model predictive control, Automatica 49 (5) (2013) 1216–1226.
[163] X. Yang, J. Maciejowski, Risk-sensitive model predictive control with Gaussian process models, IFAC-PapersOnLine 48 (28) (2015) 374–379.
[164] A. Alessio, A. Bemporad, A survey on explicit model predictive control, Nonlinear Model Predictive Control, Springer, 2009, pp. 345–369.
[165] A. Bemporad, F. Borrelli, M. Morari, et al., Model predictive control based on linear programming — the explicit solution, IEEE Trans. Autom. Control 47 (12) (2002) 1974–1985.
[166] T. Parisini, R. Zoppoli, A receding-horizon regulator for nonlinear systems and a neural approximation, Automatica 31 (10) (1995) 1443–1451.
[167] B. Karg, S. Lucia, Efficient representation and approximation of model predictive control laws via deep learning, IEEE Trans. Cybern. 50 (9) (2020) 3866–3878.
[168] E. Maddalena, et al., A neural network architecture to learn explicit MPC controllers from data, IFAC-PapersOnLine 53 (2) (2020) 11362–11367.
[169] Y. Cao, R.B. Gopaluni, Deep neural network approximation of nonlinear model predictive control, IFAC-PapersOnLine 53 (2) (2020) 11319–11324.
[170] S. Chen, et al., Approximating explicit model predictive control using constrained neural networks, in: 2018 Annual American Control Conference (ACC), 2018, pp. 1520–1527.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[171] L.H. Csekő, M. Kvasnica, B. Lantos, Explicit MPC-based RBF neural network controller design with discrete-time actual Kalman filter for semiactive suspension, IEEE Trans. Control Syst. Technol. 23 (5) (2015) 1736–1753.
[172] S.S. Pon Kumar, et al., A deep learning architecture for predictive control, IFAC-PapersOnLine 51 (18) (2018) 512–517.
[173] A. Chakrabarty, et al., Support vector machine informed explicit non-linear model predictive control using low-discrepancy sequences, IEEE Trans. Autom. Control 62 (1) (2017) 135–148.
[174] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Syst. 2 (1989) 303–314.
[175] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Netw. 4 (2) (1991) 251–257.
[176] B. Åkesson, et al., Neural network approximation of a nonlinear model predictive controller applied to a pH neutralization process, Comput. Chem. Eng. 29 (2) (2005) 323–335.
[177] H. Yin, P. Seiler, M. Arcak, Stability analysis using quadratic constraints for systems with neural network controllers, IEEE Trans. Autom. Control 67 (4) (2021) 1980–1987, doi:10.1109/TAC.2021.3069388.
[178] H.H. Nguyen, et al., Stability certificates for neural network learning-based controllers using robust control theory, in: 2021 American Control Conference (ACC), 2021, pp. 3564–3569.
[179] M. Hertneck, et al., Learning an approximate model predictive controller with guarantees, IEEE Control Syst. Lett. 2 (3) (2018) 543–548.
[180] R. Ivanov, et al., Verisig: verifying safety properties of hybrid systems with neural network controllers, arXiv (2018).
[181] H. Nguyen, et al., Towards nominal stability certification of deep learning-based controllers, in: 2020 American Control Conference (ACC), 2020, pp. 3886–3891.
[182] D.P. Bertsekas, Reinforcement Learning and Optimal Control, Athena Scientific, 2019.
[183] J. Shin, et al., Reinforcement learning: overview of recent progress and implications for process control, Comput. Chem. Eng. 127 (2019) 282–294.
[184] R. Nian, J. Liu, B. Huang, A review on reinforcement learning: introduction and applications in industrial process control, Comput. Chem. Eng. 139 (2020) 106886.
[185] D.P. Bertsekas, Dynamic programming and suboptimal control: a survey from ADP to MPC, Eur. J. Control 11 (4) (2005) 310–334, doi:10.3166/ejc.11.310-334.
[186] D. Görges, Relations between model predictive control and reinforcement learning, IFAC-PapersOnLine 50 (1) (2017) 4920–4928.
[187] F.L. Lewis, D. Vrabie, Reinforcement learning and adaptive dynamic programming for feedback control, IEEE Circuits Syst. Mag. 9 (3) (2009) 32–50.
[188] P. Petsagkourakis, et al., Reinforcement learning for batch bioprocess optimization, Comput. Chem. Eng. 133 (2020) 106649.
[189] M. Zanon, S. Gros, Safe reinforcement learning using robust MPC, IEEE Trans. Autom. Control 66 (8) (2021) 3638–3652.
[190] H. Zhang, S. Li, Y. Zheng, Q-learning-based model predictive control for nonlinear continuous-time systems, Ind. Eng. Chem. Res. 59 (40) (2020) 17987–17999.
[191] K. Alhazmi, F. Albalawi, S. Sarathy, A reinforcement learning-based economic model predictive control framework for autonomous operation of chemical reactors, Chem. Eng. J. 428 (2022) 130993.
[192] K.P. Wabersich, M.N. Zeilinger, Scalable synthesis of safety certificates from data with application to learning-based control, in: 2018 European Control Conf. (ECC), 2018, pp. 1691–1697.
[193] S. Muntwiler, et al., Distributed model predictive safety certification for learning-based control, IFAC-PapersOnLine 53 (2) (2020) 5258–5265.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[194] T.J. Perkins, A.G. Barto, Lyapunov design for safe reinforcement learning, J. Mach. Learn. Res. 3 (2002) 803–832.
[195] M. Jin, J. Lavaei, Stability-certified reinforcement learning: a control-theoretic perspective, IEEE Access 8 (2020) 229086–229100.
[196] E. Pan, et al., Constrained model-free reinforcement learning for process optimization, Comput. Chem. Eng. 154 (2021) 1–10, doi:10.1016/j.compchemeng.2021.107462.
[197] D.A. Bristow, M. Tharayil, A.G. Alleyne, A survey of iterative learning control, IEEE Control Syst. Mag. 26 (3) (2006) 96–114.
[198] J.H. Lee, K.S. Lee, Iterative learning control applied to batch processes: an overview, Control Eng. Pract. 15 (10) (2007) 1306–1318.
[199] K.L. Moore, M. Dahleh, S. Bhattacharyya, Iterative learning control: a survey and new results, J. Robot. Syst. 9 (5) (1992) 563–594.
[200] H.-S. Ahn, Y.Q. Chen, K.L. Moore, Iterative learning control: brief survey and categorization, IEEE Trans. Syst. Man Cybern. C 37 (6) (2007) 1099–1121.
[201] J.-X. Xu, A survey on iterative learning control for nonlinear systems, Int. J. Control 84 (7) (2011) 1275–1294.
[202] T.-Y. Doh, Robust iterative learning control with current feedback for uncertain linear systems, Int. J. Syst. Sci. 30 (1) (1999) 39–47.
[203] J.-X. Xu, Y. Tan, Linear and Nonlinear Iterative Learning Control, 291, Springer, 2003.
[204] M. Norrlöf, S. Gunnarsson, Time and frequency domain convergence properties in iterative learning control, Int. J. Control 75 (14) (2002) 1114–1126.
[205] T. de Avila Ferreira, et al., Real-time optimization of uncertain process systems via modifier adaptation and Gaussian processes, in: 2018 European Control Conference (ECC), 2018, pp. 465–470.
[206] E. del Rio Chanona, et al., Modifier-adaptation schemes employing Gaussian processes and trust regions for real-time optimization, IFAC-PapersOnLine 52 (1) (2019) 52–57.
[207] N. Amann, D. Owens, E. Rogers, Iterative learning control for discrete-time systems with exponential rate of convergence, IEE Proc. Control Theory Appl. 143 (2) (1996) 217–224.
[208] L.E. Andersson, L. Imsland, Real-time optimization of wind farms using modifier adaptation and machine learning, Wind Energy Sci. 5 (3) (2020) 885–896.
[209] H.C. Chen, Optimal fuzzy PID controller design of an active magnetic bearing system based on adaptive genetic algorithms 4 (2008) 2054–2060, doi:10.1109/ICMLC.2008.4620744.
[210] Y. Chen, C. Hu, J. Wang, Human-centered trajectory tracking control for autonomous vehicles with driver cut-in behavior prediction, IEEE Trans. Veh. Technol. 68 (9) (2019) 8461–8471.
[211] W. Dai, et al., Multi-rate layered optimal operational control of industrial processes, Acta Autom. Sin. 45 (10) (2019) 1946–1959.
[212] O. Dogru, et al., Reinforcement learning approach to autonomous PID tuning, Comput. Chem. Eng. 161 (2022) 107760.
[213] S. Kim, et al., On-line set-point optimization for intelligent supervisory control and improvement of Q-learning convergence, Control Eng. Pract. 114 (2021) 1–13, doi:10.1016/j.conengprac.2021.104859.
[214] D. Kucherov, et al., PID controller machine learning algorithm applied to the mathematical model of quadrotor lateral motion, in: 2021 IEEE 6th International Conference on Actual Problems of Unmanned Aerial Vehicles Development (APUAVD), 2021, pp. 86–89.
[215] D. Lee, S.J. Lee, S.C. Yim, Reinforcement learning-based adaptive PID controller for DPS, Ocean Eng. 216 (2020) 1–12. https://doi.org/10.1016/j.oceaneng.2020.108053.
[216] J.H. Lee, K.S. Lee, W. Kim, Model-based iterative learning control with a quadratic criterion for time-varying linear systems, Automatica 36 (2000) 641–657.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[217] H. Liu, D. Liu, Self-tuning PID controller for a nonlinear system based on support vector machines, Control Theory Appl. 25 (3) (2008) 468–474.
[218] I. Markovsky, Closed-loop data-driven simulation, Int. J. Control 83 (10) (2010) 2134–2139.
[219] I. Markovsky, F. Dörfler, Data-driven dynamic interpolation and approximation, Automatica 135 (2022) 110008.
[220] B. Powell, D. Machalek, T. Quah, Real-time optimization using reinforcement learning, Comput. Chem. Eng. 143 (2020).
[221] A. Rajkomar, J. Dean, I. Kohane, Machine learning in medicine, N. Engl. J. Med. 380 (14) (2019) 1347–1358.
[222] K. P. Wabersich, M. N. Zeilinger. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. arXiv:1812.05506, (2018).

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

CHAPTER SEVEN

Learning first principles systems knowledge from data: Stability and safety with applications to learning from demonstration

Ashwin Dani and Iman Salehi
Electrical and Computer Engineering, University of Connecticut, Storrs, CT, United States

7.1 Introduction

Cognitive science researchers often study different ways in which individuals learn skills from others. Examples of such learning mechanisms are imitation by infants of simple responses and by toddlers of more complex replications of tasks. Robotics researchers have used these social learning mechanisms from cognitive sciences to develop methods that can be used by robots to acquire new knowledge autonomously and increase their skill set without needing much human intervention. Methods such as imitation learning and learning from demonstrations (LfD) [1–7] are often used for training robots without much human intervention or manual programming. This chapter discusses the applicability of such methods to robotics or autonomous systems used in the context of manufacturing applications. With the advances in machine learning (ML), controls, and artificial intelligence (AI), the applications of social learning mechanisms in humans and animals to robotics research have made substantial progress. Learning by imitation means the robot engages in imitative behavior which reinforces learning. It involves learning and generalization, not just record and playback. In imitation learning, a correspondence problem is solved, i.e., if the robot is imitating by perceiving human actions, the problem is to find a sequence of robot actions that correspond to the actions perceived by humans to achieve a given task. LfD is used when a new task is required to be learned, but the imitative behavior may not be present [8]. In robotics research, imitation learning is used to program a robot easily by showing them how the task can be performed. An example of architecture is shown in Fig. 7.1. Two main questions that are addressed by various researchers to date in imitation

Artificial Intelligence in Manufacturing: Concepts and Methods.
DOI: https://doi.org/10.1016/B978-0-323-99134-6.00001-3

Copyright © 2024 Elsevier Inc.
All rights are reserved, including those for text and data mining, AI training, and similar technologies.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY
Copyright Elsevier 2024

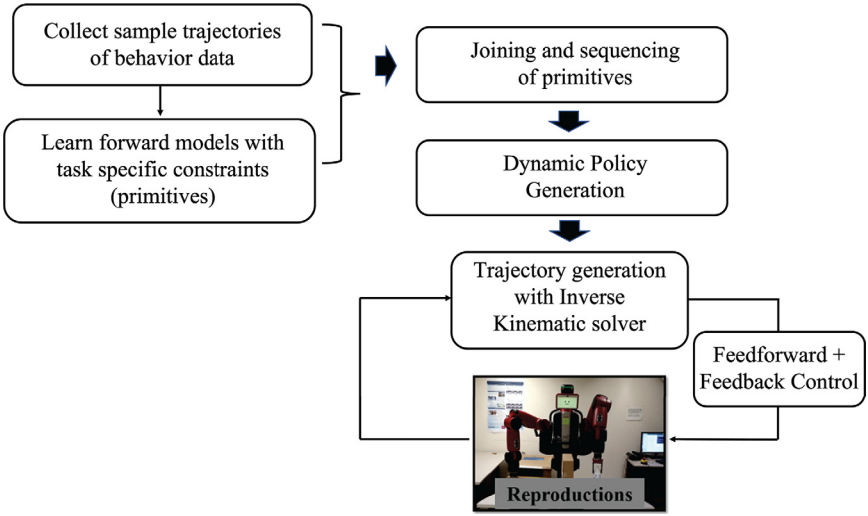


Figure 7.1 An example architecture of movement primitives for robot motion generation.

learning are: how does a robot know, what to imitate through perception and how to imitate?

7.1.1 What to imitate?

The first question is related to the perception of human motion and actions using sensors. For perceiving human motion, motion capture systems are typically used. Advances in AI and computer vision and newer sensors such as RGB-D sensor (Microsoft Kinect) can be used to track human skeletons. In addition to tracking human skeletons, the attention of the person is also important while carrying out the task. Certain attention models are developed to date using cues such as gaze, head pose information, or pointing to a particular object. Deep learning methods are developed to detect human head pose and eye gaze, which can be used to estimate a user's attention.

7.1.2 How to imitate?

The second question is related to mapping the observations of human actions to the robot's motor commands. Converting perceived action in a sequence of robot actions to carry out the task is termed as a correspondence problem [9,10]. Two main methods are used for mapping human actions to robot motions. The first method involves transforming human joint actions from the human reference frame to the robot reference frame and then using

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Figure 7.2 Example of kinesthetic teaching in a human-robot collaborative task.

the converted actions in the robot reference frame to imitate the motion. The second method is to record the human trajectory in task space—e.g., human hand's position tracking—which then can be directly transferred to the robot's actions for carrying out the task [1]. Dynamic system models of the task space motion can be learned as forward models, which can then be combined with a priori knowledge of the system to generate robot motor commands. Methods to learn system models from data that can embed system knowledge of convergence and safety are described and new advances in this category of methods are shown in this chapter.

7.1.3 What are the user interfaces for demonstrations?

In LfD, there are different ways of teaching or demonstrating the task to be carried out by the robot. The two common ways are kinesthetic teaching and observational teaching. In kinesthetic teaching, the robot is manually guided by the human through the task. The robot records its motion while teaching is in process using its proprioceptive sensors. It is also referred to as direct teaching. Since the user demonstrates the skill with the robot's own body, the advantage of this approach is that no explicit physical correspondence is required, which simplifies the correspondence problem. It also provides a simple way to correct the skill acquired by the robot. The drawback of this approach is that the user cannot directly control multiple joints of the robot as it is often difficult to move multiple robot joints simultaneously using two hands. Also, the kinesthetic teaching interface is difficult for teaching tasks that require synchronization of multiple hand motions, e.g., bimanual manipulation tasks or highly dynamic tasks. An example of this method is shown in Fig. 7.2.

In observational teaching, the user is observed using sensors such as vision while the user carries out the task. The full body motion capture (mocap) system or wearable sensor systems (IMU, gyroscopes) are used to track human skeletal motion. The motion of human limb joint angles is recorded

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Figure 7.3 Example of observational teaching in a human-robot collaborative assembly task.

while the user performs the task. This method is useful when the task transfer is limited to kinematic motion teaching. The advantage of this method is that the user can move freely while performing the actions. The disadvantage is that the correspondence of human motions to the robot's motor actions needs to be established accurately to transfer the skill. The correspondence problem becomes challenging when the robot differs physically from the user/demonstrator. It also becomes challenging when more than one user is involved. An example of this method is shown in Fig. 7.3.

The third way of teaching/demonstrating interfaces is immersive teleoperation. This includes using joysticks, commands from graphical user interfaces, or sophisticated devices such as exoskeletons or haptic devices. The method is useful when a human operator is limited to using the robot's own sensors and robot end-effectors to perform the task. Teleoperation performed using joysticks would provide kinematic information such as position, velocity, etc. whereas haptic devices can provide information for tasks that require precise control of forces. Newer advances in immersive technologies such as augmented and virtual reality (AR/VR) are also used for demonstrating a task in virtual environments [11]. An example of this method is shown in Fig. 7.4. The big advantage of this method is that the correspondence problem is solved and the robots can be trained remotely, which is required in many applications where the robot is operating in hazardous and/or remote environments. Acrobatic trajectories of a helicopter are learned using apprenticeship learning by recording the tilt and pan motions demonstrated using teleoperation [12]. A robot dog is taught to play soccer with the operator demonstrating the motions via a joystick [13]. The main disadvantage of the method is that user training is required for using the teleoperation devices, which can be an issue in certain cases, where complex devices such as an exoskeleton needs to be manipulated [14,15] or a helicopter or an unmanned aerial vehicle needs to be operated.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Figure 7.4 Examples of teaching interfaces using HoloLens mixed reality glasses.

In the subsequent paragraphs of the introduction, different technical ways of solving imitation learning or an LfD problem are summarized starting with low-level motor learning using movement primitives to the high-level composition of primitives. Then we will discuss other learning methods that can be used along with imitation learning.

7.1.4 Low-level learning of motions using movement primitives

For successful LfD, a robot must be able to learn from demonstrations generated by a nonexpert for a given task [16–18]. The demonstrations of the task can be broken down into simpler primitive motions, which are referred to as movement primitives (MP). There are several ways to represent the MPs of a user carrying out a complex task. The contributions in the area of dynamic movement primitives (DMP), state space models for representing MPs, and statistical learning approaches such as hidden Markov models (HMM) are summarized below.

Dynamic movement primitives is a deterministic dynamical system representation that contains a linear proportional derivative (PD) like term coupled with a nonlinear term [1,2,19]. Two terms are coupled through a phase variable. The purpose of the PD term is to drive the solutions of the DMP to a stable attractor point (e.g., converge to a position in 3D space) and the purpose of the nonlinear term is to capture the shape of the trajectory in transient (e.g., a specific way of moving from point A to point B). Tasks such as drumming [20], pouring [21], and pick-and-place have been recreated using DMPs on various robotic platforms. The deterministic MP is extended to probabilistic MP in [22]. The DMP formulation models each dimension of motion using one dynamical system, thus, providing a simple, yet efficient,

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

way for learning and storing the primitives. For modeling multidimensional systems, multiple DMPs are learned separately—one for each dimension. This can sometimes neglect the combined effect of all the dimensions. State space models are used for this purpose which will be described next.

The state space representation generally uses nonlinear dynamics or piecewise linear form, which can be approximated using a Gaussian mixture model (GMM) or a neural network (NN) or its variants. In [3], a GMM approximation method is used to derive a stable estimator of dynamical systems (SEDS) learning scheme, which is able to learn globally stable dynamical systems. The convergence property of the goal location is encoded via a Lyapunov stability conditions on the parametric form of the dynamics. The parameters of GMMs are learned subject to the stability constraint to ensure asymptotic stability of the goal location. The SEDS algorithm uses a quadratic Lyapunov function ($V = x^T x$) for computing the stability constraints so it can only model trajectories whose 2-norm distances to the goal decrease monotonically in time. To circumvent this issue, an approach called control Lyapunov function-based dynamic movements (CLF-DM) is developed by Khansari-Zadeh and Billard [23], which uses a weighted sum of asymmetric quadratic functions (WSAQF) to approximate a Lyapunov function whose parameters are learned using state trajectory data. The Lyapunov function is then used to learn the GMM parameters of the dynamical system model that can generate convergent trajectories to the goal location. The CLF-DM is able to learn models that represent a much wider variety of motion trajectories compared to SEDS. Since the CLF-DM uses an additional optimization problem for learning the Lyapunov function, it may potentially lead to numerical stability issues [24]. The τ-SEDS algorithm introduced by Neumann and Steil [24] uses diffeomorphic transformations to transform the state trajectory data such that it is compatible with a quadratic Lyapunov function of the SEDS algorithm. By learning the system model in a transformed space, τ-SEDS is able to encode a much wider class of system trajectories in the system model. The learned model can be back-transformed to the original space of system variables. The Neurally-imprinted Lyapunov candidate (NILC) [25] approach is developed besides many different parametrization approaches to approximate Lyapunov functions. Neurally-imprinted vector fields (NiVF) approach that uses extreme learning machines (ELM) NN to approximate the system model is presented in [26]. The ELM approximation is capable of learning a variety of motions but the stability is restricted to finite regions. LfD method for learning a control policy called task-parametrized GMMs (TpGMM) is developed in

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[27]. Recently, nonlinear model learning methods of system identification of nonlinear systems under contraction constraints are developed in [28,29].

7.1.5 High-level composition of complex tasks

Statistical models such as HMMs are used to represent high-level tasks. Complex behaviors of high-level tasks can be created by an adaptive selection of the MP modules. Task models represented by HMMs are used [30] which are combined with probabilistic road maps (PRMs) for motion planning. For extracting the tasks from demonstrations, a keyframe-based approach [31] uses the clustering of keyframes to extract task information. The problem of autonomous segmentation and abstraction of the continuous movements are also solved in the context of creating and learning high-level tasks.

7.1.6 Combining imitation learning with reinforcement learning and meta-learning

Imitation learning can only train the robot as well as the demonstration data. In tasks such as grasping, precision operations are required when the robot is about to grasp an object. Only imitation learning may not be sufficient in such cases. Reinforcement learning (RL) allows robots to explore and improve the control policy by exploration of the state space and using rewards for policy improvements [32–34]. The drawback of RL is that it requires a lot of data and convergence can be slow. A combination of imitation and RL can be useful in various ways. Imitation learning can provide initial estimates of the policy parameters to RL or the demonstrated data can be used to estimate reward function [35]. Imitation learning can be used to generate an initial set of primitives in conjunction with RL, which then can be used to select motions from these primitives [36]. The objective function or reward is learned from data in a framework called inverse optimal control (IOC) or inverse reinforcement learning (IRL) [37–43]. The estimated objective function is then optimized to compute robot motion trajectories. A comparison between trajectory-based imitation learning methods and optimal control methods is provided by Schaal et al. [44]. Meta-learning can also be combined with imitation learning as an initialization process for representing prior learned models. Finn et al. [45] developed a one-shot visual imitation learning method. In this method, by reusing past experience, the robot is able to learn new skills from a single

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

demonstration. Meta-learning is used as a pretraining procedure to learn past skills from demonstrations collected over a diverse range of environments, then a vision-based policy is fine-tuned using a single demonstration.

7.1.7 Contracting and safe movement primitives

This chapter will summarize our results in the area of MP-based on state space modeling of a system using nonlinear system representation. Specifically, the results are described by Ravichandar et al. [46], Ravichandar and Dani [47], Salehi et al. [48], and Salehi et al. [49]. The presented approaches use GMM and ELM neural network approximations of the functions of differential equations. An unconstrained training of the function approximators cannot guarantee stability or convergence property of the trajectories generated from the models. Thus, if the learned models are used to generate trajectories starting from regions of the state space where the training data was not available, then they may not keep the property of convergence to a target/goal state on which the model was learned. The convergence property of the trajectories is useful in the context of generating reference motions for autonomous systems to follow using a controller. Many autonomous systems or robotic tasks exhibit the property of goal convergence in motion. The first part of the chapter (Section 7.2) presents an algorithm referred to as contracting dynamical systems primitives (CDSP). In this method, a contraction analysis tool [50] for analyzing nonlinear systems is used. The model learning problem is formulated as a constrained parameter learning problem of the approximators such that the state space system model has convergent solutions. The contraction analysis constraints are used, which guarantee that the solutions of the learned model converge to a reference trajectory or a collection of trajectories, thereby keeping a similar shape of the trajectories in the position state and guaranteeing the convergence to the desired position state.

Lyapunov-based LfD approaches use the goal convergence property of the learned model with respect to a particular equilibrium point. On the other hand, the contraction analysis-based approach learns state space system models that encode incremental stability or the convergence of trajectories with respect to each other [50–52]. A system model that is contracting generates system trajectories that retain the shape in the position states for reproducing the demonstrations and its generalization from different initial conditions. If the demonstration data converges to a goal location then the contracting trajectories converge to that particular goal location.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Additionally, robustness under perturbations is achieved and adaptation to a new goal location can also be achieved.

The second method described in this chapter shows our novel contribution of learning point-to-point motion generation that stays within a prescribed region, thereby ensuring the generation of safe trajectories or trajectories that operate in a prescribed confined region. The MP uses barrier functions (BF) and Lyapunov stability tools from nonlinear systems theory. We will term these primitives as safe MPs (SMPs). Given an invariant region in which the primitive motion need to be generated, BF constraints are developed for the nonlinear system approximated as an ELM NN. Similarly, Lyapunov constraints are developed such that the nonlinear system is stable and converges to an equilibrium in the state space. The parameters of ELM are learned using an optimization problem subject to BF and Lyapunov constraints. Using a small set of data, a state space system model using the ELM NN function is learned, which then can be used as SMPs to generate robot motions. The SMPs are easy to store and transfer using only a few parameters of the NN model.

7.2 Learning robot motions using dynamical systems primitive

Two methods of learning task-level movement primitives using the state space modeling approach of nonlinear systems are described in this section. The first approach uses contraction analysis for learning a dynamic system model approximated by GMM. The second approach uses Lyapunov and barrier function analysis for learning dynamic system models approximated by ELM.

7.2.1 System model

Let $x(t) \in \mathbb{R}^n$ be a state variable. Consider the following differential equation model for the state $x(t)$

$$\dot{x}(t) = f(x(t)) \tag{7.1}$$

where, $f : \mathbb{R}^n \mapsto \mathbb{R}^n$ is a continuously differentiable nonlinear function. The objective is to learn the model in Eq. (7.1) in a parametric form from a given set of state and its derivative data that may correspond to a trajectory from one point to another in the state space. The data consists of the trajectories of the states $\{x(t), \dot{x}(t)\}_{t=0:T_n}^{i=1:N}$ with the property that the states converge to

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

a goal state $\{x^*, 0\}$ for all the trajectories, which is an equilibrium point in the state space. The initial state for each demonstration data can be different. The function $f(\cdot)$ is parameterized using GMM and ELM NN, details for which are provided next.

7.2.2 Function approximation

In this section, two function approximation techniques—GMMs and ELM NN—are discussed.

7.2.2.1 Gaussian mixture models

The function $f(\cdot)$ in Eq. (7.1) is approximated using a GMM [3,53], which results in:

$$\dot{x}(t) = \sum_{j=1}^{L} h_j(x(t))\big(A_j x(t) + b_j\big) = f(x(t)) \tag{7.2}$$

where, the scalar weight $h_j(x) \triangleq \frac{p(j)p(x|j)}{\sum_{j=1}^{L} p(j)p(x|j)}$ is associated with the jth Gaussian with the properties $0 \leq h_j(x) \leq 1$ and $\sum_{j=1}^{L} h_j(x) = 1$, $p(j) = \pi_j$ is the prior probability, $A_j(x) = \Sigma_{j\dot{x}x}\big(\Sigma_{jx}\big)^{-1}$, $b_j = \mu_{j\dot{x}} - A_j\mu_{jx}$, the mean of the jth Gaussian is $\mu_j \triangleq \big[\mu_{jx}, \mu_{j\dot{x}}\big]^T$ and the covariance of the jth Gaussian is $\Sigma_j \triangleq \begin{pmatrix} \Sigma_{j\dot{x}} & \Sigma_{jx\dot{x}} \\ \Sigma_{j\dot{x}x} & \Sigma_{j\dot{x}} \end{pmatrix}$. The GMM parametrization has a summation of linear part multiplied by the nonlinear weight terms $h_j(\cdot)$.

7.2.2.2 Extreme learning machine

The function $f(\cdot)$ in Eq. (7.1) is approximated using an ELM NN, which results in the following expression:

$$\dot{x}(t) = W^T\sigma(qs(t)) + \epsilon(x(t)) \tag{7.3}$$

where, the bounded output weight matrix is denoted by $W \in \mathbb{R}^{n_n \times n}$, the number of hidden layers ELM neurons are represented by n_n, and the ELM function reconstruction error is denoted by $\epsilon(x) \in \mathbb{R}^n$. The ELM activation function selected as a vector Sigmoid, denoted as $\sigma(\cdot) \in \mathbb{R}^{n_n}$, is given by $\sigma\big(qs(t)\big) = \left[\frac{1}{1+\exp\left(-(qs(t))_1\right)}, \cdots, \frac{1}{1+\exp\left(-(qs(t))_{n_n}\right)}\right]^T$, where $(qs(t))_i$ is the ith element of the vector $qs(t)$, $q \triangleq \big[P_E, b_p\big] \in \mathbb{R}^{n_n \times (n+1)}$, $P_E \triangleq \mathrm{diag}\big(s_p\big)U^T \in \mathbb{R}^{n_n \times n}$, $s_p \in \mathbb{R}^{n_n}$ is the internal slope and $b_p \in \mathbb{R}^{n_n}$ is the bias vector. The bounded input layer weight matrix is given by $U \in \mathbb{R}^{n \times n_h}$, and the ELM input vector is given by $s(t) = \big[x(t)^T, 1\big]^T \in \mathbb{R}^{n+1}$.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

7.2.3 Learning GMM parameters with contraction constraints

Contraction analysis constraints for GMM approximation of the system model are first derived in this section, and then a constrained optimization problem used for computing the GMM parameters is shown.

7.2.3.1 Contraction constraints for GMM

The GMM parameter learning is addressed using constraints derived by partial contraction analysis. A small sample of N trajectory data containing position and velocity state $x(t)$ is considered. The state derivative contains the velocity and acceleration data. For deriving the contraction constraints, let us consider the following auxiliary system written in the state variable $x(t)$ and an auxiliary variable $y \in \mathrm{R}^n$:

$$\dot{y}(t) = f(y(t), x(t)) = \sum_{j=1}^{L} h_j(x(t))\big(A_j y(t) + b_j\big) \tag{7.4}$$

which is obtained by replacing the variable $x(t)$ by variable $y(t)$ for certain terms of Eq. (7.2) such that the system model is linear in $y(t)$. The system in Eq. (7.4) is contracting in $y(t)$ if the following inequalities are satisfied.

$$\big(A_j\big)^T M(y) + \dot{M}_j(y) + M(y)A_j \preceq -\gamma M(y),\ j = 1, .., L, \forall y \tag{7.5}$$

$$A_j x^* + b_j = 0,\ j = 1, 2, \ldots, L \tag{7.6}$$

where, γ is a positive constant, $M(y) \in \mathbb{R}^{n\times n}$ is a positive definite (PD) symmetric matrix called as a contraction metric, the ik^{th} element of $\dot{M}_j(y)$ is given by $\dot{M}_{j_{ik}}(y) \triangleq \frac{dM_{ik}(y)}{dy}\big(A_j y + b_j\big)$. The implication of Eq. (7.2) being partially contracting is that the trajectories of Eq. (7.4) will converge to the trajectories of Eq. (7.2) which converge to x^* as represented by the data used to learn the system.

The constraints in Eqs. (7.5)–(7.6) can be verified using the following reasoning. The constraint in Eq. (7.6) verifies that $y = x^*$ is the system equilibrium, which is also a particular solution of the system in Eq. (7.4). Consider the following reasoning for verifying Eq. (7.5). For $y(t) = x(t)$ in Eq. (7.4), Eq. (7.4) is equivalent to Eq. (7.2), which means particular solutions of Eq. (7.2) are solutions of Eq. (7.4). Application of the contraction analysis [50] to the system in Eq. (7.4) leads to following constraint:

$$\delta y^T\left(\frac{\partial f}{\partial y}^T M(y) + \dot{M}(y) + M(y)\frac{\partial f}{\partial y}\right)\delta y \le -\gamma \delta y^T M(y)\delta y \tag{7.7}$$

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

where, $M(\gamma) \in \mathbb{R}^{n\times n}$ is a PD symmetric matrix uniformly in γ, $\dot{M}(\gamma)$ is given by $\dot{M}_{ik}(\gamma) = \frac{dM_{ik}(\gamma)}{d\gamma} f(\gamma) = \frac{dM_{ik}(\gamma)}{d\gamma}\left(\sum_{j=1}^{L}\left\{h_j(x)\left(A_j\gamma + b_j\right)\right\}\right)$, which can be written as $\dot{M}(\gamma) = \sum_{j=1}^{L}\left\{h_j(x)\dot{M}_j(\gamma)\right\}$. The $\frac{\partial f}{\partial \gamma}$ of Eq. (7.4) is written as $\frac{\partial f}{\partial \gamma} \triangleq \sum_{j=1}^{L}\left\{h_j(x)A_j\right\}$. The following inequality is derived after some algebraic calculations and using Eq. (7.5).

$$\delta\gamma^T\left(\frac{\partial f}{\partial \gamma}^T M(\gamma) + \dot{M}(\gamma) + M(\gamma)\frac{\partial f}{\partial \gamma}\right)\delta\gamma \leq -\delta\gamma^T\left(\sum_{j=1}^{L}\left\{h_j(x)[\gamma M(\gamma)]\right\}\right)\delta\gamma \tag{7.8}$$

Since $\sum_{j=1}^{L} h_j(x) = 1$, the inequality in Eq. (7.8) can be rewritten as:

$$\delta\gamma^T\left(\frac{\partial f}{\partial \gamma}^T M(\gamma) + \dot{M}(\gamma) + M(\gamma)\frac{\partial f}{\partial \gamma}\right)\delta\gamma \leq -\gamma\delta\gamma^T M(\gamma)\delta\gamma \tag{7.9}$$

and the auxiliary system in Eq. (7.4) is contracting with respect to γ [50].

Based on the partial contraction theory [54], if the auxiliary system in Eq. (7.4) is contracting, the trajectories $\gamma(t) = x(t), \forall t \geq 0$ and $\gamma = x^*$ are particular solutions of Eq. (7.4), then the trajectories of Eq. (7.2) globally exponentially converge to the goal location x^*. The contraction analysis also provides robustness properties with respect to an external disturbance to the learned system model.

7.2.3.2 Constrained GMM parameter learning

A constrained optimization problem is presented in this section using which the GMM model parameters are estimated from the trajectory data. The problem formulation is given as follows:

$$\left\{\theta_G^*, \theta_M^*\right\} = \arg\min_{\theta_G,\theta_M} \frac{1}{2\mathcal{T}}\sum_{i=1}^{N}\sum_{t=0}^{T_n}\left\|\hat{\dot{x}}_i(t) - \dot{x}_i(t)\right\|^2 \tag{7.10}$$

$$\text{s.t.}\left(A_j\right)^T M(x) + \dot{M}_j(x) + M(x)A_j \preceq -\gamma M(x),\ j = 1, \ldots, L, \forall x \tag{7.11}$$

$$A_j x^* + b_j = 0,\ j = 1, \ldots, L, \tag{7.12}$$

$$M(x, \theta_M) \succ 0, \forall x \tag{7.13}$$

$$\Sigma_j \succ 0, \sum_j \pi_j = 1, 0 \leq \pi_j \leq 1,\ j = 1, \ldots, L \tag{7.14}$$

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

where, $\theta_G = \{\mu_1 \ldots \mu_L, \Sigma_1 \ldots \Sigma_L, \pi_1 \ldots \pi_L\}$ contains GMM parameters, $\theta_M \in \mathbb{R}^m$ contains constant coefficients of $M(x)$, $\mathcal{T} = \sum_{n=1}^{N} T_n$ is the total number of data points. The predicted state derivatives are computed using $\hat{\dot{x}}_n(t) = \hat{f}(x_n(t))$ based on Eq. (7.2). The contraction metric M is also learned from the data. The constraints Eqs. (7.11)–(7.13) ensure the global attraction of the goal location x^* and the constraints in Eq. (7.14) are the hyper parameters of the GMM.

Eqs. (7.11) and (7.13) should be satisfied for all points in the state space so that the convergent behavior can be reproduced from the points in state space that are not present in the data used for learning. Thus, the optimization problem becomes intractable. A two-fold solution is proposed for this optimization problem. First, a polynomial form approximation of the contraction metric in x is used by defining the metric as $M(x) \triangleq \Psi(x)^T \Psi(x)$, where $\Psi(x) \in \mathbb{R}^{n \times n}$ and the elements of $\Psi(x)$ are given by $\Psi ij(x) = r_{ij}(x)$ where $r_{ij}(x)$ is a polynomial function in x up to a maximum order $d_{\max}$. Then, the constraint in Eq. (7.11) is reformulated so that it can be written as a function of the parameters. Using,

$$G_j(\theta_G, \theta_M, x) \triangleq \left(A_j\right)^T M(x) + \dot{M}_j(x) + M(x)A_j + \gamma M(x),\ j = 1, \ldots, L, \tag{7.15}$$

Eq. (7.11) can be reformulated as $y^T G_j(\theta_G, \theta_M, x) y \leq 0, \forall x, y, j = 1, .., L$, where $y \in \mathbb{R}^n$ is a vector of indeterminates [55], which then can be decomposed as:

$$y^T G_k(\theta_G, \theta_M, x) y = m(x, y)^T \bar{G}_k(\theta_G, \theta_M) m(x, y) \tag{7.16}$$

where, $m(x, y) \in \mathbb{R}^{\bar{n}}$ consists of monomial vectors in x and y [55]. Coefficient matching can be used to compute the elements of $\bar{G}_k(\theta_G, \theta_M)$. Similarly, the constraint $M(x, \theta_M) \succ 0, \forall x$ can be rewritten as:

$$y^T M(x, \theta_M) y = l(x, y)^T \bar{M}(\theta_M) l(x, y) \tag{7.17}$$

where, $l(x, y) \in \mathbb{R}^{\bar{m}}$ is a monomial vector in x and y and the elements of $\bar{M}(\theta_M)$ are polynomials in θ_{M_i}.

The system model in Eq. (7.2) is partially contracting and all its solutions will converge to the goal location x^*, if $\bar{M}(\theta_M) \succ 0$, and for $k = 1, 2, \ldots, K$, $\bar{G}_k(\theta_G, \theta_M) \preceq 0$, and $A_k x^* + b_k = 0$.

If $\bar{G}_k(\theta_G, \theta_M) \preceq 0$, $k = 1, 2, \ldots, K$, it directly follows that,

$$m(x, y)^T \bar{G}_k(\theta_G, \theta_M) m(x, y) \leq 0, \forall x, y, k = 1, .., K \tag{7.18}$$

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

where, $m(x, y)$ is a vector of monomials in the elements of x and y. Further, based on Eq. (7.16), the inequality in Eq. (7.18) implies that $y^T G_k(\theta_G, \theta_M, x) y \leq 0, \forall x, y, k = 1, 2, \ldots, K$ and consequently;

$$G(\theta_G, \theta_M, x) \preceq 0, \forall x, k = 1, 2, \ldots, K \tag{7.19}$$

Similarly, based on Eq. (7.17), if $\bar{M}(\theta_M) \succ 0$, then $M(x) \succ 0, \forall x$. Thus, based on Eqs. (7.15) and (7.19), along with the fact that $A_k x^* + b_k = 0$, $k = 1, 2, \ldots, K$, it can be seen that the constraints in Eqs. (7.5) and (7.6) are satisfied. Hence, the system in Eq. (7.2) is partially contracting.

The constrained optimization problem in an implementable form is given by:

$$\left\{\theta_G^*, \theta_M^*\right\} = \arg\min_{\theta_G, \theta_M} \frac{1}{2\mathcal{T}} \sum_{n=1}^{N} \sum_{t=0}^{T_n} \left\| \hat{\dot{x}}_n(t) - \dot{x}_n(t) \right\|^2 \tag{7.20}$$

$$\text{s.t.}\, \bar{G}_k(\theta_G, \theta_M) \preceq 0, k = 1, \ldots, K, \bar{M}(\theta_M) \succ 0, \tag{7.21}$$

$$A_k x^* + b_k = 0, \Sigma_k \succ 0, \sum_k \pi_k = 1, 0 \leq \pi_k \leq 1, \ k = 1, \ldots, K \tag{7.22}$$

7.2.4 Learning ELM model parameters with Lyapunov and barrier constraints

In this section, a method of learning ELM model parameters subject to Lyapunov and barrier constraints is presented.

7.2.4.1 Barrier constraints for ELM

A BF $B : Int(\mathcal{X}_0) \to \mathbb{R}$ is a real-valued nonnegative function used to certify the forward invariance of a closed set enclosing the demonstration data with respect to the system model. The BF is constructed using a continuously differentiable function, $h(x)$, to take smaller values if the states move away from the system constraints and approach infinity if they get closer to the set boundaries. Therefore, the smooth function $h(x)$ is selected such that $h(x) = 0$ indicates the boundary enclosing the demonstration data; positive values show adherence to the system constraints and negative values show violations.

Consider an inverse-type reciprocal BF candidate $B(x) = \frac{1}{h(x)}$, which is defined on the convex connected set $\mathcal{X}_0 \subset \mathbb{R}^2$ given by:

$$\begin{aligned} \mathcal{X}_0 &= \{x \in \mathbb{R}^n : h(x) > 0\} \\ \partial\mathcal{X}_0 &= \{x \in \mathbb{R}^n : h(x) = 0\} \end{aligned} \tag{7.23}$$

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

An example of a continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$ in the form of the general equation of an ellipse is given by:

$$h(x) = \frac{\left((x_1 - x_{1g})\cos\alpha + (x_2 - x_{2g})\sin\alpha\right)^2}{a_1^2} + \frac{\left((x_1 - x_{1g})\sin\alpha + ((x_2 - x_{2g})\cos\alpha\right)^2}{a_2^2} - 1$$

where, $x = [x_1, x_2]^T \in \mathcal{X}_0$, x_{1g} and x_{2g} are the coordinates of the center of gravity of the ellipse, a_1 and a_2 are the major and minor axes of the ellipse, respectively, and α is the orientation of ellipse.

For the reciprocal BF, the constraint can be formulated as:

$$\dot{B}(x) \leq \frac{\gamma}{B(x)}, \quad \forall x \in \mathcal{X}_0 \tag{7.24}$$

which, given the ELM parameterization in Eq. (7.3), $h(x)$ as follows:

$$\nabla B(x)^T f(x) \leq \frac{\gamma}{B(x)} \tag{7.25}$$

$$-\nabla h(x)^T W^T \sigma(qs) \leq \gamma h^3(x), \quad \forall x \in \mathcal{X}_0 \tag{7.26}$$

where the operator $\nabla : C^1(\mathbb{R}^n) \to \mathbb{R}^n$ is defined as the gradient $\frac{\partial}{\partial x}$ of a scalar-valued differentiable function with respect to x and C^1 denotes all the differentiable functions whose derivatives are continuous. Enforcing constraint Eq. (7.26) while learning the DS is a difficult task because the constraint Eq. (7.26) has to be satisfied $\forall x \in \mathcal{X}_0$, which is an uncountably infinite set. To overcome the challenge, Lipschitz continuity of system dynamics is used to evaluate the constraint at only a sample number of points in $\mathcal{X}_0$. Using the fact that both $B^{-1}(x)$ and $\dot{B}(x)$ are Lipschitz continuous in x with the corresponding Lipschitz constants $L_{B^{-1}}$ and $L_{\dot{B}}$, the safety constraint in Eq. (7.26) can be modified to assure the ELM parameters are learned while the safety constraint is enforced using only finite number of samples. Other ways of solving semi-infinite program include [56].

7.2.4.2 Lyapunov constraint for ELM

Consider a Lyapunov function of following form $V(x) = \frac{1}{2}(x - x^*)^T(x - x^*)$. Taking the time derivative of the Lyapunov function for the system Eq. (7.3) yields:

$$\dot{V}(x) = \left(x - x^*\right)^T \dot{x} = \left(x - x^*\right)^T \left(W^T \sigma(qs)\right) \tag{7.27}$$

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

The constraint Eq. (7.27) can be rewritten as a linear constraint in decision variable $\theta = \left[w_1^T, \cdots, w_n^T\right]^T \in \mathbb{R}^{(n \cdot n_h)}$ given by:

$$\dot{V}(x) = \left(x - x^*\right)^T \Sigma \theta, \tag{7.28}$$

$$\Sigma = \begin{bmatrix} \sigma(\cdot)^T & & 0_{1\times(n-1)\cdot n_h} \\ & \ddots & \\ 0_{1\times(n-1)\cdot n_h} & & \sigma(\cdot)^T \end{bmatrix}. \tag{7.29}$$

Using the definition of $\dot{V}(x)$ in Eq. (7.28), the Lyapunov constraint can be written as:

$$\left(x - x^*\right)^T \Sigma \theta \leq -\rho \left\| x - x^* \right\|^2 \tag{7.30}$$

The requirement of the satisfaction of constraints in Eqs. (7.26) and (7.30) at all points in the state space makes the ELM parameter learning optimization problem intractable. To overcome this challenge, the modified constraints are formulated, which can be evaluated at finite sample points in the state space.

Let $\mathcal{X}_\tau \subset \mathcal{X}_0$ be a discretization of the state space $\mathcal{X}_0$ with the closest point in $\mathcal{X}_\tau$ to $x \in \mathcal{X}_0$ be denoted by $[x]_\tau$, where $x - [x]_\tau \leq \frac{\tau}{2}$. The modified BF constraint is given by:

$$-\nabla h([x]_\tau)^T \Sigma \theta - \gamma h^3([x]_\tau) \leq -h^2([x]_\tau)\left(\mathcal{L}_{\dot{B}} + \gamma L_{B^{-1}} \frac{\tau}{2}\right), \quad \forall [x]_\tau \in \mathcal{X}_\tau \tag{7.31}$$

where, $\mathcal{L}_{\dot{B}} \triangleq \left(L_{\partial B}\left(\bar{W}\sqrt{n_h} + \bar{\epsilon}\right) + L_B L_f\right)\frac{\tau}{2}$ and L_B^{-1} is the Lipschitz constant of $B^{-1}(x)$. Similarly, a modified Lyapunov constraint can be derived as follows:

$$\left([x]_\tau - x^*\right)^T \Sigma \theta \leq -\beta([x]_\tau) - \mathcal{L}_V \frac{\tau}{2}, \quad \forall [x]_\tau \in \mathcal{X}_\tau \tag{7.32}$$

where, $\mathcal{L}_V = L_{\dot{V}} + 2\rho L_V$, L_V is a Lipschitz constant of the Lyapunov function and $L_{\dot{V}}$ is a suitable constant. More details of the derivation can be found in the study of Salehi et al. [49].

7.2.4.3 Constrained ELM parameter learning

The constrained ELM parameter learning problem with stability and barrier safety constraints is formulated as a quadratic program (QP). For maintaining the feasibility of the stability constraint a relaxation variable δ is added

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

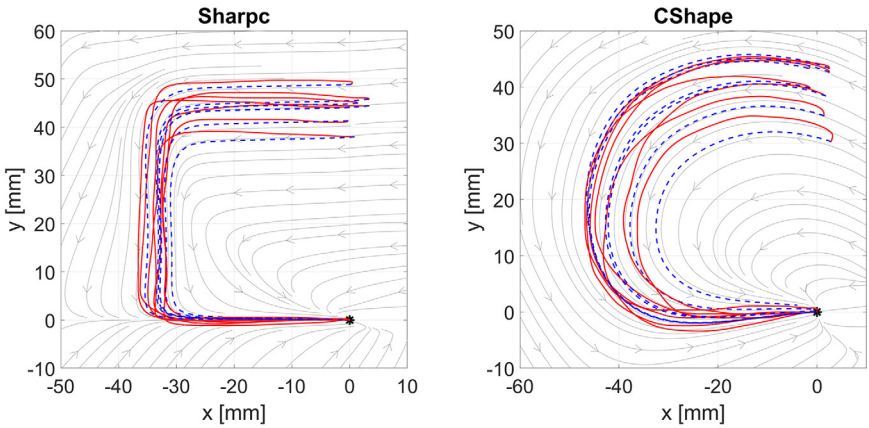


Figure 7.5 Trajectory data used for learning contracting dynamical systems primitives (CDSP) models of two shapes shown in *solid red lines*, trajectories generated from the learned model shown in *dashed blue lines*. The streamlines of the trajectories generated from the learned models are shown in the *gray line*. For interpretation of the references to color in this figure legend, the reader is referred to the web version of this chapter.

to the constraint. The following QP with safety and stability constraint is formulated:

$$\left[W^{*^T}, \delta^*\right]^T = \arg \min_{(W,\delta)\in\mathbb{R}^{n_h+1}} \sum_{t=0}^{T_n} \left[\dot{x}(t) - \hat{\dot{x}}(t)\right]^T \left[\dot{x}(t) - \hat{\dot{x}}(t)\right] + \mu_W \left(tr\left(W^T W\right)\right) + p\delta^2 \tag{7.33}$$

$$-\nabla h([x]_\tau)^T \Sigma\theta - \gamma h^3([x]_\tau) \leq -h^2([x]_\tau)\left(\mathcal{L}_{\dot{B}} + \gamma L_{B^{-1}} \frac{\tau}{2}\right), \quad \forall [x]_\tau \in \mathcal{X}_\tau, \tag{7.34}$$

$$\left([x]_\tau - x^*\right)^T \Sigma\theta \leq -\beta([x]_\tau) - \mathcal{L}_V \frac{\tau}{2} + \delta, \quad \forall [x]_\tau \in \mathcal{X}_\tau, \tag{7.35}$$

where, p and δ are positive constants.

7.2.5 Simulations

Simulations are conducted using a standard computer configuration for both CDSP and SMP algorithms. The CDSP algorithm is coded using MATLAB and a constrained GMM parameter learning problem is solved. The results of the method are shown on two shapes from the LASA dataset [3]. The results are shown in Fig. 7.5. From the results, it can be seen that the method learns shape retention and goal convergence from different points in the state space.

The SMP algorithm is coded using MATLAB and a QP problem is solved with linear constraints. LASA human handwriting dataset is used for

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

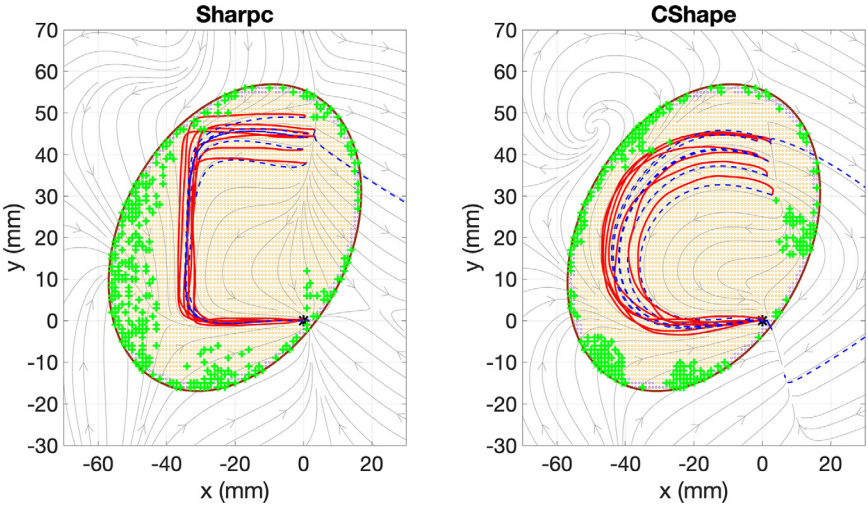


Figure 7.6 Dynamic system models learned without barrier safety constraints from trajectory data shown using *solid red lines* and trajectories generated from the learned models shown using *dashed blue lines*. For interpretation of the references to color in this figure legend, the reader is referred to the web version of this chapter.

testing the performance of the SMP method. The parameters of the inner layers of the ELM network, i.e., the slopes and biases are computed using the BIP algorithm by Neumann and Steil [25], and two sets of results are shown. The first result is generated by learning a model without constraints. The trajectories generated using the learned model are shown in Fig. 7.6. Model with Lyapunov and BF constraints is learned. The trajectories with constraints are shown in Fig. 7.7. From Fig. 7.6, it can be seen that the trajectories generated by the model without constraints can go out of the safe region whereas from Fig. 7.7, it can be seen that the trajectories of the constrained model stay within the safe region. Hence, the safety objective is achieved. It is also noticed that with the preference given to safety constraints, the goal convergence property may take lesser priority and some trajectories inside the safe region may converge to spurious attractors.

7.3 Conclusion

In this section, concluding remarks on the constrained model learning methods presented in this chapter are provided. Some future directions of research for learning models and processes from data, safety considerations, and decentralized learning in the context of manufacturing are also provided.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

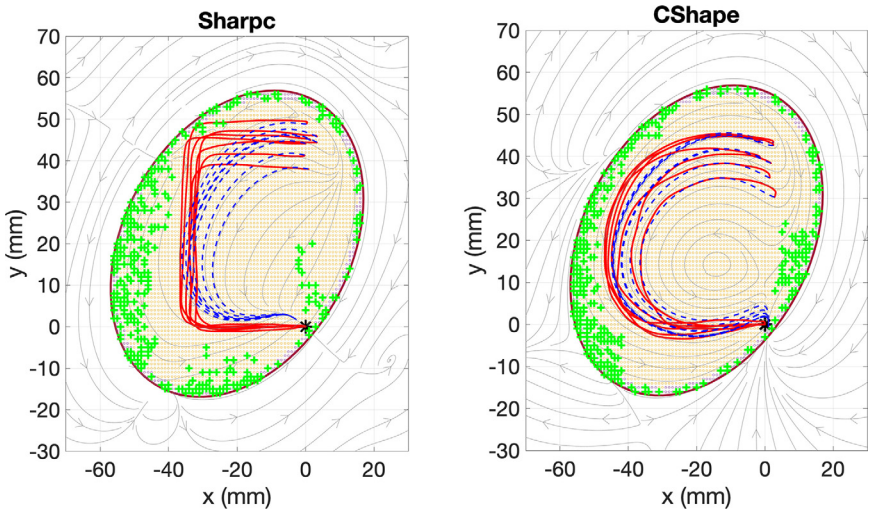


Figure 7.7 Models of two shapes, from the LASA handwriting dataset, learned using the safe movement primitives (SMP) algorithm. The streamlines of the models are shown in *gray line* along with the demonstrations in *solid red lines* and the reproductions in *broken blue lines* generated using the learned models. For interpretation of the references to color in this figure legend, the reader is referred to the web version of this chapter.

7.3.1 Discussion on the constrained model learning methods

This chapter presents two constrained learning methods for nonlinear DS model learning. The first method, called CDSP, uses a contraction analysis tool from systems theory for generating constraints on the trajectories of the learned model such that the trajectories follow a particular shape in the position state and converge to the goal state. The second method, called SMP, uses barrier function and Lyapunov function tools from systems theory for generating constraints on the trajectories of the learned model such that the trajectories remain inside a prescribed region of the state space called as a safe region, and converge to the goal state. Both CDSP and SMP methods were trained using a small dataset from a publicly available LASA dataset and were tested using data from the same dataset. For the SMP algorithm, it is observed that when the models are learned without safety constraints the trajectories of the learned models are likely to go outside the prescribed safe region whereas with the constrained learning all the trajectories of the learned model stay within the prescribed region in state space. The trajectories of the CDSP model tend to retain the shape of the position states because the contraction analysis provides incremental stability, i.e., the

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

convergence with respect to the neighboring trajectories. The trajectories generated from the learned SMP model may converge to a spurious state inside the safe region because the stability constraints are generated using a quadratic Lyapunov function, similar to the SEDS algorithm. This drawback can be overcome by using other forms of Lyapunov function constrained such as the ones in CLF-DM or using contraction analysis constrained in conjunction with the barrier function constraints.

7.3.2 Who to imitate?

This is a critical question in imitation learning which requires more research. Usually, the demonstration is provided by one or more teachers who are modeling a specific task. When different teachers have different ways of teaching or when the demonstrations are conflicting across teachers who to imitate is a question that needs further studies. With humans, this is a learned behavior that depends, among other things, on the relative importance or the status of teachers, as well as the outcomes. In machines, these hierarchies, processes, and evaluation of outcomes, have to be coded in. This is a fertile ground for research and might even lead to the standardization of learning processes. What are the selection processes, the success criteria, the metrics that govern learning? When the teacher is another robot or a computer program, the issues become even more interesting, and is an area of research that needs further investigation.

7.3.3 Using imitation learning when a robot collaborates with a human

Many manufacturing tasks that involve collaboration between a human and a robot require sequential task execution between these two actors. In such tasks, the robot needs to use the skills learned from demonstration and develop optimal collaborative actions with the human by using its perception capabilities. Using perception capabilities, the robot can figure out the motion intent of the human and adapt its own motions or switch its primitives based on if the human switches its own motion. This means that the robot needs to have some level of autonomy built into it with safeguards and fail-safe positions that ensure an acceptable level of safety in the working environment. Without some form of demonstrable safety levels, regulatory statutes will not allow such collaborative work to continue. This will engender a number of interesting areas of research in human-machine interfaces, safety analysis, perception, visualization, AI, etc. This can also lead to a study that results in standardization and safety regulations.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

7.3.4 Imitation across a large number of robots

When there are many robots operating in different environments, e.g., different houses or rooms, a key element is the decentralization of the learning process; this is an area of research that needs further investigation. Robots can learn a base model in a centralized manner. However, to fine-tune their skills, they can learn from data that is locally available in each of their specific working environments. The updated models can be sent to a centralized server to aggregate the knowledge learned from different robots to improve their actions. In this method of learning, a permanent communications-link to the centralized data server is not required. The robots can refine their skills by using reinforcement learning or any other learning method with local data and locally optimal actions. A combination of meta-learning with imitation can also be used for quickly learning the skills by utilizing prior experience from meta-learned models. However, processes for reconciling actions and consequences across different robots on a periodic basis can result in improved "fleet-wide" learning and thereby more globally optimal behaviors. This concept can also be used in manufacturing factories or factories at different locations where the concept of skill can be generalized to methods or ways of doing things. For example, optimal motions for similar actions, such as assembly, and welding might be learned at one location and then broadcast across the entire family of robots performing similar operations elsewhere, with appropriate modifications to accommodate for local differences.

Acknowledgment

Authors would like to acknowledge technical discussions and inputs from Dr. Ravi Rajamani. This work was supported by NSF grant no. SMA-2134367.

References

[1] S. Schaal, Is imitation learning the route to humanoid robots? Trends Cogn. Sci. 3 (6) (1999) 233–242.
[2] A.J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, S. Schaal, Dynamical movement primitives: learning attractor models for motor behaviors, Neural Comput. 25 (2) (2013) 328–373.
[3] S.M. Khansari-Zadeh, A. Billard, Learning stable nonlinear dynamical systems with Gaussian mixture models, IEEE Trans. Rob. 27 (5) (2011) 943–957.
[4] J. Rey, K. Kronander, F. Farshidian, J. Buchli, A. Billard, Learning motions from demonstrations and rewards with time-invariant dynamical systems based policies, Auton. Robots 42 (1) (2018) 45–64.
[5] C. Wang, Yu Zhao, C.Y. Lin, M. Tomizuka, Fast planning of well conditioned trajectories for model learning, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2014, pp. 1460–1465.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[6] B.D. Argall, S. Chernova, M. Veloso, B. Browning, A survey of robot learning from demonstration, Rob. Autom. Syst. 57 (5) (2009) 469–483.
[7] S. Calinon, Learning from demonstration (programming by demonstration), Encyclopedia of Robotics, Springer, Berlin, Heidelberg, Germany, 2018, pp. 1–8.
[8] C. Breazeal, B. Scassellati, Robots that imitate humans, Trends Cogn. Sci. 6 (11) (2002) 481–487.
[9] G.M. Hayes, J. Demiris, A Robot Controller Using Learning by Imitation, University of Edinburgh, Department of Artificial Intelligence, 1994.
[10] C.L. Nehaniv, K. Dautenhahn, et al., The correspondence problem, Imitation in Animals and Artifacts, 41, Citeseer, 2002.
[11] F. Stramandinoli, K.G. Lore, J.R. Peters, P.C. ONeill, B.M. Nair, R. Varma, J.C. Ryde, J.T. Miller, K.K. Reddy, Robot learning from human demonstration in virtual reality, Hum.-Rob. Interact. 437 (2018).
[12] A. Coates, P. Abbeel, A.Y. Ng, Apprenticeship learning for helicopter control, Commun. ACM 52 (7) (2009) 97–105.
[13] D.H. Grollman, O.C. Jenkins, Incremental learning of subtasks from unsegmented demonstration, in: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010, pp. 261–266.
[14] P. Agarwal, J. Fox, Y. Yun, M.K. OMalley, A.D. Deshpande, An index finger exoskeleton with series elastic actuation for rehabilitation: design, control and performance characterization, Int. J. Rob. Res. 34 (14) (2015) 1747–1772.
[15] B. Kim, A.D. Deshpande, An upper-body rehabilitation exoskeleton harmony with an anatomical shoulder mechanism: design, modeling, control, and performance evaluation, Int. J. Rob. Res. 36 (4) (2017) 414–435.
[16] G.F Rossano, C. Martinez, M. Hedelind, S. Murphy, T.A. Fuhlbrigge, Easy robot programming concepts: An industrial perspective, in: IEEE International Conference on Automation Science and Engineering (CASE), 2013, pp. 1119–1126.
[17] H. Ravichandar, P. kumar Thota, A.P. Dani, Learning periodic motions from human demonstrations using transverse contraction analysis, in: American Control Conference (ACC), 2016, IEEE, 2016, pp. 853–4858.
[18] N. Koenig, MJ. Mataríc, Robot life-long task learning from human demonstrations: a bayesian approach, Auton. Rob. 40 (6) (2016) 1–16.
[19] A. Rai, F. Meier, A. Ijspeert, S. Schaal, Learning coupling terms for obstacle avoidance, in: International Conference on Humanoid Robotics, 2014, pp. 512–518.
[20] A. Ude, A. Gams, T. Asfour, J. Morimoto, Task-specific generalization of discrete and periodic dynamic movement primitives, IEEE Trans. Rob. 26 (5) (2010) 800–815.
[21] B. Nemec, M. Tamo, F. Worgotter, A. Ude, Task adaptation through exploration and action sequencing, in: IEEE-RAS International Conference on Humanoid Robots, 2009, pp. 610–616.
[22] A. Paraschos, C. Daniel, J.R. Peters, G. Neumann, Probabilistic movement primitives, Adv. Neural Inf. Process. Syst. (2013) 2616–2624.
[23] S.M. Khansari-Zadeh, A. Billard, Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions, Rob. Autom. Syst. 62 (6) (2014) 752–765.
[24] K. Neumann, J.J. Steil, Learning robot motions with stable dynamical systems under diffeomorphic transformations, Rob. Autom. Syst. 70 (2015) 1–15.
[25] K. Neumann, J.J. Steil, Optimizing extreme learning machines via ridge regression and batch intrinsic plasticity, Neurocomputing 102 (2013) 23–30.
[26] A. Lemme, K. Neumann, R.F. Reinhart, J.J. Steil, Neurally imprinted stable vector fields, in: European Symposium on Artificial Neural Networks, Citeseer, 2013, pp. 327–332.
[27] S. Calinon, D. Bruno, D.G. Caldwell, A task-parameterized probabilistic model with minimal intervention control, in: IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 3339–3344.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[28] I.R. Manchester, M. Revay, R. Wang, Contraction-based methods for stable identification and robust machine learning: a tutorial, arXiv preprint arXiv:2110.00207 (2021).
[29] H. Ravichandar, A.P. Dani, Learning contracting nonlinear dynamics from human demonstration for robot motion planning, in: ASME Dynamic Systems and Control Conference (DSCC), 2015.
[30] C. Bowen, R. Alterovitz, Closed-loop global motion planning for reactive execution of learned tasks, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 1754–1760.
[31] B. Akgun, M. Cakmak, K. Jiang, A.L. Thomaz, Keyframe-based learning from demonstration, Int. J. Soc. Robot 4 (4) (2012) 343–355.
[32] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[33] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.
[34] F.L. Lewis, D. Vrabie, K.G. Vamvoudakis, Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers, IEEE Control. Syst. Mag. 32 (6) (2012) 76–105.
[35] N. Jetchev, M. Toussaint, Fast motion planning from experience: trajectory prediction for speeding up movement generation, Auton. Rob. 34 (1) (2013) 111–127.
[36] F. Stulp, E.A. Theodorou, S. Schaal, Reinforcement learning with sequences of motion primitives for robust manipulation, IEEE Trans. Rob. 28 (6) (2012) 1360–1370.
[37] E. Todorov, M.I. Jordan, Optimal feedback control as a theory of motor coordination, Nat. Neurosci. 5 (11) (2002) 1226–1235.
[38] P. Abbeel, A.Y. Ng, Apprenticeship learning via inverse reinforcement learning, in: International Conference on Machine Learning, ACM, 2004, pp. 1–8.
[39] J.P. Laumond, N. Mansard, J.B. Lasserre, Optimality in robot motion: optimal versus optimized motion, Commun. ACM 57 (9) (2014) 82–89.
[40] MC. Priess, J. Choi, C. Radcliffe, The inverse problem of continuous-time linear quadratic gaussian control with application to biological systems analysis, in: ASME 2014 Dynamic Systems and Control Conference, 2014.
[41] J.P. Laumond, N. Mansard, J. Bernard Lasserre, Optimization as motion selection principle in robot action, Commun. ACM 58 (5) (2015) 64–74.
[42] K. Dupree, P.M. Patre, M. Johnson, W.E. Dixon, Inverse optimal adaptive control of a nonlinear Euler-Lagrange system, part I: full state feedback, in: IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference, 2009, pp. 321–326.
[43] K. Dupree, M. Johnson, P.M. Patre, W.E. Dixon, Inverse optimal control of a nonlinear Euler-Lagrange system, part II: output feedback, in: IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference, 2009, pp. 327–332.
[44] S. Schaal, P. Mohajerian, A. Ijspeert, Dynamics systems vs. optimal control-a unifying view, Prog. Brain Res. 165 (2007) 425–445.
[45] C. Finn, T. Yu, T. Zhang, P. Abbeel, S. Levine, One-shot visual imitation learning via meta-learning, in: Conference on Robot Learning, 2017, pp. 357–368.
[46] H. Ravichandar, I. Salehi, A. Dani, Learning partially contracting dynamical systems from demonstrations, in: Proceedings of the 1st Annual Conference on Robot Learning, 78, PMLR, 2017, pp. 369–378.
[47] H.C. Ravichandar, A. Dani, Learning position and orientation dynamics from demonstrations via contraction analysis, Auton. Rob. 43 (4) (2019) 897–912.
[48] I. Salehi, G. Yao, A.P. Dani, Active sampling based safe identification of dynamical systems using extreme learning machines and barrier certificates, in: International Conference on Robotics and Automation, 2019, pp. 22–28.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[49] I. Salehi, G. Rotithor, G. Yao, A.P. Dani, Dynamical system learning using extreme learning machines with safety and stability guarantees, Int. J. Adapt. Control Signal Process. 35 (6) (2021) 894–914.
[50] W. Lohmiller, J.-J.E. Slotine, On contraction analysis for nonlinear systems, Automatica 34 (6) (1998) 683–696.
[51] D. Angeli, A lyapunov approach to incremental stability properties, IEEE Trans. Autom. Control 47 (3) (2002) 410–421.
[52] A.P. Dani, S-Jo Chung, S. Hutchinson, Observer design for stochastic nonlinear systems via contraction-based incremental stability, IEEE Trans. Autom. Control 60 (3) (2015) 700–714.
[53] D.A. Cohn, Z. Ghahramani, M.I. Jordan, Active learning with statistical models, J. Artif. Intell. Res. 4 (1996) 129–145.
[54] W. Wang, J.J.E. Slotine, On partial contraction analysis for coupled nonlinear oscillators, Biol. Cybern. 92 (1) (2005) 38–53.
[55] D. Henrion, A. Garulli, Positive Polynomials in Control, Vol. 312, Springer Science & Business Media, 2005.
[56] B. Bhattacharjee, P. Lemonidis, W.H. Green Jr, P.I. Barton, Global solution of semi-infinite programs, Math. Program. 103 (2) (2005) 283–307.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

CHAPTER EIGHT

Artificial intelligence for materials damage diagnostics and prognostics

Sarah Malik and Antonios Kontsos
Theoretical & Applied Mechanics Group, Department of Mechanical Engineering and Mechanics, Drexel University, Philadelphia, PA, United States

8.1 Introduction

The purpose of this chapter is to provide a review of the artificial intelligence (AI) methods used in relation to investigations of materials level damage. This topic is generally very broad as it refers, for example, to the use of AI methods for materials discovery (e.g., in the federally funded materials genome initiative [1]), for the development of novel materials (e.g., the national nanotechnology initiative [2]) for which AI methods have been used to obtain unprecedented properties [3], or in design strategies to make new materials [4,5].

The focus of this chapter is to provide a perspective on how the manufacturing industries can benefit from advances in using AI in different stages of materials qualification and usage which span the preprocessing, manufacturing operation, as-manufactured states as well as postprocessing phases to succeed in producing manufactured parts with desired or at least acceptable final properties, attributes, and behavior. In this context, the use of AI for materials damage in this chapter will refer to the various imaging, characterization, monitoring, assessment, and testing methods that are either already in use or are emerging for the cases, including advanced, additive, and smart manufacturing, which are gaining significant interest with respect to other more established manufacturing methods. More specifically, such AI methods are now implemented in: (1) data-driven quality control in the digital thread and twin implementations in which the material state evaluation including damage, is assessed and updated based on sensing (e.g., imaging) and processing parameters (e.g., laser power and speed in metal additive manufacturing) [6]; (2) assessment of the as-manufactured material state in relation to the process parameters and manufacturing methods

Artificial Intelligence in Manufacturing: Concepts and Methods.
DOI: https://doi.org/10.1016/B978-0-323-99134-6.00004-9
Copyright © 2024 Elsevier Inc.
All rights are reserved, including those for text and data mining, AI training, and similar technologies.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY
Copyright Elsevier 2024

used which involves a plethora of metrology and nondestructive evaluation (NDE) methods (e.g., X-ray computer tomography, microscopy, etc.) [7]; and (3) connections of the as-built state with performance and usage data which assist in assessing the role of manufacturing in creating the multiscale morphological characteristics that affect the functional performance of materials [8–10].

This chapter is organized as follows: Section 8.1 provides a brief definition of materials damage in relation to the various length scales that such damage may be observed and in relation to the manufacturing methods used to produce them. It further provides details on the definition of the terms damage diagnostics and prognostics. Finally, it briefly explains the need for the use of AI methods in this field and provides a taxonomy of the most well-known AI methods used for materials damage diagnostics and prognostics. This taxonomy serves also as the backbone to present details in Section 8.2 on the technical background, applications, and relative advantages and disadvantages of each AI method in this field. Furthermore, Section 8.3 discusses the current challenges and opportunities for using AI methods for damage diagnostics and prognostics. The chapter concludes with Section 8.4 that provides concluding remarks on this topic.

8.1.1 Background on materials damage

Damage at the materials' level refers to the set of states which adversely impact their properties, behavior, performance, and function with respect to their physical or designed properties [11–13]. This practically means that a material damage state, especially in the as-manufactured condition, relates to defects (e.g., cracks, pores, phase transformations, etc.) which eventually result in several degradation processes such as fracture, corrosion and changes in plasticity, conductivity, and emissivity, etc. These processes can be considered both irreversible and degenerative. A couple of points should be mentioned here for clarification. First, damage is multiscale which practically means that depending on the length scale of analysis, the characteristic features or defects which may be the precursor to subsequent (catastrophic) damage development, vary significantly among materials. For example, Fig. 8.1 describes a categorization of such defects in the case of metallic materials from atomistic to macroscale dimensions [14]. Second, defects themselves are not necessarily damage. For example, a material might have a statistically significant number of defects such as pores in it, however, it may not necessarily experience compromised properties or behavior at a

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

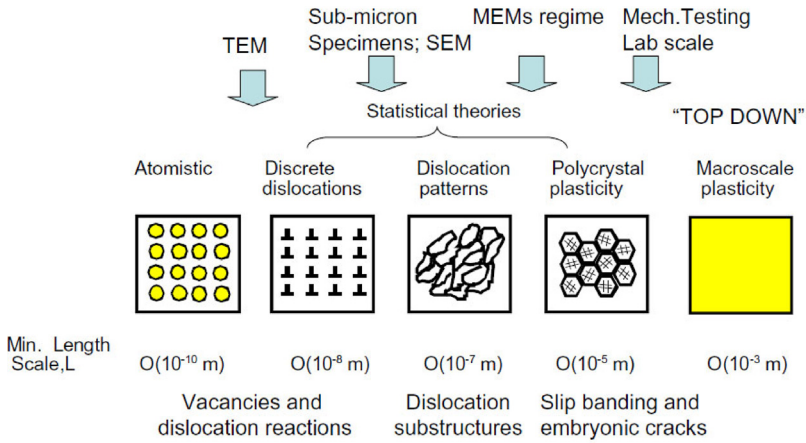


Figure 8.1 Representative categorization of material damage effects in metals in terms of length scale ranging from atomistic to macroscale. Corresponding methods of imaging and characterization for such damage effects at respective length scales are also provided [14]. *(From: This figure was published in [14], copyright Elsevier, 2010).*

level at which these could impact the as-designed specification and until some critical density of such defects or their interactions becomes significant based on external loading and operational conditions [15]. Third, there are multiple ways to evaluate or assess damage in materials which include a variety of sensing, imaging, characterization, and testing methods [16–18]. In this context, AI methods for assessing materials damage relate to all stages of manufacturing including preprocessing, manufacturing operation, as manufactured states, as well as postprocessing phases. For example, in the case of metal additive manufacturing, defects in the feedstock material (e.g., powder size distribution spread, sphericity, etc.), affect the manufacturing process parameters (e.g., temperature control, byproduct generation, etc.), which produce defects in the as-manufactured states (including porosity, delaminations, cracks, residual stresses, etc.) [19]. The complex interactions among those defects result in damage appearance at the length scales similar to Fig. 8.1, which creates the need for efficient procedures to detect such damage as early as possible, e.g., at the design and manufacturing stages.

8.1.2 Need for AI in materials damage diagnostics and prognostics

Given the description of damage above, we can define damage diagnostics as any approach to detect, identify and classify damage states in materials. This

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

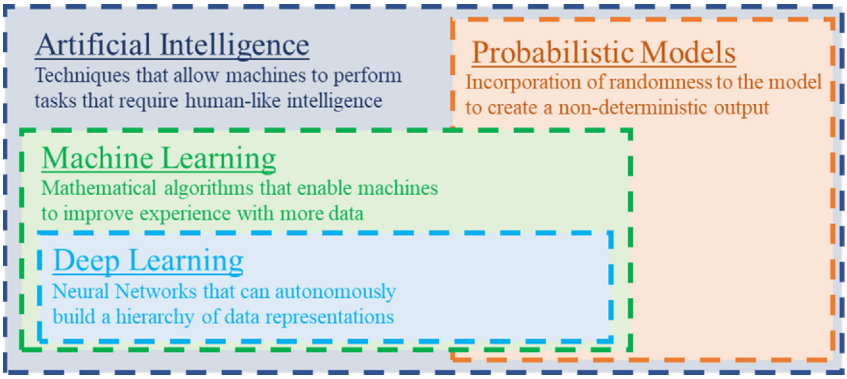


Figure 8.2 Brief representation of the AI methods presented in this chapter and their relationships.

type of classification is typically based on measurement and evaluation data which are processed for visualization and analysis purposes to inform damage prognostics efforts. By materials prognostics, we refer here to predictions based on modeling and simulations of damage initiation, development, and of ultimate failure. Therefore, prognostics in this context refers to a variety of efforts to infer how existing information, e.g., by applying diagnostics methods, combined with archived information and other types of prior knowledge, could be used to issue any predictions of future material states. Overall, by viewing materials damage diagnostics and prognostics together, the damage identification process has been further suggested to be classified into four levels, namely: Level 1 (damage existence), Level 2 (Level 1 and damage location), Level 3 (Level 2 and damage severity), Level 4 (Level 3 and damage state prediction) [20]. Such classification is case-dependent; however, it demonstrates the reasons for which the use of advanced methods of analysis and processing are needed to understand and improve ways by which materials are used, which is crucial in all aspects of manufacturing.

It should be further mentioned here that, before measurement and evaluation data are analyzed and used by, e.g., AI methods, it is important to preprocess the data using methods of normalization, filtering, etc. In fact, it is often said that this is the lengthiest piece of the data analysis process. Algorithms such as principal component analysis (PCA), Pearson's correlation coefficient, forward-backward elimination, and many others are all methods to reduce the dimensionality of data [21–23]. Once these methods are applied, various processing algorithms can be applied based on the type of variables for training and testing. In this context, Fig. 8.2

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

outlines the types of AI algorithms discussed in this chapter. Specifically, AI is defined here as the class of all methods and techniques that allow machines to perform tasks that require human-type intelligence. These methods include machine learning (ML), a subset of AI, which is data-driven mathematical modeling. Deep learning (DL) is an ML method, which involves the use of neural networks (NNs) to automatically build hierarchical data representations. Finally, it should be noted that probabilistic models (PMs) will also be discussed in relation to the AI methods in this chapter, with the understanding that PMs may overlap with all of the AI, ML, and DL methods.

Furthermore, Tables 8.1 and 8.2 provide a more detailed list of the particular AI methods that will be discussed in this chapter along with their outputs once applied in a particular case of materials damage investigation.

Regarding potential overlaps in the presentation of the technical background of these methods with similar details presented in other chapters of this book, in this chapter our focus is on presenting the fundamental technical background and concepts for the AI methods that have an application in materials damage diagnostics and prognostics.

8.2 AI methods for materials diagnostics and prognostics

In this section, specific AI methods are presented in accordance with Tables 8.1 and 8.2. A brief overview of each method's technical background is provided and is followed by representative application cases in the context of materials damage diagnostics and prognostics. The discussion in each section is then followed by the relative advantages and disadvantages of using the particular AI method presented, in relation to manufacturing.

8.2.1 Support vector machines

Support vector machines (SVMs) represent a type of supervised ML algorithm which can be used for classification and regression analysis. The goal of SVMs is to characterize data points in n-dimensional space using the concept of a decision boundary, which is practically a function used to mathematically distinguish classes in the dataset, depending on the selected feature space. In

This book belongs to COMEX COMEX (comexcom83@gmail.com Copyright Elsevier 2024

Table 8.1 Artificial intelligence methods for materials diagnostics in manufacturing.

Primary AI model	Key model output for classification	Representative references	Representative output
Support vector machine (SVM)	Decision boundary	[24–27]	Flaw in materials (discontinuities, porosity, cracks, or inclusions); machine condition monitoring; machine tool breakage; bearing fault diagnosis
Tree-based classifiers	Tree architecture	[28–31]	Machine parameters; machine state; composite damage states; materials properties
Bayesian classifiers	Posterior distribution	[32–37]	Composite damage states; 3D Printing faults; bearing faults diagnosis; gearbox fault diagnosis; material degradation
Unsupervised machine learning	Cluster parameters	[38–42]	Additive manufacturing process failures; composite damage states; material damage mechanisms; machine tool breakage
Deep learning methods (artificial neural network (ANN), multi-layer perceptron (MLP), convolutional neural network (CNN))	Architecture and weights	[43–48]	Material defects (e.g., crack, gas porosity, and lack of fusion); material build quality; damage detection; machine condition monitoring

This bookbelongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY

Copyright Elsevier 2024

Table 8.2 Artificial intelligence methods for materials prognostics in manufacturing.

Primary AI model	Key model output for classification	Representative references	Representative output
Support vector machine (SVM)	Decision boundary	[49–51]	Material fatigue life; remaining useful life (RUL) of cutting tools; RUL of bearings
Tree-based classifiers	Tree architecture	[52, 53]	Material fatigue life; RUL of manufacturing systems
Deep learning methods (ANN, autoencoders, CNN)	Architecture and weights	[48,49,52–54]	Material fatigue life; machine health monitoring; machine tool wear
Adaptive neuro-fuzzy inference system (ANFIS)	Architecture and weights	[50,55–57]	Mateiral fatigue life; machine tool wear
Hidden Markov models	Hidden states	[58–61]	Machine condition monitoring; material time-to-failure
Filtering-based approaches	Filter weights	[62–65]	Crack Length; RUL of bearings; machine condition monitoring

two dimensions this decision boundary is a line, while in three dimensions it is a plane and in n-dimensions it is a hyperplane.

8.2.1.1 Technical background

For a n-dimensional dataset, a hyperplane is described by Eq. (8.1), where $f(x)$ is the hyperplane, w are the n-dimensional vector of weights, X denotes the n-dimensional vector of the input data, and b is a bias scalar number. The optimal hyperplane is determined by computing the weight values that maximize the margin or distance between the left and right hyperplane regions, defined in Eqs. (8.2)–(8.3). The hyperplane with the smallest 2-norm of $\boldsymbol{w}$ will have the largest margin and, thus, the goal is to solve the optimization problem in Eqs. (8.4)–(8.5) to find the optimal hyperplane.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

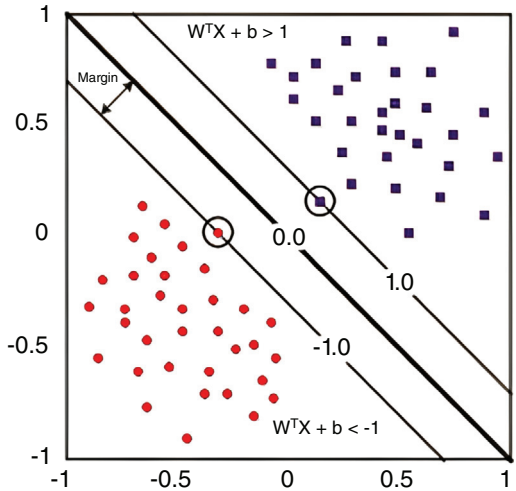


Figure 8.3 Example of a support vector machines (SVM) classifier with a linear decision boundary for a two-dimensional dataset [66]. *(From: This figure was published in [66], copyright EDPSciences, 2015).*

The slack variable ξ_i and the error penalty C are additionally introduced to solve this optimization problem.

$$f(x) = b + \boldsymbol{w}^T \boldsymbol{X} = \sum_{i=1}^{n} w_i x_i + b = 0 \tag{8.1}$$

$$f(x) = b + \boldsymbol{w}^T \boldsymbol{X} = \sum_{i=1}^{n} w_i x_i + b = 1 \tag{8.2}$$

$$f(x) = b + \boldsymbol{w}^T \boldsymbol{X} = \sum_{i=1}^{n} w_i x_i + b = -1 \tag{8.3}$$

$$\min \left\{ \frac{1}{2} \, ||\boldsymbol{w}||^2 + C \sum_{i=1}^{n} \xi_i \right\} \tag{8.4}$$

$$\text{subject to: } \begin{cases} f_i\left(\boldsymbol{w}^T \boldsymbol{x}_i + b\right) \geq 1 - \xi_i, & i = 1 \ldots, n \\ \xi_i \geq 0, & i = 1 \ldots, n \end{cases} \tag{8.5}$$

Fig. 8.3 portrays a visual representation of the decision boundary for an exemplary linear classification in two dimensions. Support vectors are the data points closest to the decision boundary and are circled in Fig. 8.3 for clarity. Once the SVM model is trained, a new data point is evaluated with respect to the hyperplane equation to determine its label.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

If the points are not linearly separable, then the kernel trick can be applied to create a nonlinear decision boundary. Furthermore, the data may not be linearly separable in the original input dataset, however, if a transformation is applied, the resulting classes of data could become linearly separated in a higher dimensional space. Applying such transformation functions to large datasets can have, however, considerable computational costs, and therefore the kernel trick provides an alternative opportunity by using kernel functions. This kernel functions use the original input and return the dot product of the transformed vector in a higher dimensional space. Such kernel methods provide temporary representations of the data using transformed coordinates in a higher dimensional feature space. Moreover, instead of using features, SVM models can input kernel functions together with input data and labels to generate a classifier. Eq. (8.6) provides the mathematical definition of such a kernel function, for which x and y are the inputs, while f is the map from n-dimensional to m-dimensional space. The angle brackets denote the inner product for which m is much larger than n. Common types of kernel functions include Gaussian, sigmoid, polynomial, and hyperbolic tangent functions. Further details about the computation of SVMs are discussed by Evgeniou and Pontil [67] and Cortes and Vapnik [68].

$$K(x,\ y) = \langle f(x),\ f(y) \rangle \tag{8.6}$$

8.2.1.2 Diagnostics applications

In-situ defect detection at the material level during the manufacturing process is an emerging application field for AI methods. This type of defect detection depends on the sensing and instrumentation available, e.g., on modern additive manufacturing systems, but also on additional sensors (including imaging, acoustics, interferometry, pyrometry, etc.), which can be added to the manufacturing system to collect sensing datasets related to "just-formed" material defects. The implementation of AI methods in this type of defect detection further depends on the successful use of data-processing, leveraging hardware and software at the edge (i.e., near the machine), and on the integration with the cloud for data storage, model training, etc. Successful data visualization of processed data is also very important so that either decisions on the manufacturing strategy or even automated procedures, e.g., in closed-loop operations could be achieved. In this context, a representative implementation of SVMs for defect detection was presented by Gobert et al. [24] for laser powder bed fusion (PBF) additive

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

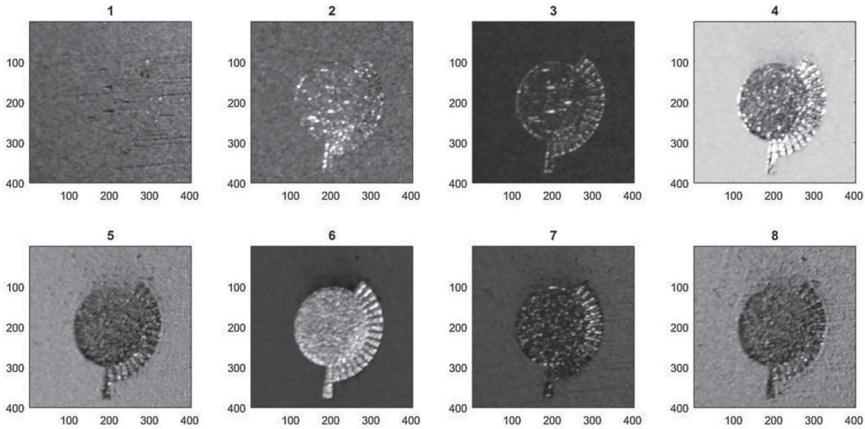


Figure 8.4 In situ image collection during laser powder bed fusion additive manufacturing [24]. *(From: This figure was published in [24], copyright Elsevier, 2018).*

manufacturing (AM). Layer-wise imaging collected by a digital single lens reflex (DSLR) camera was leveraged for the collection of data, as shown in Fig. 8.4. The camera captured multiple images for each build layer. Images were then collected using combinations of various light sources to generate a total of eight different lighting conditions.

Linear SVMs were chosen in this case as binary classifiers to detect flaws using features extracted from images similar to the ones in Fig. 8.4. To achieve this goal, grayscale representations of the images were stacked to create multiple 3D representations of the build process, one for each of the eight types of images captured from each layer. Instead of training a single SVM, the authors used feature matrices computed at each layer to feed into the model. The output decision along with accompanying confidence levels were then fed into a second model for the final classification. Fig. 8.5 shows the ensemble model classification scheme used to train this model. The equations mentioned in the aforementioned section for the hyperplane (Eqs. 8.1–8.3) were optimized via Eqs. (8.4)–(8.5) for each classifier and master model. The relatively high performance of the classification ensemble demonstrated the potential to discriminate between anomalous and nominal voxels using data comprising layer-wise imaging. Table 8.3 provides results for the accuracy, precision, and recall of the SVM and ensemble classifiers. The authors further concluded that this method is well suited for *in situ* layer-wise discontinuity mitigation in an industrial setting, providing accurate information related to part quality from image voxel information alone.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

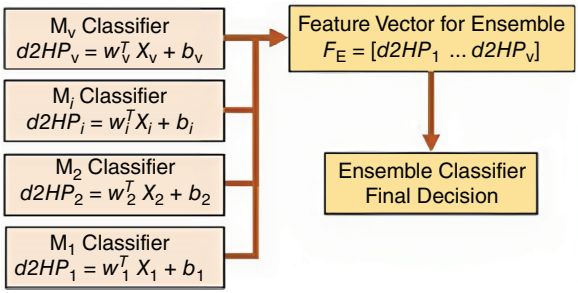


Figure 8.5 Ensemble classification scheme, merging V individual classifiers into a second support vector machines (SVM) model for final decision based on an ensemble type classifier [24]. *(From: This figure was published in [24], copyright Elsevier, 2018).*

Table 8.3 Average values and standard deviations of accuracy, precision and recall for each x support vector machines (SVM) classifier and ensemble classifier [24].

Feature matrix of flash model	Accuracy ± 1 Std. Dev.	Precision ± 1 Std. Dev.	Recall ± Std. Dev.
1	0.72 ± 0.012	0.37 ± 0.015	0.42 ± 0.049
2	0.62 ± 0.023	0.24 ± 0.051	0.53 ± 0.121
3	0.71 ± 0.034	0.35 ± 0.044	0.55 ± 0.023
4	0.66 ± 0.042	0.31 ± 0.049	0.55 ± 0.105
5	0.63 ± 0.058	0.27 ± 0.052	0.51 ± 0.100
6	0.73 ± 0.042	0.39 ± 0.047	0.63 ± 0.083
7	0.64 ± 0.049	0.29 ± 0.042	0.54 ± 0.023
8	0.68 ± 0.008	0.33 ± 0.022	0.60 ± 0.079
Ensemble classifier	0.85 ± 0.015	0.64 ± 0.029	0.60 ± 0.049

From: This figure was published in [24], copyright Elsevier (2018).

Other applications of SVM for diagnostics have been reported for machine condition monitoring and fault diagnosis [25]. Tool breakage and rolling bearing faults applications have also been reported [26,27].

Similar examples of using SVM for material defect detection were reported by using metal surface defect images as inputs [69], and also by using ultrasonic wave-related signals as inputs [70].

8.2.1.3 Prognostics applications

As mentioned in Section 8.1.1, the presence of material defects cannot be deterministically related to the state of damage in the material, at least until the behavior of the material is evaluated or tested. In this context, ML

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

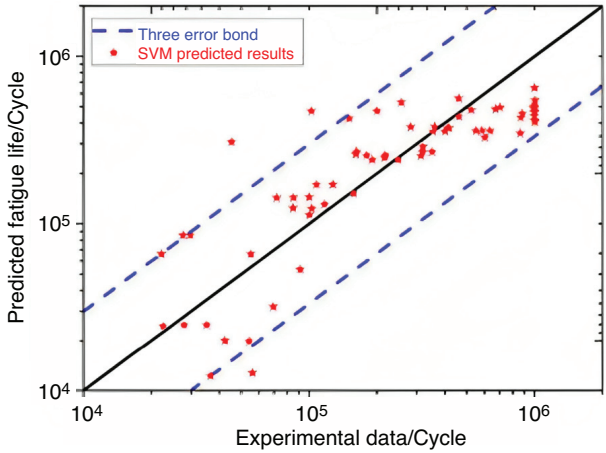


Figure 8.6 Variations in predicted fatigue lives by the support vector machines (SVM) model against the experimental data for AM stainless steel 316L produced using laser powder bed fusion [49]. *(From: This figure was published in [49], copyright Elsevier, 2021).*

methods such as SVMs have been used in relation to using data to estimate the material remaining useful life (RUL), given their as-manufactured states. In this context, data from numerical simulations to predict the RUL of 316L stainless steel were produced using an AM method (laser powder bed fusion) [49]. The authors used SVM modeling to find: (1) variations in the predicted accuracy and predicted fatigue life with the number of training data, and (2) variations in the predicted fatigue life with the AM parameters and fatigue loading. The database for the training was composed of six input and one output parameter. The inputs included AM process parameters (laser power, scan speed, hatch space, and powder layer thickness) and fatigue loading parameters (maximum stress and stress ratio). The output was the predicted fatigue life. The database consisted of more than one thousand datasets to train the ML model. Based on the proposed method, the fatigue lives were predicted and then compared with the experimental data from open publications (Fig. 8.6). Key results obtained with this method included the observation that powder layer thickness has a larger impact on fatigue life than the laser power used. In addition, the predicted fatigue life was found to be extremely sensitive to the maximum loading stress. The authors also used tree-based models, artificial neural networks, and adaptive neuro-fuzzy inference systems which are discussed in the following sections of this chapter. Additional applications of material fatigue life and RUL of cutting tools have been reported [50,51].

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

8.2.1.4 Advantages and challenges

SVMs are advantageous as they solve classification and regression problems while integrating nonlinearity [71]. SVMs work relatively well when there is a large separation between classes and there are high dimensional data spaces. The SVM algorithm also works effectively when the number of dimensions is greater than the number of samples. Moreover, SVMs are relatively stable because a small change in inputs does not greatly affect the computed hyperplane. Furthermore, it should be stated here in relation to manufacturing processes that SVM could be used for live classification, therefore, several efforts [72,73] have proposed the LaSVM which is an efficient online SVM algorithm that uses selective sampling on large datasets. Some disadvantages include its relatively large computational cost when larger datasets are used or when the data may have noise [74]. In addition, SVMs place data points above and below the decision surface, thus there is no probabilistic integration with the classifier which may reduce the overall confidence in the outputs produced by SVMs. Furthermore, it has been reported that it may be difficult to choose the appropriate kernel function which can therefore directly impact the training speed and accuracy.

8.2.2 Tree-based classifiers

Two of the most common tree-based classifiers include Decision Trees and Random Forests. Both are supervised ML algorithms used for classification and regression problems. With respect to Fig. 8.2, this type of AI model is an overlap between ML and PMs. Details and examples of tree-based classifiers are provided in this section.

8.2.2.1 Technical background

A Decision Trees model iteratively partitions the input using a set of rules until computing the final label. The initial node contains heterogenous data and thus is described as impure. The subsequent splits of the data reduce the impurity using the Gini impurity defined in Eq. (8.7) or Shannon's entropy in Eq. (8.8), where p_i represents the probability of randomly selecting a sample in class i. The overall goal is to ultimately get the purest homogenous output. Information gain is measured by taking the difference between the initial and final entropy. A decrease in entropy results in greater information gain and the higher the information gain, the better the split. Variations of these impurity metrics are used in various types of Decision Trees models

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

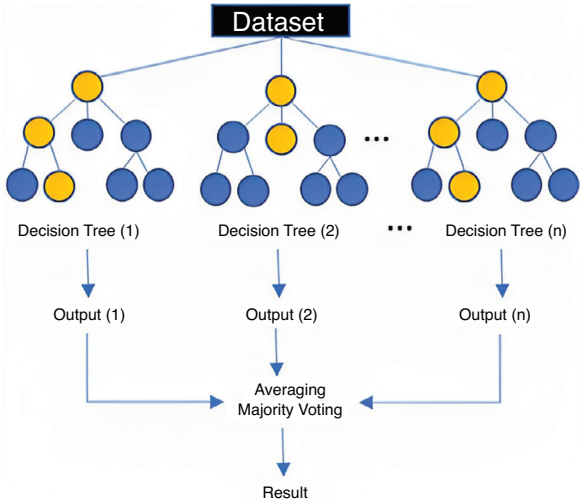


Figure 8.7 Exemplary structure of a Random Forest algorithm.

(e.g., ID3, CART, C4.5, and SLIQ) to determine the optimal Decision Trees [75,76]. The regular nodes of the Decision Tree algorithm represent features of the n-dimensional dataset, the leaf nodes represent the continuous or discrete label, and each branch represents the decision rule.

$$\text{Gini Index} = 1 - \sum_{i=1}^{n} p_i^2 \tag{8.7}$$

$$\text{Entropy} = -\sum_{i=1}^{n} p_i \log(p_i) \tag{8.8}$$

Random Forests leverage the usefulness of Decision Trees by using many of them for decision-making. This process is often called ensemble learning in which multiple ML algorithms are trained and combined for better prediction performance [77]. They exploit the idea that a large group of uncorrelated Decision Trees can produce more accurate and stable results than any of the individual Decision Trees, since a voting scheme is used to find the final output (Fig. 8.7). The algorithm integrates feature randomness in which the iterations of a subset of features is used, and bagging in which each Decision Tree is trained on a different random sample of the training data. Random Forests also calculate feature importance scores which enable the removal of unnecessary features to improve accuracy and

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

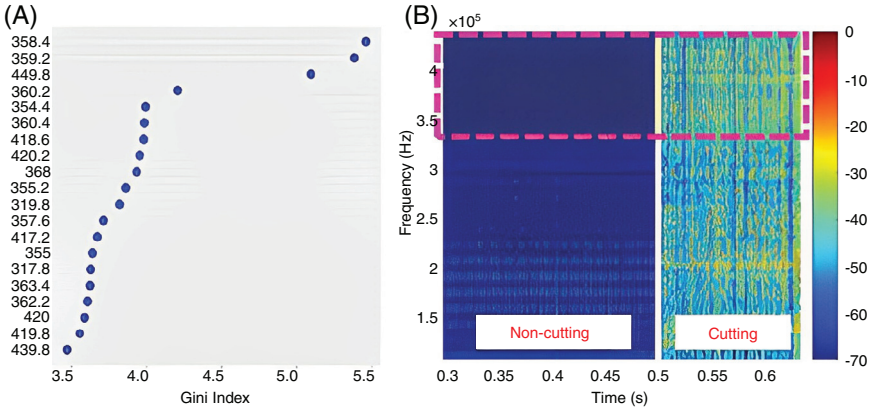


Figure 8.8 Representative classification results from applying a RF model to classify fiber orientation in natural fiber reinforced plastics leveraging Acoustic Emission monitoring data collected from the manufacturing process. (A) Frequency (kHz) band vs. Gini Index, and (B) a spectrogram representation of the noncutting phase versus the cutting phase (white line segmented); identified significant frequency bands are highlighted in the pink frame [30]. *(From: This figure was published in [30], copyright ASME, 2020).* For interpretation of the references to color in this figure legend, the reader is referred to the web version of this chapter.

decrease training time. Alternative tree-based ensemble methods include bagged Decision Trees and extra trees [78,79].

8.2.2.2 Diagnostic applications

The heterogeneous structure of natural fiber-reinforced plastics (NFRP) resulting from different fiber orientations and their complex multiscale structure introduce a distinct set of material removal mechanisms that inherently vary over time. This structure has an adverse effect on the surface integrity of machined NFRPs. Therefore, a real-time monitoring approach is desirable for timely intervention for quality assurance. A Random Forest (RF) algorithm was implemented to understand time-frequency patterns of acoustic emission (AE) sensing relating to different cutting speeds and fiber orientations of NFRP machining [30]. The sensors collected multimodal information with AE and vibration data during the cutting process. The RF model was trained to predict fiber orientation based on the spectral features of AE. The Gini Index was used as the impurity measure to split the samples into the respective classifications. The higher Gini Index indicated a greater contribution to the spectral feature (Fig. 8.8A). The spectrogram in Fig. 8.8B suggests that more energy is contained in such a frequency range (highlighted in the pink frame) during the cutting compared with the

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

noncutting phase. By classifying the fiber orientation, the authors concluded that alternative cutting mechanisms can be used accordingly to minimize fiber failures (fiber breakages, pull-out, and some inefficient shearing causing microfiber failures of the cellulose structure). Overall, the model produced an accuracy of 95% in distinguishing fiber orientations with respect to the cutting direction. Alternatively, tree-based methods were used to determine material properties and processing parameters [31]. Decision Tree-based methods have been further used to detect anomalies in manufacturing systems and predict operating conditions [28]. Furthermore, RF has been reported for machine health state classification [29].

8.2.2.3 Prognostics applications

Manufacturers have faced an increasing need for the development of predictive models that predict mechanical failures and the RUL of manufacturing systems or components. Therefore, a RF-based prognostic method for machine tool wear prediction was proposed [53]. The authors compare their results with other AI methods and concluded that RFs performed optimally. The data, in this case, was generated using a high-speed CNC machine in which cutting force, vibration, and AE was monitored. Although the RF model had longer training times, it had a lower error (higher accuracy) for tool wear prediction. Representative results are shown in Fig. 8.9.

Furthermore, Zhan et al. [49] computed the theoretical RUL using continuum damage mechanics and compared the results to a RF implementation. The authors concluded that the RF model works better than other AI models for the three AM materials. Furthermore, the predicted fatigue lives were found to be sensitive to the number of trees, the number of sensitivity parameters, and the depth of trees. Alternatively, tree-based models were leveraged to characterize elastoplastic fatigue damage fabricated through AM use [52]. Other applications include tree-based models for tool wear prediction [53].

8.2.2.4 Advantages and challenges

Decision Tree algorithms have some advantages with respect to other AI methods. For example, it does not require normalization or scaling of data, and missing values do not affect the process of the model development, and therefore, less time is needed for data preprocessing. Decision Tree algorithms are easier to visualize and interpret regardless of continuous or categorical variables. Moreover, Random Forest algorithms work with missing data, as they substitute such missing values with others that appear

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

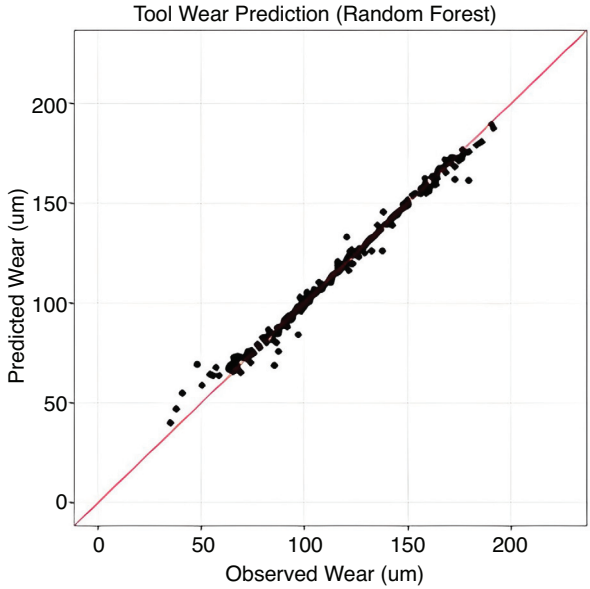


Figure 8.9 Comparison of observed and predicted tool wear [53]. *(From: This figure was published in [53], copyright ASME, 2017).*

most often. Both types of algorithms can also solve both classification and regression problems. Some disadvantages include instability because a small change in data input can cause rather significant changes in the computed output, thus making them sensitive to noise. Decision Trees also tend to be computationally expensive depending on the number of nodes and samples, therefore causing greater computational expense for the ensemble in Random Forests. Overfitting is another common issue with Decision Trees, which causes high variance in the outputs if an inaccurate input is given. Random Forests can reduce this variance by training on various samples and features to train the model.

8.2.3 Bayesian classifiers

Bayesian classifiers are probabilistic ML methods. Two of the most common Bayesian classifiers include Naïve Bayes and Bayesian networks. The backbone of the classifiers is the well-known Bayes theorem, which allows for revising predictions using additional evidence and provide a likelihood of a particular outcome.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

8.2.3.1 Technical background

Considering two random variables A and B, the Bayes theorem is expressed in Eq. (8.9), where $P(A)$ is the probability of A occurring, $P(B)$ is the probability of B occurring, $P(B|A)$ is the probability of B occurring given A, and $P(A|B)$ is the probability of A occurring given B. Often $P(A|B)$ is called the posterior distribution, $P(B|A)$ the likelihood, $P(A)$ the prior distribution, and finally $P(B)$ the marginalization.

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A) * P(B|A)}{P(B)} \tag{8.9}$$

In the ML context, the Bayes formula can be rewritten in terms of the feature space $X = \{x_1, x_2, \ldots, x_n\}$ and labels y (see Eq. (8.10)). Further simplification is done in Eq. (8.11) with the product of the likelihood distribution. For notational convenience, the denominator $P(x_1)P(x_2) \ldots P(x_n)$ is always constant, and thus it can be ignored to find the maximum probability for the posterior. The Naïve Bayes classifier in Eq. (8.12) uses the derived probability formulation for decision-making regarding the final label.

$$P(y|x_1,\ x_2, \ldots x_n) = \frac{P(y)\ P(x_1|y)\ P(x_2|y) \ldots\ P(x_n|y)}{P(x_1)\ P(x_2) \ldots\ P(x_n)} \tag{8.10}$$

$$P(y|x_1,\ x_2, \ldots x_n) = \frac{P(y)\ \prod_{i=1}^{n} P(x_i|y)}{P(x_1)\ P(x_2) \ldots\ P(x_n)} \tag{8.11}$$

$$P(y|x_1,\ x_2, \ldots x_n) = argmax_y\ P(y) \prod_{i=1}^{n} P(x_i|y) \tag{8.12}$$

Another Bayesian classifier is Bayesian networks (sometimes referred to as Bayesian Belief networks) which relax the independence assumption of the Naïve Bayes classifier. The algorithm uses directed acyclic graphs to encode dependence relationships amongst features. Bayesian Networks introduce a probabilistic relationship among a set of random variables where the output is a probability table associated with each random variable. Fig. 8.10 presents a simple architecture in which each random variable is a node, and the arcs represent the relationship between random variables. In Fig. 8.10, B is the parent node while A and C are the child nodes. A is conditionally dependent on B and C is conditionally dependent on B. Also, A is conditionally independent from C and C is conditionally independent from A. There are no loops when navigating from node to node. A probability distribution is associated with each node. The probability distribution table contains only

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

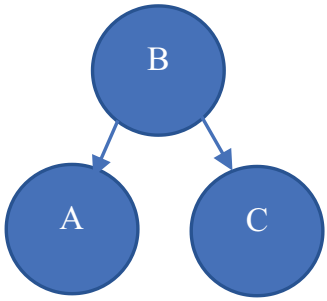


Figure 8.10 Simple Bayesian network architecture graph.

prior probabilities if the node does not have any parents as seen in Fig. 8.10 as node B.

Additionally, if the node has only one parent, then the table contains conditional probabilities as the case with nodes A and C. The respective tables for nodes A and C would contain $P(A|B)$ and $P(C|B)$. If the node had multiple parents, then the table would include conditional probabilities given all parent nodes. These probability tables are then used for finding the respective joint probability depending on the input conditions in Eq. (8.13). Additional details about Bayesian Networks can be found in related literature [80,81].

$$P(A, B, C) = P(A|B)\ P(C|B)\ P(B) \tag{8.13}$$

8.2.3.2 Diagnostics applications

As composite materials undergo environmental exposure, changes in conditions cause chemical reactions to weaken the material. A Bayesian network to monitor the environmental degradation of composite materials was proposed [37]. The initial setup of the model included defining the nodes and the relationship between the nodes using expert knowledge. Data regarding initial diffusion parameters, loss of glass fibers, and matrix plasticization were determined through experimental data while changes in water concentration and change in strain were found from a parametric fit based on structural finite element analysis. These nodes were used as input to the network. The authors further added Gaussian noise to account for the uncertainty in the data. The developed Bayesian network by authors portrays the nodes of material properties along with the inputted noise. Once the model was developed, the authors developed a series of posterior distributions for various parameters related to composite degradation. These included moisture content, stiffness in the fiber direction, and strain in the fiber direction.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

This particular model was chosen by the authors because given evidence of one of the parameters, the distribution of other parameters can also be obtained (e.g., if the strain is known, then the stiffness can be adjusted). A resulting probability distribution of moisture for 5–25 years. The distributions of parameters (moisture content, stiffness, and strain) obtained could then be used for inspection planning of the composite structures. Alternatively, Bayesian approaches have been used for defect detection in graphite/epoxy laminates such as delamination, crack, and holes [82] and faults in 3D Printing machines [33]. Also, on fault diagnosis of bearings (e.g., out race fault at center, out race fault at orthogonal, out race fault at opposite [34], and gearbox faults [35]).

8.2.3.3 Advantages and challenges

Naïve Bayes can handle categorical input variables and solve multiclass problems [83]. The computation is usually fast because there are no iterations or backpropagation and simply depends on conditional probabilities. Naïve Bayes classifiers assume that all features are independent, and this is considered "naïve" because it is a rather strong assumption for real-world data. If the assumption is true for the scenario, then it can be an advantage otherwise, this assumption may hinder the performance of the algorithm. Naïve Bayes methods have a zero-frequency problem in which the algorithm assigns zero probability to the categorical variables which are not available in training but are available in testing. Furthermore, Bayesian networks are useful when a model is needed to visualize the structure of probabilities and dependencies [84]. The network can also provide information about feature importance and how much it contributes to the classification. Some disadvantages include the construction of the network, which is based on trial and error, therefore the networks may take a while to construct. Often expert knowledge is needed to determine the dependencies between random variables. Bayesian Networks also do not define cyclic relations therefore may create increased assumptions on variable patterns.

8.2.4 Unsupervised machine learning

Clustering is a common unsupervised learning method in which the goal is to find underlying patterns, structures, and/or groupings of unlabeled data. Unlike supervised methods in which there can be percent error or accuracy metrics to determine the performance of the model, unsupervised techniques do not have a specific criterion to validate their correctness.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Techniques like cluster proximity, densities, and compactness can help in validating the results depending on the ground truth.

8.2.4.1 Technical background

K-means is a partition clustering method in which k different groups are created and each data point is classified into only one group. The cluster designation is determined by minimizing the sum of squared distances between the point and the centroid. The objective function to be minimized is given in Eq. (8.14), where k is the number of clusters, n is the number of points, and x_i is the data point.

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} ||x_i - c_j||^2 \tag{8.14}$$

Other clustering algorithms that incorporate probabilistic modeling and neural networks are Gaussian mixture modeling (GMM) and self-organizing maps (SOM), respectively. K-means orients a circle around the center of each cluster with the defined radius. GMM relaxes this assumption by considering the variance of the samples in each cluster thus allowing for oblong clusters. Furthermore, K-means provides discrete categories for each sample while GMM provides "soft" clustering in which each sample is given a probability of belonging to a possible cluster. A series of Gaussian curves are created and then the likelihood of the sample coming from the kth Gaussian is determined iteratively using the expectation-maximization algorithm [85]. Moreover, SOM are a type of artificial neural network (ANN) that is used for unsupervised learning. SOMs use a neighborhood function to understand the topological properties of the input feature space [86]. Unlike traditional ANNs, SOMs do not learn weights using optimization techniques, instead, they use competitive learning. Additional unsupervised techniques are presented in the related literature [87,88] which include density-based methods, hierarchical-based methods, and grid-based methods.

8.2.4.2 Diagnostics applications

Unsupervised clustering to detect different types of damage on a constrained composite was reported [89]. In this work, supervised (K-nearest neighbors) and unsupervised learning (K-means) were compared. Polyester resin and glass/polyester unidirectional specimens were used for the experiment and loaded with tensile load. Further analysis was conducted on single-fiber composites. The two classes of signals that were identified were A-type

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

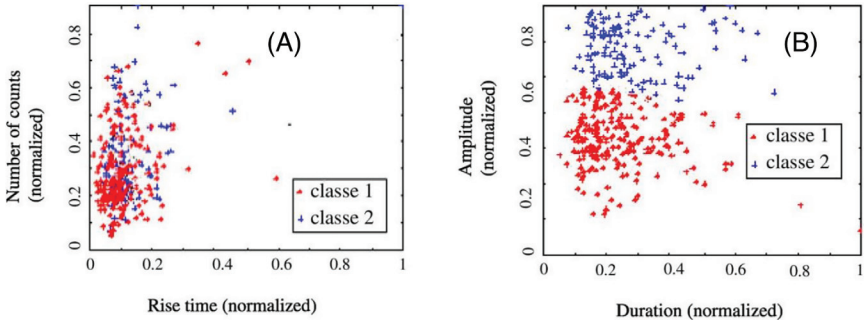


Figure 8.11 Clustering results for damage in composite materials using 2D projections of Acoustic Emission data features: (A) in the counts-rise time projection, and (B) in the amplitude-duration projection [89]. *(From: This figure was published in [89], copyright Elsevier, 2004).*

and B-type corresponding to fracture matrix and interface decohesion, respectively with the input data as acoustic emission. The "k" in K-means was thus inputted as two to the model. The work uses the signal feature space and respective means of the centroids to verify the clustering. Fig. 8.11 portrays the signals in a scatter plot with respective classes in two feature projections. After analyzing the feature behavior, the authors conclude class 1 is referring to A-type and class 2 is B-type. When the authors added signals related to fiber failures from a single-fiber composite, they found it difficult to distinguish between fiber failure and the other two previous classes. The study concluded that the analysis proved effective and can be used for the classification of data from future tests. Alternatively, incremental clustering to assess damage mechanisms of glass fiber-reinforced composite materials [39] was used, and a clustering fusion method to detect signs of failure in high-performance carbon fiber-reinforced thermoset matrix composites [40] was proposed. Moreover, unsupervised waveform clustering to classify damage in carbon fiber composites was used [41]. Machine tool health was also assessed with unsupervised learning [42].

8.2.4.3 Advantages and challenges

As discussed, labeling datasets with the appropriate damage types can be difficult and/or time-consuming. Therefore, unsupervised clustering provides an opportunity to determine the data classes through a mathematical representation. Unsupervised learning also has the capability of discovering hidden patterns that may not be visible to the human eye. These methods are less complex than supervised techniques and thus can be reasonably

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

easier to visualize. However, since the true classes are unknown, unsupervised techniques require additional sensing and validation to ensure the correct patterns are identified [90]. This can further add time and cost with testing machinery or materials and might result in clustering that is less accurate if the validation is not correctly identified. Since the model does not learn from prior knowledge, the clustering output may not provide accurate results unless there is validation or domain expert to verify the results. Algorithmically, K-means assumes the clusters are spherical in shape however if the clusters are oblong or alternative geometric shapes, the algorithm does not perform optimally [91]. Each algorithm may assume independence of features, compactness, or other metrics thus selecting the right one is essential for domain-specific applications.

8.2.5 Deep learning methods

Neural networks are inspired by the human brain in which neurons feed information to the network, and the weights between nodes act as connections between neurons. The interconnected group of artificial neurons uses mathematical representations to process complex patterns in the data to output the label or probability. This interconnection between layers is called an artificial neural network (ANN).

8.2.5.1 Technical background

The first layer of an ANN inputs the raw data input, similar to a biological optic nerve which receives signals. Each proceeding layer receives the previous output and applies a function to the data. The respective output of the layer is then transported further down the system. The mathematical representation for a hidden layer is shown in Eq. (8.15), where σ is the activation function, N is the number of input neurons, W_{ij} are the weights, x_j are the inputs to the input neurons, and T_i are the threshold terms of the hidden neurons. Finally, the last layer produces the output of the network.

$$h_i = \sigma\left(\sum_{j=1}^{N} W_{ij}x_j + T_i\right) \tag{8.15}$$

This layering concept is portrayed in Fig. 8.12 with the respective inputs, hidden layers, weights, and outputs. With this basic architecture in mind, there are several complex neural networks that can be generated. These include multilayer perceptron's (MLPs), convolutional neural networks (CNNs), and auto-encoders among others [92–94].

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

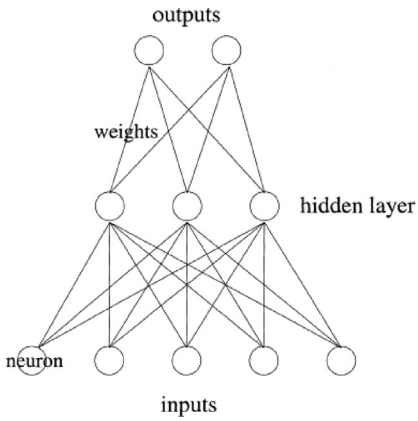


Figure 8.12 Architecture of an artificial neural network [95]. *(From: This figure was published in [95], copyright Elsevier, 2003).*

Neural networks are often described by the number of layers or depth. Accordingly, a multilayer perceptron (MLP) is built on the concept of ANN which it has multiple hidden layers. As a result, the models are referred to as deep learning. The models can also be described in terms of their activation functions, parameters, and other variation among the layers. Moreover, CNNs use the concept of kernels to extract relevant features from the given image. The filters are convolved with the image to result in a reduced feature space in each layer. The previous architectures mentioned are supervised learning techniques in that the labels are inputted with the raw data for training. Autoencoders are unsupervised neural networks that create a compressed representation of the raw data. After training the autoencoder, if the inputs are independent, the autoencoder will output the deviation of the selected testing dataset compared to the training data. The bottleneck architecture of an autoencoder is advantageous in reducing overfitting and simply 'memorizing' the given input data. Furthermore, adaptive neuro-fuzzy inference system (ANFIS) methods are a combination of neural networks and fuzzy logic principles [96]. The final prediction in an ANFIS model is determined by simple ANN proceeded by a series of membership functions and IF-THEN rules. Fig. 8.13 portrays the architecture of a traditional ANFIS model.

8.2.5.2 Diagnostics applications

Metal additive manufacturing (AM) methods require quality inspection to ensure product improvement. However, the traditional inspection process

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

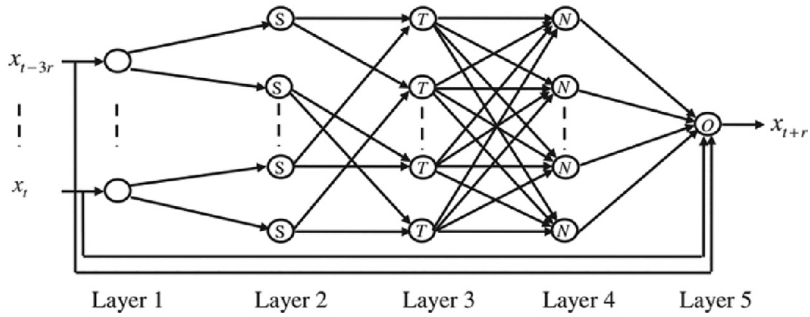


Figure 8.13 Architecture of an adaptive neuro-fuzzy inference system (ANFIS) model with S as the sigmoid function, T as the fuzzy T-norm operator, N as the normalization operation, and O as the output [97]. *(From: This figure was published in [97], copyright Elsevier, 2012).*

relies on manual recognition, which could suffer from low efficiency. A convolutional neural network (CNN) approach to inspect damage during metal AM was proposed [43]. The authors used specimens created through a laser metal deposition (LMD) process in which a melt pool is created from the laser beam across the sample surface and the powder material is injected into the melt pool. Specifically, specimens were manufactured with AISI 304 stainless steel, AISI 316 stainless steel, Ti-6Al-4V, AlCoCrFeNi, and Inconel 718 alloys. The work focuses on classifying damage common to LMD such as good quality, crack, gas porosity, and lack of fusion during the inspection. Once images of the four types of defects were taken (Fig. 8.14), a series of preprocessing techniques were applied to increase the dataset (e.g., rotation, flipping, cropping, etc.).

The authors tried various CNN architectures and compared the computational time and validation accuracy. Ultimately, the authors found that the model presented in Fig. 8.15 provided a robust solution for the quality inspection of metal additive manufacturing parts. Alternatively, CNNs to detect processing defects for laser powder bed fusion AM were reported [44], as well as CNNs to characterize fault detection (abnormal, normal) during manufacturing [45]. Methods such as long–short-term memory (LSTM) networks and gated recurrent unit (GRU) networks were used [46–48].

8.2.5.3 Prognostics applications

Changes in the fatigue response can directly affect the processing and postprocessing of materials. Therefore, an ANFIS to predict the high cycle fatigue life of laser powder bed fusion stainless steel was reported [55].

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

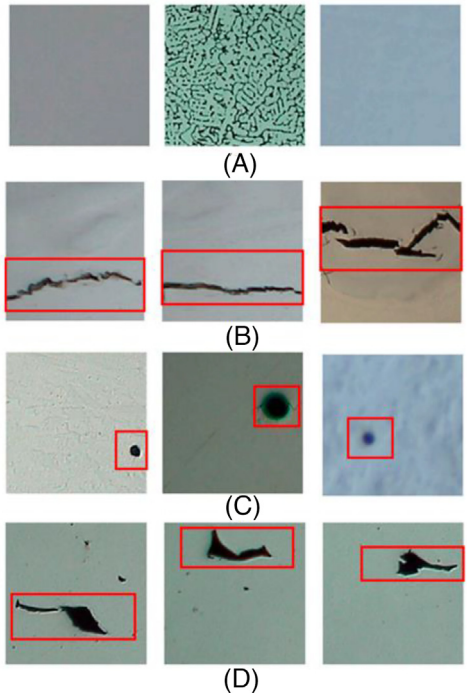


Figure 8.14 Examples of laser metal deposition images used in a convolutional neural network (CNN) approach resulting in specimens of: (A) good quality, (B) crack, (C) gas porosity, and (D) lack of fusion [43]. *(From: This figure was published in MDPI [43], modified from Creative Common CC License 4.0, 2020).*

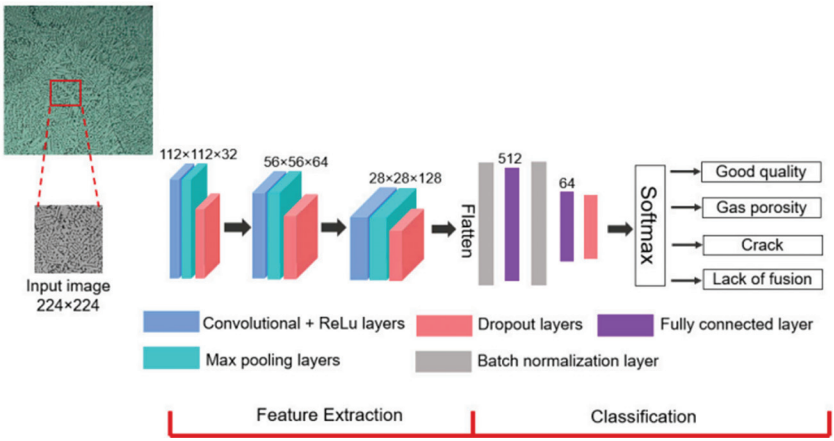


Figure 8.15 Convolutional neural network (CNN) for metal additive manufacturing quality [43]. *(From: This figure was published in MDPI [43], modified from Creative Common CC License 4.0, 2020).*

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Damage mechanisms such as crack initiation and deformation were integrated into the model along with a variety of material properties and cyclic stress. Stainless steel 316L was manufactured with a laser bed powder fusion process with various processing patterns, which resulted in different defect characteristics. Some samples were subjected to postprocessing annealing. Two models were curated to represent the processing and postprocessing conditions. The defects being classified were microstructure-driven crack initiation, defect-driven crack initiation, and ratcheting-dominated deformation. The inputs to the process-based model were processing and postprocessing parameters such as laser power, scan speed, layer thickness, and temperature. The "property-based" model used inputs pertaining to ultimate tensile strength and elongation to failure. The ANFIS models were designed with Gaussian-type membership functions and the results were compared with the author's prior work. Fig. 8.16 shows that the predicted S-N curves fit well the experimental data across the different failure modes. The authors concluded that the model was able to account for variations in the dataset due to fatigue scatter. The authors further compared the model with results from other literature to verify the center clusters for the ANFIS model. The authors further concluded that the "cluster center," or alpha from the membership functions, verification allowed the ANFIS to effectively find the fracture behaviors of the samples and that their work was able to successfully predict the fatigue life of stainless steel. Alternative applications of deep learning networks for material prognostics have been used to detect elastoplastic fatigue damage and steel, respectively [49,52]. Applications for machine health monitoring such as tool wear have been also reported [53,54].

8.2.5.4 Advantages and challenges

Neural networks provide a methodology to autonomously extract features from the dataset. This can be beneficial in capturing the important facets of the input data. One of the main advantages of DL methods is their parallel processing capability [98]. Most of these models require, however, large data sets, therefore the parallel processing provides computational efficiency as opposed to traditional ML algorithms. The training samples can be spread across different processors and compute updates to the neural network in parallel. However, if parallel processing hardware is unavailable, it can become cumbersome to train these large models. Accordingly, there is a data dependency with neural networks because large amounts of data are needed to train the model, however, the user must be watchful of errors

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

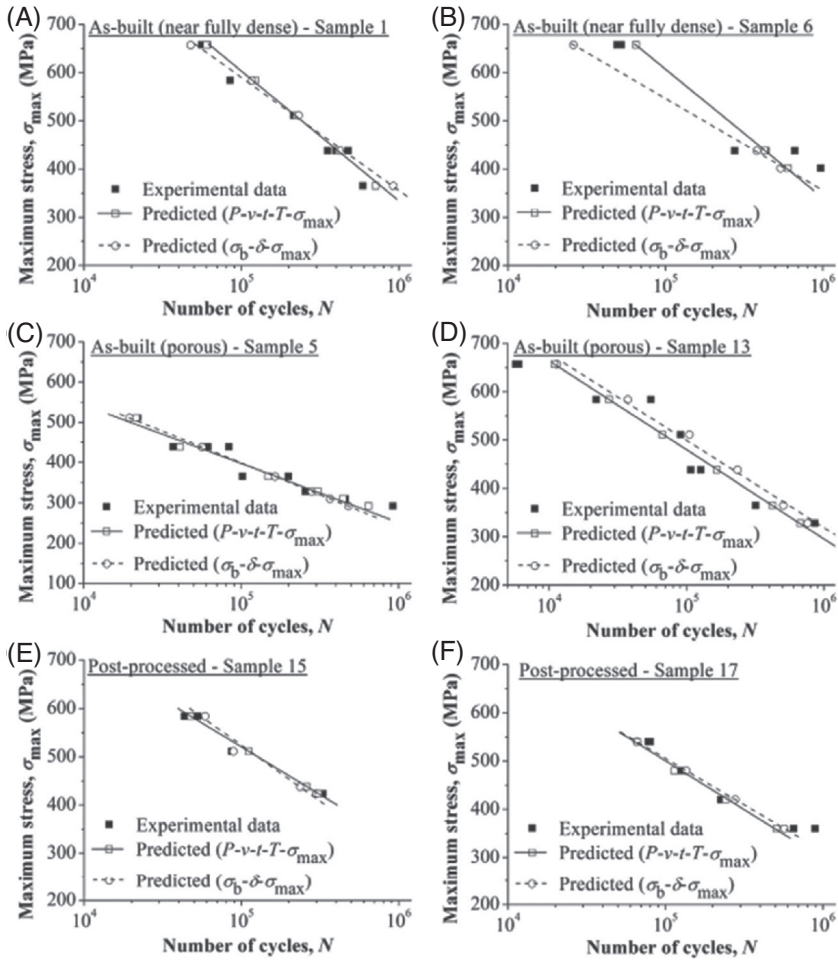


Figure 8.16 Comparison of experimental and predicted S-N data for (A), (B) near fully dense samples, (C), (D) porous samples, and (E), (F) postprocessed samples [55]. *(From: This figure was published in [55], copyright Elsevier, 2019).*

in the data which will affect the output [99]. Furthermore, there are no specific criteria for determining the structure of neural networks. The ideal network structure is achieved through experience and trial and error. For this reason, neural networks are often classified as "black box" models because it is difficult to see the internal workings of each layer. Similar to other supervised methods, the issue with some DL models includes the need for data labels. Additional sensing methods or expert knowledge may be needed to validate signals.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

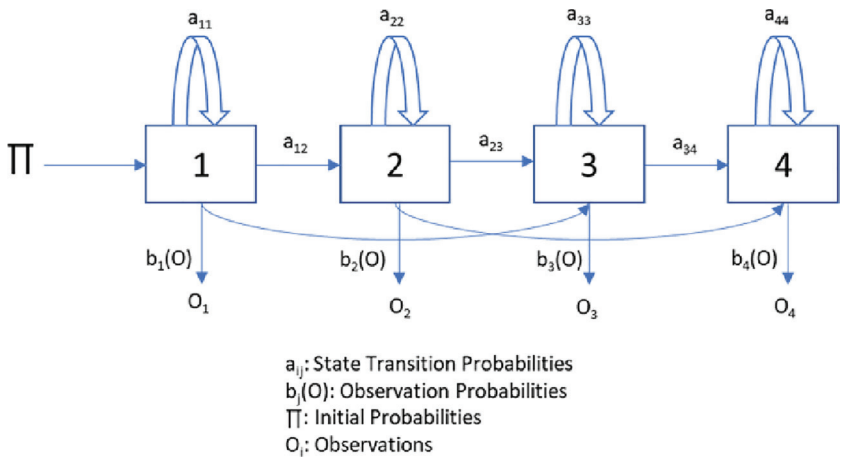


Figure 8.17 Hidden Markov Models (HMM) structure with prior distribution (π), state transition probabilities (a_{ij}), and observation probabilities ($b_j(O)$).

8.2.6 Hidden Markov models

Hidden Markov Models (HMMs) are probabilistic ML algorithms in which the observations are the training data and the number of hidden states is a hyperparameter. It is a stochastic process with an underlying stochastic process (hidden states) that is not observable. The models rely on the Markovian assumption in which the future is independent of the past given the present, and therefore historical information is not needed for future state predictions [100,101].

8.2.6.1 Technical background

Hidden Markov Models assume that transitions can be made from any state (i.e., from any damaged state in the context of this chapter) to any other state, thus completing a state transition. In order to make a prediction of the future state, there are three key pieces of information needed: the transition matrix (probability of transitioning to a new state given the current state), the emission matrix (the probability of transitioning to an observed state given a hidden state), and the initial state distribution (the initial probability of transitioning to a hidden state). Fig. 8.17 visualizes the three key pieces of information to form a sequence in a HMM. The model is defined accordingly with these three pieces of information. Once the model is trained, the HMM can determine: (1) what is the likelihood of a sequence of observations, (2) what is the optimal hidden state sequence given a model and observation sequence, and (3) what are the optimal

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

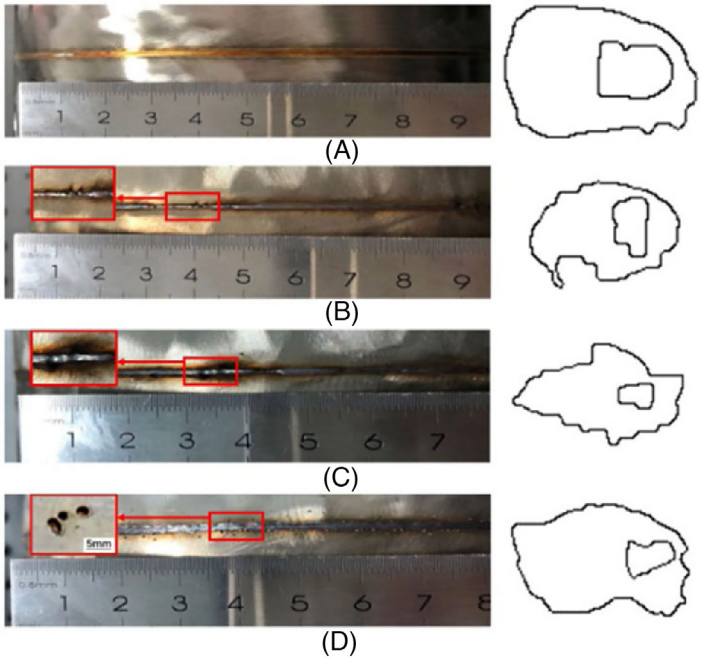


Figure 8.18 Samples of laser welding (*left* is weld seam and defects; *right* is extracted keyhole and full penetration hole contour). (A) Sound well weld; (B) lack of fusion; (C) burn through; and (D) porosity [109]. *(From: This figure was published in IEEE XPLORE [109], modified from Creative Common CC License 4.0, 2020).*

parameters to optimize the probability of observations given a sequence of observations and hidden states. Variations of an HMM provide various distribution assumptions and optimization techniques [102–104]. There have also been efforts towards combining HMM with other ML techniques such as ANNs, SVMs, and unsupervised clustering [105–108].

8.2.6.2 Prognostics applications

High-power laser welding at high temperatures can directly affect the quality of laser welding. Therefore, a HMM approach to classify the defect around the keyhole of the specimen was used [109]. The keyhole is subjected to metal vapor and spatter which makes it difficult to assess the quality of the material with a human eye. Thus, data was collected during the weld process by an image acquisition system to train the HMM. The defects include porosity, lack of fusion, burn-through, and sound well weld. Fig. 8.18 shows the images during the weld process with the respective

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

contour extracted during preprocessing to train the model. The authors developed grayscale preprocessing techniques to separate the keyhole region from the penetration hole. The results in Table 8.4 indicate high accuracy for classification for various defect detection. Therefore, the authors concluded that the results show the system can accurately identify the quality of welding and can effectively identify the defects related to the penetration state and the generation of porosity. Alternatively, HMMs have been applied to determine defect degradation [110] and machine condition monitoring [58,60,61].

8.2.6.3 Advantages and challenges

Hidden Markov Models have been used in a variety of applications and therefore constitute a well-studied probabilistic model for learning and inference. HMMs allow for unobserved variables which provides greater freedom in the application. Due to the Markov assumption, the HMM does not consider the series of states leading up to the state [111]. Furthermore, the time at a given state is not explicitly defined. Alternative variations of HMM may provide greater flexibility in relaxing these assumptions [58,112]. Depending on the problem the HMM is solving, the Viterbi algorithm or forward/backward algorithm are often used. These can become computationally expensive with large datasets as the model learns very long training sequences of data.

8.2.7 Filtering-based approaches

Particle filters are Monte Carlo algorithms used to represent a posterior distribution of a stochastic process [113]. The particles represent probabilities and then can be used to form a state-space model. This model is known to work with nonlinear and nongaussian systems thus can be applied to a variety of applications.

8.2.7.1 Technical background

First, the particles are drawn from an initial distribution, and they are then propagated forward and updated by the importance weights. Next, the particles are resampled to form a prediction. This is done until a termination criterion is reached. Particle filtering is used to solve many algorithms such as HMMs, and other nonlinear filtering problems. A block diagram of particle filtering is portrayed in Fig. 8.19 with the respective steps for initialization, normalization, resampling, and output. For a linear system, a Kalman filter may provide better results. In Kalman filtering, the first step is initializing

This book belongs to COMEX COMEX (comexcom83@gmail.com Copyright Elsevier 2024

Table 8.4 Results for weld defect detection experiments [109].

	The number of test samples	Proportion to total	The number of false detections	Accuracy rate
(a) Experiment data of 2-class detection				
Sound well weld	5000	56.56%	0	100%
Defective	3840	43.44%	277	92.79%
Total	8840	100%	277	96.86%
(b) Experiment data of porosity detection				
Sound well weld	5000	83.13%	59	98.82%
Porosity	1015	16.87%	143	85.91%
Total	6015	100%	202	96.64%
(c) Experiment data of penetration detection				
Sound well weld	5000	63.90%	8	98.12%
Burn through	1522	19.45%	55	99.47%
Lack of fusion	1303	16.65%	45	99.54%
Total	7825	100%	108	98.62%
(d) Experiment data of 4-class detection				
Sound well weld	5000	56.56%	77	94.24%
Burn through	1522	17.22%	156	98.36%
Porosity	1015	11.48%	214	81.38%
Lack of fusion	1303	14.74%	148	92.86%
Total	8840	100%	595	93.27%

From: This table was published in IEEE XPLORE [109], modified from Creative Common CC License 4.0 (2020).

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY

Copyright Elsevier 2024

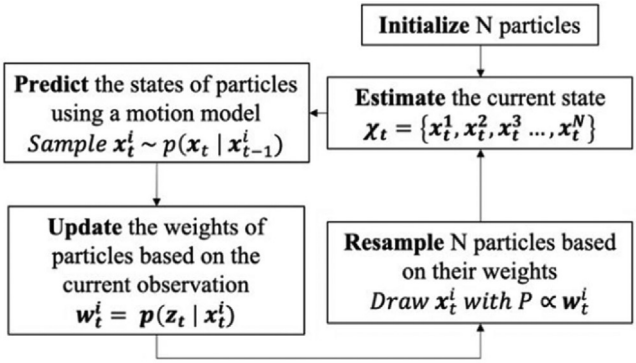


Figure 8.19 Block diagram of particle filtering [116]. *(From: This figure was published in MDPI [116], modified from Creative Common CC License 4.0, 2021).*

the system state and equations. Next, the system is reinitialized to estimate the second measurement. When the third measurement is received, the state estimate is propagated forward and the Kalman gain is calculated. Finally, the Kalman filter estimates the system state and error covariance matrix [114]. In a particle filtering approach, the particles are drawn randomly, however in an unscented Kalman filter, the points are drawn from a specific algorithm [115]. Particle filters are more flexible as they do not assume linearity and the Gaussian nature of noise in data, but they are more computationally expensive.

8.2.7.2 Prognostics applications

Bearings are an essential component of rotational machinery in manufacturing. Prognostics for bearings is especially important to reduce downtime and uphold efficiency in the manufacturing system. Thus an extended Kalman filter to predict the RUL of bearings under various operating conditions was used [63]. The authors initially use time and frequency domain signal features to quantify vibrations. Two accelerometers and one temperature sensor were used. After initializing the variables, the next state is predicted, and the model recursively inputs the next measurement to update the prediction using Kalman Gain. The current value is then extrapolated out to the specified threshold for the RUL prediction and confidence interval. The results indicate that the predictions are within the 20% bound of RUL. Alternative applications of particle filtering have been applied to tool wear estimation and equipment RUL [63–65] and damage on gear plates [117].

8.2.7.3 Advantages and challenges

Particle filtering is scalable, therefore it can work on hardware capable of parallel processing thus providing an advantage for real-time processing [62,117].

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

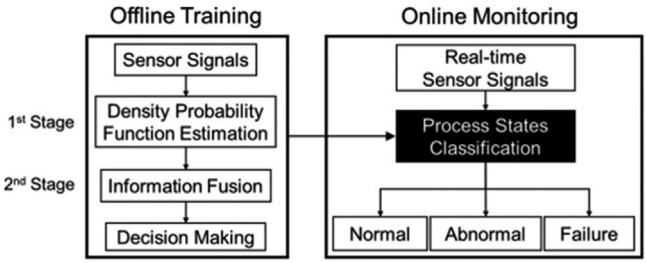


Figure 8.20 Damage model for real-time monitoring of additive manufacturing processes [119]. *(From: This figure was published in [119], copyright ASME, 2015).*

The algorithm is also used to approximate posterior distributions and thus can be integrated with other ML algorithms pertaining to expectation. The algorithm works for high-dimensional data as well. Particle filtering is nondeterministic therefore at the resampling step, the set chosen will likely gain particles [118]. For the same input, the algorithm can result in different results each time it is run. Thus, it is difficult to produce consistent results and to measure the relative accuracy of the number of particles and the confidence of the readings.

8.3 Challenges and opportunities for AI methods for damage diagnostics and prognostics

The examples of materials damage diagnostics and prognostics presented in this chapter justify the need to increase the awareness of the capabilities of AI methods in this field. Quality control, standardization, equipment and material development, and applications in demanding areas including medicine, aerospace, autonomous vehicles, etc. all depend on the efficiency and reliability in detecting defects and damage. In this context, the issue of real-time implementation of AI algorithms for materials damage diagnostics and prognostics, e.g., in manufacturing is becoming more needed. The reduction of costs, trial and error approaches, the improvement of design, and the reduction of the time it takes to bring new products to market can benefit from real-time monitoring approaches both at the production and usage phases. Some related efforts towards real-time applications using the algorithms presented in this chapter were already presented [54,62,119]. A representative framework for real-time monitoring of additive manufacturing processes is presented in Fig. 8.20, which proposes an approach that combines offline training and online monitoring. Online

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

analysis poses its own sets of challenges including uncertainty considerations such as statistical limits, model limits, environmental randomness, and human error. Furthermore, certain algorithms require large data inputs, therefore may not be applicable to real-world scenarios with sparse systems. With online monitoring, it is essential to have a data schema that supports smart archival and model training. Cloud computing alone does not satisfy the storage of large datasets, rather smart archival and data schemas must be implemented to organize raw and metadata. In this context, AI methods enable implementations in already produced edge and cloud computing approaches which can be furthermore useful in big data applications [120]. In fact, there is considerable work in the related literature that reports progress towards optimizing these methods for streaming data presented in the literature [121–126].

Furthermore, many of the results presented in this chapter remain application specific. This relates to the fact that the application of AI methods for materials damage evaluation does hinder the capability to use both hardware and software resources, which in some cases it may not be readily available. The entire field of quality control has benefited from standard operating procedures and standards for training, application, results visualization, and reporting, as well as for the safe and reliable use of associated equipment, practically allowing technician-level knowledge to be sufficient. Moreover, it is a fact that the use of AI might pose hurdles for existing practitioners and organizations to adopt new tools and methods, especially since the use of AI methods is still a topic of research and development. To counter this, there is an increasing number of resources available both commercially and as open software, while training and certification can now be easily found via both noncredit and credit curricula. Therefore, it is expected that the use of AI methods in the industry will keep increasing as the workforce will become more trained in using them and as the demands will continue rising. Interestingly, governments and research funding organizations are investing currently heavily in the areas of smart and connected future manufacturing, which demonstrates the widespread awareness regarding the opportunities of using AI methods in this field.

8.4 Conclusion

The chapter provided a brief overview of the most frequently used AI methods for material damage diagnostics and prognostics. In general, AI methods provide a mathematical and numerical approach to postprocess

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

data collected in relation to materials characterization and qualification. In this sense, AI enhances existing capabilities to analyze, e.g., image data and improve the probability of defect defection. In parallel, the ongoing digitization of manufacturing systems is creating simultaneously the framework for the increasing use of autonomous, robust systems to perform quality control. Overall, locating, identifying, and classifying damage is particularly complex due to the various operating conditions and environments that all materials are subjected to. Therefore, designing datasets for training AI models is especially important to ensure the correct conditions are being predicted. This process requires collaborations between the data scientist and experimentalist which is not always straightforward to implement. No matter what the challenges are, the future outlook is positive based on the examples shown which demonstrate clearly that there is a great potential of using AI methods for damage diagnostics and prognostics.

References

[1] J.J. de Pablo, B. Jones, C.L. Kovacs, V. Ozolins, A.P. Ramirez, The materials genome initiative, the interplay of experiment, theory and computation, Curr. Opin. Solid State Mater. Sci. 18 (2) (2014) 99–117. https://doi.org/10.1016/j.cossms.2014.02.003.

[2] M.C. Roco, The long view of nanotechnology development: the national nanotechnology initiative at 10 years, J. Nanopart. Res. 13 (2) (2011) 427–445. https://doi.org/10.1007/s11051-010-0192-z.

[3] R. Ramprasad, R. Batra, G. Pilania, A. Mannodi-Kanakkithodi, C. Kim, Machine learning in materials informatics: recent applications and prospects, npj Comput. Mater. 3 (1) (2017) 54. https://doi.org/10.1038/s41524-017-0056-5.

[4] K.T. Butler, D.W. Davies, H. Cartwright, O. Isayev, A. Walsh, Machine learning for molecular and materials science, Nature 559 (7715) (2018) 547–555. https://doi.org/10.1038/s41586-018-0337-2.

[5] J.E. Saal, S. Kirklin, M. Aykol, B. Meredig, C. Wolverton, Materials design and discovery with high-throughput density functional theory: the open quantum materials database (OQMD), JOM 65 (11) (2013) 1501–1509. https://doi.org/10.1007/s11837-013-0755-4.

[6] D. Hu, R. Kovacevic, Sensing, modeling and control for laser-based additive manufacturing, Int. J. Mach. Tools Manuf. 43 (1) (2003) 51–60. https://doi.org/10.1016/S0890-6955(02)00163-3.

[7] S.K. Everton, M. Hirsch, P. Stravroulakis, R.K. Leach, A.T. Clare, Review of in-situ process monitoring and in-situ metrology for metal additive manufacturing, Mater. Design 95 (2016) 431–445. https://doi.org/10.1016/j.matdes.2016.01.099.

[8] J.J. Lewandowski, M. Seifi, Metal additive manufacturing: a review of mechanical properties, Annu. Rev. Mater. Res. 46 (1) (2016) 151–186. https://doi.org/10.1146/annurev-matsci-070115-032024.

[9] J.M. Chacón, M.A. Caminero, E. García-Plaza, P.J. Núñez, Additive manufacturing of PLA structures using fused deposition modelling: effect of process parameters on mechanical properties and their optimal selection, Mater. Design 124 (2017) 143–157. https://doi.org/10.1016/j.matdes.2017.03.065.

[10] Y. Kok, et al., Anisotropy and heterogeneity of microstructure and mechanical properties in metal additive manufacturing: a critical review, Mater. Design 139 (2018) 565–586. https://doi.org/10.1016/j.matdes.2017.11.021.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[11] R. Talreja, A. Kelly, A continuum mechanics characterization of damage in composite materials, Proc. Math. Phys. Eng. Sci. 399 (1817) (1985) 195–216. https://doi.org/10.1098/rspa.1985.0055.
[12] R. Talreja, C.V. Singh, Damage and Failure of Composite Materials, Cambridge University Press, Cambridge, 2012.
[13] D. Socie, Multiaxial fatigue damage models, J. Eng. Mater. Technol. 109 (4) (1987) 293–298. https://doi.org/10.1115/1.3225980.
[14] D.L. McDowell, F.P.E. Dunne, Microstructure-sensitive computational modeling of fatigue crack formation, Int. J. Fatigue 32 (9) (2010) 1521–1542. https://doi.org/10.1016/j.ijfatigue.2010.01.003.
[15] J.Y. Buffière, S. Savelli, P.H. Jouneau, E. Maire, R. Fougères, Experimental study of porosity and its relation to fatigue mechanisms of model Al–Si7–Mg0.3 cast Al alloys, Mater. Sci. Eng.: A 316 (1) (2001) 115–126. https://doi.org/10.1016/S0921-5093(01)01225-4.
[16] J. Cuadra, P.A. Vanniamparambil, K. Hazeli, I. Bartoli, A. Kontsos, Damage quantification in polymer composites using a hybrid NDT approach, Compos. Sci. Technol. 83 (2013) 11–21. http://dx.doi.org/10.1016/j.compscitech.2013.04.013.
[17] B.J. Wisner, P. Potstada, V.I. Perumal, Konstantinos P. Baxevanakis, M.G.R. Sause, A. Kontsos, Progressive failure monitoring and analysis in aluminium by in situ nondestructive evaluation, Fatigue Fract. Eng. Mater. Struct. 42 (9) (2019) 2133–2145. https://doi.org/10.1111/ffe.13088.
[18] M. Seifi, A. Salem, J. Beuth, O. Harrysson, J.J. Lewandowski, Overview of materials qualification needs for metal additive manufacturing, JOM 68 (3) (2016) 747–764. https://doi.org/10.1007/s11837-015-1810-0.
[19] C.L.A. Leung, S. Marussi, R.C. Atwood, M. Towrie, P.J. Withers, P.D. Lee, In situ X-ray imaging of defect and molten pool dynamics in laser additive manufacturing, Nat. Commun. 9 (1) (2018) 1355. https://doi.org/10.1038/s41467-018-03734-7.
[20] A. Rytter, Vibrational based inspection of civil engineering structures. Department of Building Technology and Structural Engineering, Aalborg University, 1993.
[21] H. Abdi, L.J. Williams, Principal component analysis, Wiley Interdiscip. Rev. Comput. Stat. 2 (4) (2010) 433–459.
[22] J. Benesty, J. Chen, Y. Huang, On the importance of the Pearson correlation coefficient in noise reduction, IEEE/ACM Trans. Audio Speech Lang. Process. 16 (4) (2008) 757–765.
[23] K.N. Berk, Forward and backward stepping in variable selection, J. Stat. Comput. Simul. 10 (3-4) (1980) 177–185.
[24] C. Gobert, E.W. Reutzel, J. Petrich, A.R. Nassar, S. Phoha, Application of supervised machine learning for defect detection during metallic powder bed fusion additive manufacturing using high resolution imaging, Addit. Manuf. 21 (2018) 517–528.
[25] A. Widodo, B.-S. Yang, Support vector machine in machine condition monitoring and fault diagnosis, Mech. Syst. Sig. Process. 21 (6) (2007) 2560–2574.
[26] S. Cho, S. Asfour, A. Onar, N. Kaundinya, Tool breakage detection using support vector machine learning in a milling process, Int. J. Mach. Tools Manuf. 45 (3) (2005) 241–249.
[27] L. Jack, A. Nandi, Support vector machines for detection and characterization of rolling element bearing faults, Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci. 215 (9) (2001) 1065–1074.
[28] K. Kammerer, B. Hoppenstedt, R. Pryss, S. Stökler, J. Allgaier, M. Reichert, Anomaly detections for manufacturing systems based on sensor data: insights into two challenging real-world production settings, Sensors 19 (24) (2019) 5370.
[29] F. Küppers, J. Albers, A. Haselhoff, Random forest on an embedded device for real-time machine state classification, in: 2019 27th European Signal Processing Conference (EUSIPCO), IEEE, 2019, pp. 1–5.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[30] Z. Wang, et al., Acoustic emission characterization of natural fiber reinforced plastic composite machining using a random forest machine learning model, J. Manuf. Sci. Eng. 142 (3) (2020).
[31] A. Agrawal, P.D. Deshpande, A. Cecen, G.P. Basavarsu, A.N. Choudhary, S.R. Kalidindi, Exploration of data science techniques to predict fatigue strength of steel from composition and processing parameters, Integr. Mater. Manuf. Innov. 3 (1) (2014) 90–108.
[32] O. Addin, S. Sapuan, M. Othman, A naïve-bayes classifier and f-folds feature extraction method for materials damage detection, Int. J. Mech. Mater. Eng. 2 (1) (2007) 55–62.
[33] A. Bacha, A.H. Sabry, J. Benhra, Fault diagnosis in the field of additive manufacturing (3D printing) using bayesian networks, Int. J. Online Eng. 15 (3) (2019).
[34] N. Zhang, L. Wu, J. Yang, Y. Guan, Naive bayes bearing fault diagnosis based on enhanced independence of data, Sensors 18 (2) (2018) 463.
[35] K. Vernekar, H. Kumar, K. Gangadharan, Engine gearbox fault diagnosis using empirical mode decomposition method and Naïve Bayes algorithm, Sādhanā 42 (7) (2017) 1143–1153.
[36] O. Addin, S. Sapuan, E. Mahdi, M. Othman, A Naïve-Bayes classifier for damage detection in engineering materials, Mater. Design 28 (8) (2007) 2379–2386.
[37] A. Keprate, R. Moslemian, Multiscale damage modelling of composite materials using Bayesian network, in: Proceedings of 1st International Conference on Structural Damage Modelling and Assessment, Springer, 2021, pp. 135–150.
[38] H. Wu, Z. Yu, Y. Wang, Experimental study of the process failure diagnosis in additive manufacturing based on acoustic emission, Measurement 136 (2019) 445–453.
[39] Y. Ech-Choudany, M. Assarar, D. Scida, F. Morain-Nicolier, B. Bellach, Unsupervised clustering for building a learning database of acoustic emission signals to identify damage mechanisms in unidirectional laminates, Appl. Acoust. 123 (2017) 123–132. https://doi.org/10.1016/j.apacoust.2017.03.008.
[40] E. Ramasso, V. Placet, M.L. Boubakar, Unsupervised consensus clustering of acoustic emission time-series for robust damage sequence estimation in composites, IEEE Trans. Instrum. Meas. 64 (12) (2015) 3297–3307.
[41] J.P. McCrory, et al., Damage classification in carbon fibre composites using acoustic emission: a comparison of three techniques, Compos. B: Eng. 68 (2015) 424–430.
[42] T. Gittler, S. Scholze, A. Rupenyan, K. Wegener, Machine tool component health identification with unsupervised learning, J. Manuf. Mater. Process. 4 (3) (2020) 86.
[43] W. Cui, Y. Zhang, X. Zhang, L. Li, F. Liou, Metal additive manufacturing parts inspection using convolutional neural network, Appl. Sci. 10 (2) (2020) 545.
[44] L. Scime, J. Beuth, A multi-scale convolutional neural network for autonomous anomaly detection and classification in a laser powder bed fusion additive manufacturing process, Addit. Manuf. 24 (2018) 273–286.
[45] C.-Y. Hsu, W.-C. Liu, Multiple time-series convolutional neural network for fault detection and diagnosis and empirical study in semiconductor manufacturing, J. Intell. Manuf. 32 (3) (2021) 823–836.
[46] J. Yang, et al., A hierarchical deep convolutional neural network and gated recurrent unit framework for structural damage detection, Inf. Sci. 540 (2020) 117–130.
[47] Z. Wang, Y.-J. Cha, Unsupervised deep learning approach using a deep auto-encoder with a one-class support vector machine to detect damage, Struct. Health Monit. 20 (1) (2021) 406–425.
[48] R. Zhao, D. Wang, R. Yan, K. Mao, F. Shen, J. Wang, Machine health monitoring using local feature-based gated recurrent unit networks, IEEE Trans. Ind. Electron. 65 (2) (2017) 1539–1548.
[49] Z. Zhan, H. Li, Machine learning based fatigue life prediction with effects of additive manufacturing process parameters for printed SS 316L, Int. J. Fatigue 142 (2021) 105941.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[50] J. Gokulachandran, K. Mohandas, Comparative study of two soft computing techniques for the prediction of remaining useful life of cutting tools, J. Intell. Manuf. 26 (2) (2015) 255–268. https://doi.org/10.1007/s10845-013-0778-2.
[51] X. Chen, Z. Shen, Z. He, C. Sun, Z. Liu, Remaining life prognostics of rolling bearing based on relative features and multivariable support vector machine, Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci. 227 (12) (2013) 2849–2860.
[52] Z. Zhan, H. Li, A novel approach based on the elastoplastic fatigue damage and machine learning models for life prediction of aerospace alloy parts fabricated by additive manufacturing, Int. J. Fatigue 145 (2021) 106089.
[53] D. Wu, C. Jennings, J. Terpenny, R.X. Gao, S. Kumara, A comparative study on machine learning algorithms for smart manufacturing: tool wear prediction using random forests, J. Manuf. Sci. Eng. 139 (7) (2017).
[54] C. Sun, M. Ma, Z. Zhao, S. Tian, R. Yan, X. Chen, Deep transfer learning based on sparse autoencoder for remaining useful life prediction of tool in manufacturing, IEEE Trans. Ind. Inf. 15 (4) (2018) 2416–2425.
[55] M. Zhang, et al., High cycle fatigue life prediction of laser additive manufactured stainless steel: a machine learning approach, Int. J. Fatigue 128 (2019) 105194.
[56] X. Li, et al., Adaptive network fuzzy inference system and support vector machine learning for tool wear estimation in high speed milling processes, in: IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society, IEEE, 2012, pp. 2821–2826.
[57] V. Jain, T. Raj, Tool life management of unmanned production system based on surface roughness by ANFIS, Int. J. Syst. Assur. Eng. Manag. 8 (2) (2017) 458–467.
[58] M. Dong, D. He, A segmental hidden semi-Markov model (HSMM)-based diagnostics and prognostics framework and methodology, Mech. Syst. Sig. Process. 21 (5) (2007) 2248–2266.
[59] C. Su, J. Shen, A novel multi-hidden semi-Markov model for degradation state identification and remaining useful life estimation, Qual. Reliab. Eng. Int. 29 (8) (2013) 1181–1192.
[60] Y. Peng, M. Dong, A prognosis method using age-dependent hidden semi-Markov model for equipment health prediction, Mech. Syst. Sig. Process. 25 (1) (2011) 237–252.
[61] A.H. Tai, W.-K. Ching, L.-Y. Chan, Detection of machine failure: hidden Markov model approach, Comput. Ind. Eng. 57 (2) (2009) 608–619.
[62] M.E. Orchard, G.J. Vachtsevanos, A particle-filtering approach for on-line fault diagnosis and failure prognosis, Trans. Inst. Meas. Control 31 (3-4) (2009) 221–246.
[63] R.K. Singleton, E.G. Strangas, S. Aviyente, Extended Kalman filtering for remaining-useful-life estimation of bearings, IEEE Trans. Ind. Electron. 62 (3) (2014) 1781–1790.
[64] P. Wang, R.X. Gao, Adaptive resampling-based particle filtering for tool life prediction, J. Manuf. Syst. 37 (2015) 528–534.
[65] S. Butler, J. Ringwood, Particle filters for remaining useful life estimation of abatement equipment used in semiconductor manufacturing, in: 2010 Conference on Control and Fault-Tolerant Systems (SysTol), IEEE, 2010, pp. 436–441.
[66] H. Rostami, J.-Y. Dantan, L. Homri, Review of data mining applications for quality assessment in manufacturing industry: support vector machines, Int. J. Metrol. Qual. Eng. 6 (4) (2015) 401.
[67] T. Evgeniou, M. Pontil, Support vector machines: theory and applications, Advanced Course on Artificial Intelligence, Springer, 1999, pp. 249–257.
[68] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[69] Z. Xue-Wu, D. Yan-Qiong, L. Yan-Yun, S. Ai-Ye, L. Rui-Yu, A vision inspection system for the surface defects of strongly reflected metal based on multi-class SVM, Expert Syst. Appl. 38 (5) (2011) 5930–5939.
[70] S. Saechai, W. Kongprawechnon, R. Sahamitmongkol, Test system for defect detection in construction materials with ultrasonic waves by support vector machine and neural network, in: The 6th International Conference on Soft Computing and Intelligent

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Systems, and The 13th International Symposium on Advanced Intelligence Systems, IEEE, 2012, pp. 1034–1039.
[71] G.F. Smits, E.M. Jordaan, Improved SVM regression using mixtures of kernels, in: Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290), 3, IEEE, 2002, pp. 2785–2790.
[72] A. Bordes, S. Ertekin, J. Weston, L. Botton, N. Cristianini, Fast kernel classifiers with online and active learning, J. Mach. Learn. Res. 6 (9) (2005).
[73] G. Loosli, S. Canu, L. Bottou, Training invariant support vector machines using selective sampling, in: Large-Scale Kernel Machines, 2, MIT Press, 2007. https://doi.org/10.7551/mitpress/7496.003.0015.
[74] D. Anguita, A. Ghio, N. Greco, L. Oneto, S. Ridella, Model selection for support vector machines: advantages and disadvantages of the machine learning theory, in: The 2010 International Joint Conference on Neural Networks (IJCNN), IEEE, 2010, pp. 1–8.
[75] M. Somvanshi, P. Chavan, S. Tambade, S. Shinde, A review of machine learning techniques using decision tree and support vector machine, in: 2016 International Conference on Computing Communication Control and Automation (ICCUBEA), IEEE, 2016, pp. 1–7.
[76] A. Priyam, G. Abhijeeta, A. Rathee, S. Srivastava, Comparative analysis of decision tree classification algorithms, Int. J. Eng. Technol. 3 (2) (2013) 334–337.
[77] O. Sagi, L. Rokach, Ensemble learning: a survey, Wiley Interdiscip. Rev.: Data Min. Knowl. Discov. 8 (4) (2018) e1249.
[78] J.S. Rao, W.J. Potts, Visualizing bagged decision trees, KDD, 1997, pp. 243–246.
[79] A. Berrouachedi, R. Jaziri, G. Bernard, Deep cascade of extra trees, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2019, pp. 117–129.
[80] I. Ben-Gal, Bayesian networks, Qual. Reliab. Eng. Int. 1 (2008).
[81] F.V. Jensen, Bayesian networks, Wiley Interdiscip. Rev. Comput. Stat. 1 (3) (2009) 307–315.
[82] O. Addin, S. Sapuan, M. Othman, B.A. Ali, Comparison of Nave bayes classifier with back propagation neural network classifier based on f-folds feature extraction algorithm for ball bearing fault diagnostic system, Int. J. Phys. Sci. 6 (13) (2011) 3181–3188.
[83] K.M. Al-Aidaroos, A.A. Bakar, Z. Othman, Naive Bayes variants in classification learning, in: 2010 International Conference on Information Retrieval & Knowledge Management (CAMP), IEEE, 2010, pp. 276–281.
[84] J. Pearl, Bayesian networks, 2011. https://escholarship.org/uc/item/53n4f34m.
[85] I. D. Dinov, Expectation maximization and mixture modeling tutorial, 2008. https://escholarship.org/uc/item/1rb70972.
[86] T. Kohonen, The self-organizing map, Proc. IEEE 78 (9) (1990) 1464–1480.
[87] V. Panchal, H. Kundra, J. Kaur, Comparative study of particle swarm optimization based unsupervised clustering techniques, Int. J. Netw. Secur. 9 (10) (2009) 132–140.
[88] N. Grira, M. Crucianu, N. Boujemaa, Unsupervised and semi-supervised clustering: a brief survey, Rev.Mach. Learn. Techn. Process. Multimed. Cont. 1 (2004) 9–16.
[89] N. Godin, S. Huguet, R. Gaertner, L. Salmon, Clustering of acoustic emission signals collected during tensile tests on unidirectional glass/polyester composite using supervised and unsupervised classifiers, NDT E Internat. 37 (4) (2004) 253–264.
[90] T. Wuest, D. Weimer, C. Irgens, K.-D. Thoben, Machine learning in manufacturing: advantages, challenges, and applications, Prod. Manuf. Res. 4 (1) (2016) 23–45.
[91] J. Wang, X. Su, An improved K-means clustering algorithm, in: 2011 IEEE 3rd International Conference on Communication Software and Networks, IEEE, 2011, pp. 44–46.
[92] W. Samek, T. Wiegand, and K.-R. Müller, Explainable artificial intelligence: understanding, visualizing and interpreting deep learning models, in: arXiv preprint arXiv:1708.08296 (2017).

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[93] M.-P. Hosseini, S. Lu, K. Kamaraj, A. Slowikowski, H.C. Venkatesh, Deep learning architectures, Deep Learning: Concepts and Architectures, Springer, 2020, pp. 1–24.
[94] A. Khamparia, K.M. Singh, A systematic review on deep learning architectures and applications, Expert Syst. 36 (3) (2019) e12400.
[95] S.-C. Wang, Artificial neural network, Interdisciplinary Computing in Java Programming, Springer, 2003, pp. 81–100.
[96] J.-S. Jang, ANFIS: adaptive-network-based fuzzy inference system, IEEE Trans. Syst. Man Cybern. 23 (3) (1993) 665–685.
[97] C. Chen, G. Vachtsevanos, M.E. Orchard, Machine remaining useful life prediction: an integrated adaptive neuro-fuzzy and high-order particle filtering approach, Mech. Syst. Sig. Process. 28 (2012) 597–607.
[98] E. Buber, D. Banu, Performance analysis and CPU vs GPU comparison for deep learning, in: 2018 6th International Conference on Control Engineering & Information Technology (CEIT), IEEE, 2018, pp. 1–6.
[99] D.J. Livingstone, D.T. Manallack, I.V. Tetko, Data modelling with neural networks: advantages and limitations, J. Comput. Aided Mol. Des. 11 (2) (1997) 135–142.
[100] L. Rabiner, B. Juang, An introduction to hidden Markov models, IEEE ASSP Mag. 3 (1) (1986) 4–16.
[101] D.R. Upper, Theory and Algorithms for Hidden Markov Models and Generalized Hidden Markov Models, University of California, Berkeley, 1997.
[102] D.B. Springer, L. Tarassenko, G.D. Clifford, Logistic regression-HSMM-based heart sound segmentation, IEEE Trans. Biomed. Eng. 63 (4) (2015) 822–832.
[103] A. Bonafonte Cávez, X. Ros Majó, J.B. Mariño, An efficient algorithm to find the best state sequence in HSMM, in: EUROSPEECH1993: 3rd European Conference on Speech Communication and Technology, Berlin, Germany, 1993.
[104] K. Fujinaga, M. Nakai, H. Shimodaira, S. Sagayama, Multiple-regression hidden Markov model, in: 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221), 1, IEEE, 2001, pp. 513–516.
[105] H. Bourlard, N. Morgan, Hybrid HMM/ANN systems for speech recognition: overview and new research directions, in: International School on Neural Networks, Initiated by IIASS and EMFCSC, Springer, 1997, pp. 389–417. https://link.springer.com/chapter/10.1007/BFb0054006.
[106] K. Aono, H. Hasni, O. Pochettino, N. Lajnef, S. Chakrabartty, Quasi-self-powered infrastructural internet of things: the Mackinac bridge case study, in: Proceedings of the 2018 on Great Lakes Symposium on VLSI, ACM, 2018, pp. 335–340.
[107] B.Q. Huang, C. Du, Y. Zhang, M.T. Kechadi, A hybrid HMM-SVM method for online handwriting symbol recognition, in: 2006 6th International Conference on Intelligent Systems Design and Applications, 1, IEEE Computer Society, 2006, pp. 887–891.
[108] A. Panuccio, M. Bicego, V. Murino, A Hidden Markov Model-based approach to sequential data clustering, Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), Springer, 2002, pp. 734–743.
[109] X. Tang, et al., A new method to assess fiber laser welding quality of stainless steel 304 based on machine vision and hidden Markov models, IEEE Access 8 (2020) 130633–130646.
[110] Q. Chen, Y. Huang, X. Weng, W. Liu, Curve-based crack detection using crack information gain, Struct. Control Health Monit. 28 (8) (2021). https://doi.org/10.1002/stc.2764.
[111] S.R. Eddy, What is a hidden Markov model? Nat. Biotechnol. 22 (10) (2004) 1315–1316.
[112] J. Bulla, I. Bulla, O. Nenadić, HSMM—An R package for analyzing hidden semi-Markov models, Comput. Stat. Data Anal. 54 (3) (2010) 611–619.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[113] J. Elfring, E. Torta, R. van de Molengraft, Particle filters: a hands-on tutorial, Sensors 21 (2) (2021) 438.
[114] G. Welch and G. Bishop, An Introduction to the Kalman Filter, 1995. University of North Carolina at Chapel Hill, Department of Computer Science, Chapel Hill, NC 27599-3175. http://dl.icdst.org/pdfs/files3/9bf7d17440970208375c6a5e7b81a121.pdf.
[115] D. Simon, Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches, John Wiley & Sons, 2006.
[116] P. Narksri, E. Takeuchi, Y. Ninomiya, K. Takeda, Deadlock-free planner for occluded intersections using estimated visibility of hidden vehicles, Electronics 10 (4) (2021) 411.
[117] M.E. Orchard, G.J. Vachtsevanos, A particle filtering-based framework for real-time fault diagnosis and failure prognosis in a turbine engine, 2007 Mediterranean Conference on Control & Automation, IEEE, 2007, pp. 1–6.
[118] T. Li, M. Bolic, P.M. Djuric, Resampling methods for particle filtering: classification, implementation, and strategies, IEEE Signal Process. Mag. 32 (3) (2015) 70–86.
[119] P.K. Rao, J.P. Liu, D. Roberson, Z.J. Kong, C. Williams, Online real-time quality monitoring in additive manufacturing processes using heterogeneous sensors, J. Manuf. Sci. Eng. 137 (6) (2015).
[120] J. Carvajal Soto, F. Tavakolizadeh, D. Gyulai, An online machine learning framework for early detection of product failures in an Industry 4.0 context, Int. J. Computer Integr. Manuf. 32 (4-5) (2019) 452–465.
[121] S. Agarwal, V.V. Saradhi, H. Karnick, Kernel-based online machine learning and support vector reduction, Neurocomputing 71 (7-9) (2008) 1230–1237.
[122] Z. Lin, S. Sinha, W. Zhang, Towards efficient and scalable acceleration of online decision tree learning on FPGA, in: 2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), IEEE, 2019, pp. 172–180.
[123] G. Ferrer, Real-time unsupervised clustering. 27th Modern Artificial Intelligence and Cognitive Science Conference (MAICS-2016), 2016. Dayton, OH. https://ceur-ws.org/Vol-1584/paper16.pdf.
[124] D. Sahoo, Q. Pham, J. Lu, S.C. Hoi, Online deep learning: learning deep neural networks on the fly, arXiv preprint arXiv:1711.03705 (2017).
[125] F. Gumus, C.O. Sakar, Z. Erdem, O. Kursun, Online Naive Bayes classification for network intrusion detection, in: 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM2014), IEEE, 2014, pp. 670–674.
[126] Z.-J. Zhou, C.-H. Hu, D.-L. Xu, M.-Y. Chen, D.-H. Zhou, A model for real-time failure prognosis based on hidden Markov model and belief rule base, Eur. J. Oper. Res. 207 (1) (2010) 269–283.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

CHAPTER NINE

Artificial intelligence for machining process monitoring

Hakkı Özgür Ünver[a], Ahmet Murat Özbayoğlu[b], Cem Söyleyici[a] and Berk Barış Çelik[a]
[a]Department of Mechanical Engineering, TOBB University of Economics and Technology, Ankara, Turkey
[b]Department of Artificial Intelligence Engineering, TOBB University of Economics and Technology, Ankara, Turkey

9.1 Introduction

Machining has played a major role in manufacturing industries for more than a century due to its ability to generate any intricate shape, on difficult-to-cut materials, with superior precision and accuracy [1]. Driven by mainly innovation in materials and fierce competition in international markets, machine tool and cutting tool manufacturers have continuously supplied new and advanced technologies to the manufacturing industry. This has been driven by demands to improve manufacturing quality while reducing costs for satisfying rigorous design requirements of high-value-added parts and products in key industries such as aviation/defense, automotive, and biomedical [2].

Improving the product quality, meeting extensive product customization, reducing costs, and maintaining high production rates simultaneously have been the biggest challenge in the manufacturing industry. To address this challenge, rapid progress has been made in the area of predictive modeling mainly in three categories: analytical, numerical, and artificial intelligence (AI)-based modeling. Physics-based model have been developed, and numerical methods have been used to solve the governing equations. For instance, mechanistic cutting force models based on mechanical shearing actions on the processed material have been developed [3,4]. A vibration model of regenerative chatter primarily relying on dynamic characteristics of tool-holder-spindle assembly, has also been developed [5–7].

The main limitations of analytical and numerical models come from their reliance on the physical conditions of the system. Due to manufacturing variability, the system conditions often change in a daily manufacturing environment, sometimes during the process itself. Apart from various types

Artificial Intelligence in Manufacturing: Concepts and Methods.
DOI: https://doi.org/10.1016/B978-0-323-99134-6.00010-4
Copyright © 2024 Elsevier Inc.
All rights are reserved, including those for text and data mining, AI training, and similar technologies.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY
Copyright Elsevier 2024

of machining that require physics-based modeling (e.g., turning, milling, broaching, and surface profiling), even for the same kind of machining operation, when a new part order is assigned to a machining center, a different tool, holder, or fixture may be required to accommodate the geometry and material of the new part. This modification alters the physical and dynamic characteristics of the system entirely. Therefore, a new experimental procedure or a new model may be required for the new part, hindering the applicability of these solutions without machine stoppage. Furthermore, during the machining process, the geometry of the machined parts continuously change, affecting the dynamics of the workpiece and fixture system, which decreases the accuracy of mathematical models.

In order to overcome the drawbacks mentioned above, intelligent manufacturing has been a key focal area of Industry 4.0. Although the application of AI to manufacturing has its roots in the 1980s, new advances in machine learning (ML) including deep learning (DL) during the last decade have fueled the rise of data-driven smart manufacturing in many industries. AI-based systems can improve the performance of a machining operation by monitoring and adapting to variations in the process. The main advantages of AI-based machining process monitoring (MPM) can be summarized as follows:

- Analytical (physics-based) models of physical systems often involve assumptions such as linearity, time invariance, and other assumptions about the properties of materials and agents involved in the process. However, these assumptions may not hold, due to uncertainties caused by suppliers and variations due to wear or alterations in systems.
- AI-based systems can work online, continuously receive real-time data and update their models, and adapt to variations in the manufacturing floor.
- Recent DL models can handle Big Data generated from Internet of Things (IoT)-based systems throughout a plant floor and convert it to key performance metrics that can help operators and shop managers to improve productivity and reduce costs across process chains [8].
- AI can enable optimization at the equipment/process level and the line/shop level, while utilizing Big Data collected. The gap between the process level and the shop floor management level can be bridged by AI, leading the overall efficiency of the shop or enterprise [9].

Before going into the details of AI-based MPM, it is essential to understand from a general perspective "what" needs to be monitored during machining. To achieve this end, four "ILITIES" of machining are listed

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

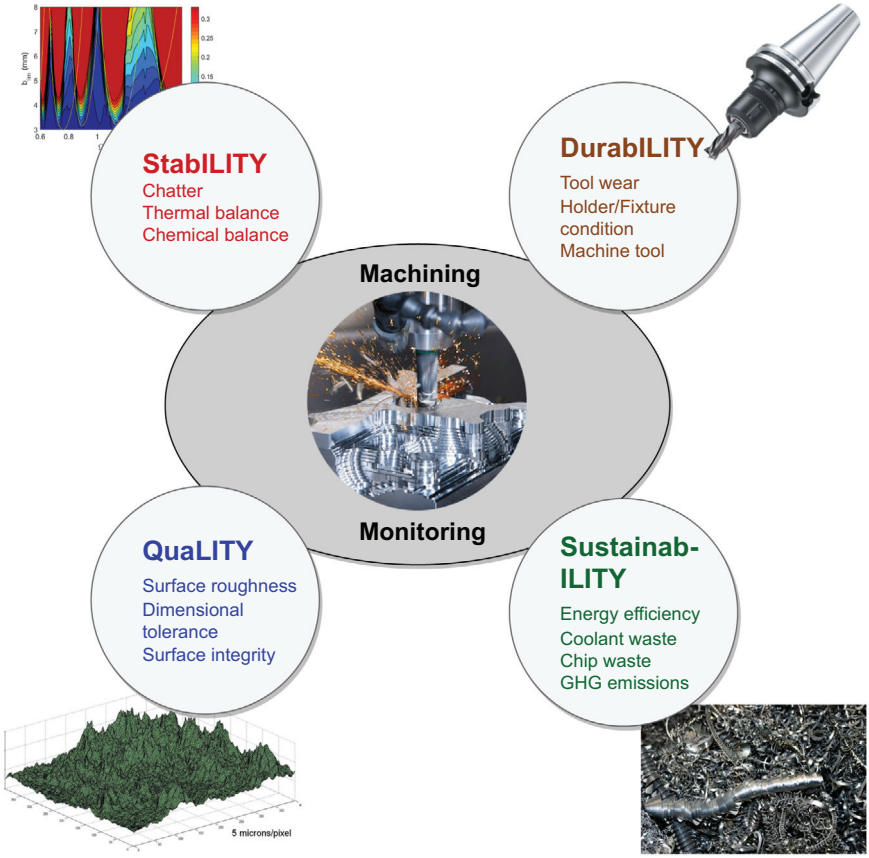


Figure 9.1 "ILITIES" of machining.

in Fig. 9.1; they represent various aspects that should be continuously monitored and optimized during an operation:

- **StabILITY**: A stable cutting operation free of chatter vibration and good thermal and chemical balance is essential for attaining the required functional performance of a high-value-added part.
- **DurabILITY**: The main effect of high cutting forces and friction is tool wear. Since it is a major cost, it is essential to extend tool life as much as possible and avoid premature wear or breakage. Secondary implications are on the durability of fixtures and the machine tool itself.
- **QuaLITY**: The critical quality measure of a part is dimensional tolerance and surface roughness. High forces and energy generation during cutting could also affect material integrity at the surface vis-a-vis mechanism of residual stress.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

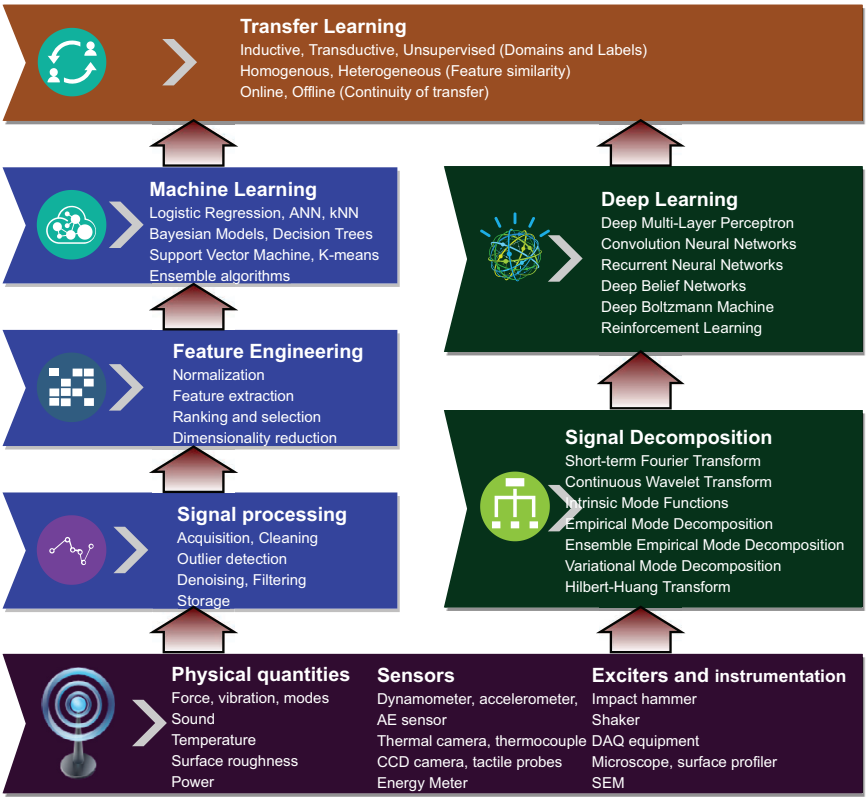


Figure 9.2 Artificial intelligence for machining process monitoring.

- **SustainabILITY**: Machining is high energy consuming. Thus, there is a great economic incentive to minimize the energy consumption. Furthermore, metalworking fluids are environmentally hazardous, and removed chips require recycling.

Implementing intelligent MPM systems requires a solid understanding of the technology hierarchy, consisting of AI advances, signal processing methods, and hardware used at the machine tool level. As presented in Fig. 9.2, there are two de-facto stacks today for AI implementation; machine learning and deep learning. Although both can use similar sensing and measurement equipment, the methods used for preprocessing measured data are often different, and their proper selection is critical for the performance of the learner model when used to make a decision. The upward arrows in the figure represent the data flow from the sensor level up to transfer learning (TL) level, indicating the connection between each layer.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

The objective of this chapter is to present an overview of the latest models and techniques used in each layer of the hierarchy given in Fig. 9.2.

9.2 Data acquisition systems

Monitoring and optimizing the "ILITIES" are vital to maintaining the continuity of the machining process and increasing tool life without system stoppage. A data acquisition (DAQ) system, which is essential for the monitoring process, consists of sensors, a signal conditioner, DAQ hardware, DAQ software, and a computer. The critical parts of the monitoring process are sensing devices which are transducers and sensors.

Transducers are capable of transforming energy from one form to another. A sensor is a device that detects a change in a dynamic, physical environment and produces an electrical output or a signal that corresponds to the change. Transducers perform the part of converting measurements to an electrical signal. For instance, strain gauges and pressure transducers produce electric signals correlated with the measured physical value [10]. Hence, sensors can be viewed as a type of transducers.

There are two main types of sensing methods for MPM, direct (offline) and indirect (online), as shown in Table 9.1 [11–17]. Indirect sensing methods are employed to observe an anomaly (e.g., chatter) through measured values (e.g., electric current or vibration) that are not directly related to an anomaly. However, direct sensing methods cannot collect data during cutting processes but only between cutting processes [12]. Although the direct sensing transducers (e.g., optic sensors, voltmeters and displacement transducers) have a high measurement accuracy, their disadvantages are their high cost and sensitivity to other noise sources such as pumps, coolants, and conveyors [13]. Moreover, the setup of the direct methods interrupts the machining process and decreases production time. On the other hand, indirect methods' measurement accuracy is relatively low compared to offline methods, and internal computation may take more time due to noisy data. Nevertheless, most indirect sensors (e.g., accelerometers, AE sensors and thermocouples) are low cost and can make measurements in real time with sufficient quality during the machining process.

Sensor selection, quantity, and application depend on the machining type such as turning, milling, boring, grinding, and broaching and also on the workpiece materials used in the machining operation [16]. For instance, a tool wear monitoring study on a turning machine employed three different sensors, a three-axis dynamometer, an accelerometer, and a surface roughness

This book belongs to COMEX COMEX (comexcom83@gmail.com Copyright Elsevier 2024

Table 9.1 Sensor types and measurement purposes.

Sensing method	Signal acquisition	Transducer	Measurement purpose	Ease of installation	Cost
Direct (offline)	Visual/optical	CCD camera, optic sensor	Changes in machine tool integrity and position	*****	****
	Displacement	Optoelectronic sensor, micrometer, pneumatic gauges, electromagnetic sensor	The gap between the workpiece and machine tool	***	**
	Electrical resistance	Voltmeter	Changes in resistance depending on contact area between workpiece and machine tool	**	*
Indirect (online)	Cutting force	Dynamometer, strain gauge, piezoelectric	Cutting forces variance on turret and milling spindle	*	****
	Vibration	Accelerometer	Recording vibration on machine tools and chuck	****	**
	Temperature	Thermocouple, pyrometer, infrared	Cutting tool and workpiece temperature changes	****	**
	Current/power	Power meter, ampere meter, dynamometer	Consumed power or current drawn by milling spindle, main spindle, and feed motors	*****	*
	Sound	Acoustic emission (AE), microphone	Machining vibration sound, AE signals from tool, chip breakage, collisions, plastic deformations, etc.	*****	**
	Surface roughness	CCD camera, AE, fiber optic sensor, laser profilometer	Surface profile changes of the workpiece	***	***

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY

Copyright Elsevier 2024

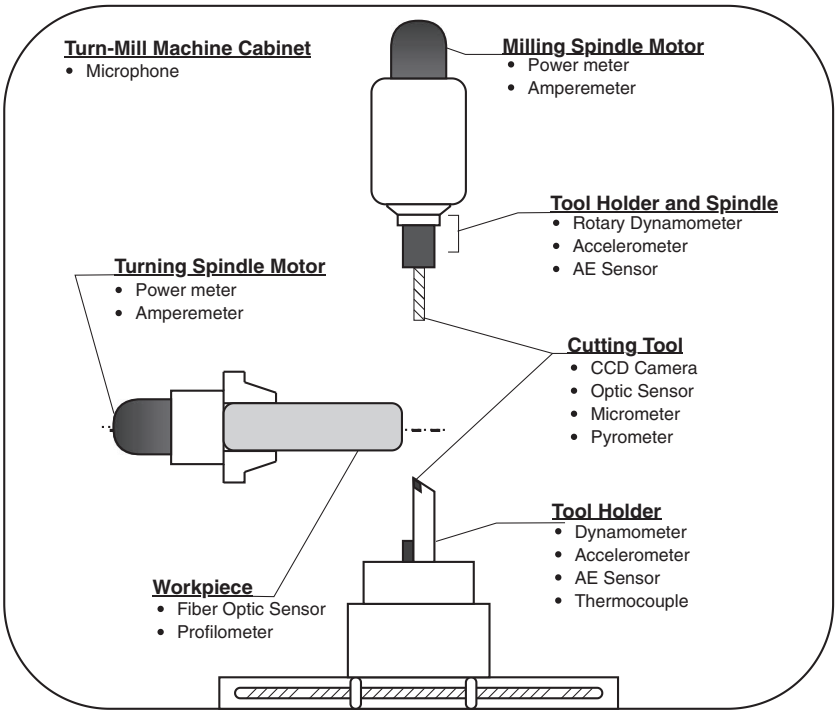


Figure 9.3 Sensor placements on a turn-mill machine tool.

tester, and fused the output signals to decrease error in the measurements [18]. As an example of a comprehensive machine tool that can employ multiple machining processes, a turn-mill machine's main components and possible sensor placement locations are illustrated in Fig. 9.3.

The electric signal that is produced by a sensor is not always readable by DAQ hardware. A signal conditioner placed on the DAQ hardware or external modules converts resistance to potential. The potential is divided, amplified, and shifted by subcircuits to enhance the quality and performance of the equipment [19]. The conditioned analog signal is fed to the DAQ hardware and the analog to digital converter converts the signal to bits and bytes. The digital signals are processed further as required by the application on the DAQ software which runs on a computer.

Various brands and types of DAQ hardware and software are available to meet a system's desired measurement specifications. For instance, the National Instruments™ DAQ equipment used for chatter detection on a turn-mill machine tool are shown in Fig. 9.4.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

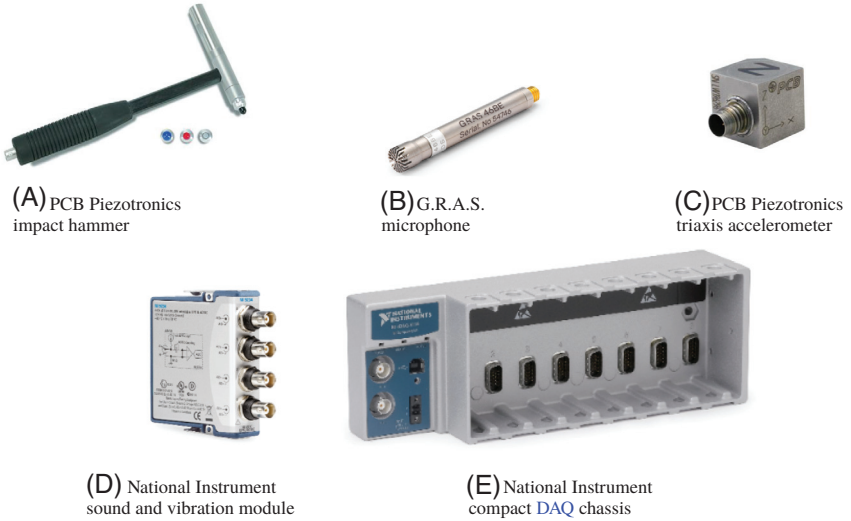


Figure 9.4 Data acquisition (DAQ) equipment.

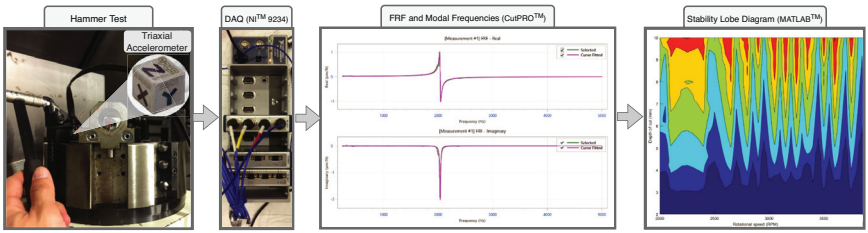


Figure 9.5 Tool-holder-spindle modal analysis to characterize the system and stability lobe diagrams (SLD) generation.

The overall DAQ process will be described using a part of a chatter detection experiment. Before collecting machining data, system dynamic behavior may often need to be identified. This is performed on a tool-holder-spindle system using experimental modal analysis techniques as shown in Fig. 9.5. After modal anlaysis, sensor placement and data collection during machining should be performed (Fig. 9.6).

The DAQ and collection activities can also be evaluated from a data processing requirements perspective. For example, some specific sensor data may need to be captured and evaluated in real time, necessitating the usage of IoT and intelligent sensors with adequate processing power for preparing high-quality and digitized data. The manufacturing industry, primarily large-scale manufacturing, employed IoT sensors due to their reduced costs recently,

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

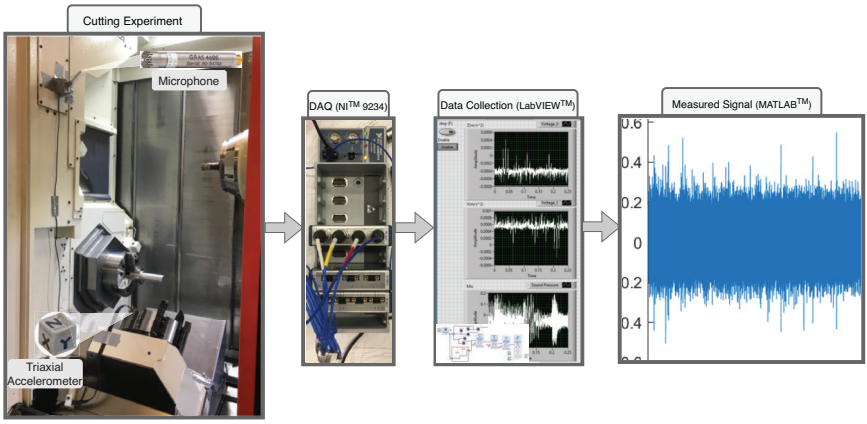


Figure 9.6 Data collection process.

wireless communication capability, minimizing human effort, machine-to-machine communication, and seamless working with automation systems [20].

Moreover, leveraging the expansion of IoT systems, edge computing technology allows the processing of collected data closer to the machine without sending the data to the cloud, enabling real-time analysis and decision-making at the moment of the need [21,22]. On the other hand, most of the sensors used in MPM may be utilized for local DAQ and processing, targeted to experimental implementations for on-the-spot problem solving or optimization. This careful balance between the processing and accessibility capabilities of the sensors requires appropriate local and distributed network architecture design, that includes corresponding edge, fog, and cloud-based systems working together within MPM systems.

The tool-holder-spindle modal analysis begins with the impact hammer test on the insert to understand the dynamic characteristics, known as "tap testing." With each tap of an impact hammer on the cutting tool, vibration signals are captured by the accelerometer attached to the cutting tool or holder. Modal analysis software then computes the modal frequencies, damping ratio, modal stiffness, and modal mass. Finally, a stability lobe diagram (SLD) is generated either by using an analytical or numerical solution method for identifying chatter regions, using MATLAB or directly by a commercial software product such as CutPRO™.

Usually, data collection must be preplanned with different cutting parameters (e.g., depth of cut, feed rate, and spindle speed). In addition, all sensors, such as accelerometers, AE sensors, and dynamometer, should be placed in

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

their appropriate locations before data collection. After data collection, data are stored and used by signal processing algorithms and for training and validation of ML algorithms.

The sensors can produce a large volume of data in different (e.g., structured, semi-structured, and unstructured) forms with high velocity (e.g., streams and real time) [23]. However, during and afterDAQ, there might be some challenges related to the data, such as data volume, velocity, variety, quality, variability, veracity, visualization, and value [24].

The collected data may be corruped with noise, have missing values, and/or have outliers. These problems degrade the performance of ML/DL models. In order to enhance signal quality, DAQ devices have specialized systems and software. Furthermore, analog or digital filters can reduce the noise in the signal and improve data quality.

Data that contain missing values cannot be accepted as an input by ML/DL models. To overcome this issue, missing values can be discarded or replaced with estimated values based on statistical indicators. Outliers decrease the test accuracy of the ML/DL models. They can be eliminated by discarding them from the data set and then treating them as missing data points.

9.3 Feature engineering and machine learning

Machine learning algorithms enable tool wear estimation, chatter detection, surface roughness estimation, and energy consumption estimation using various types of data collected from different sources. However, as data obtained from sensors are usually raw, before they can be used in ML algorithms, they must undergo extensive processing and conditioning. Common stages of data-preprocessing and ML steps are illustrated in Fig. 9.7.

A feature represents raw data that is an individual, quantifiable trait often represented by columns in n-dimensional datasets. Observations are commonly represented as rows. The whole array of features in all observations creates an n-dimensional feature matrix, also known as the vector of features, with each row representing a feature array.

In the ML pipeline, feature engineering is a critical phase [25]. An algorithm's artificial features are created through a process known as feature engineering. Feature engineering creates new features from existing data to improve predictive learning performance. The algorithm uses these synthetic features to enhance its functionality or to rephrase it, to provide better outcomes. It is a time-consuming and labor-intensive stage essential to the

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

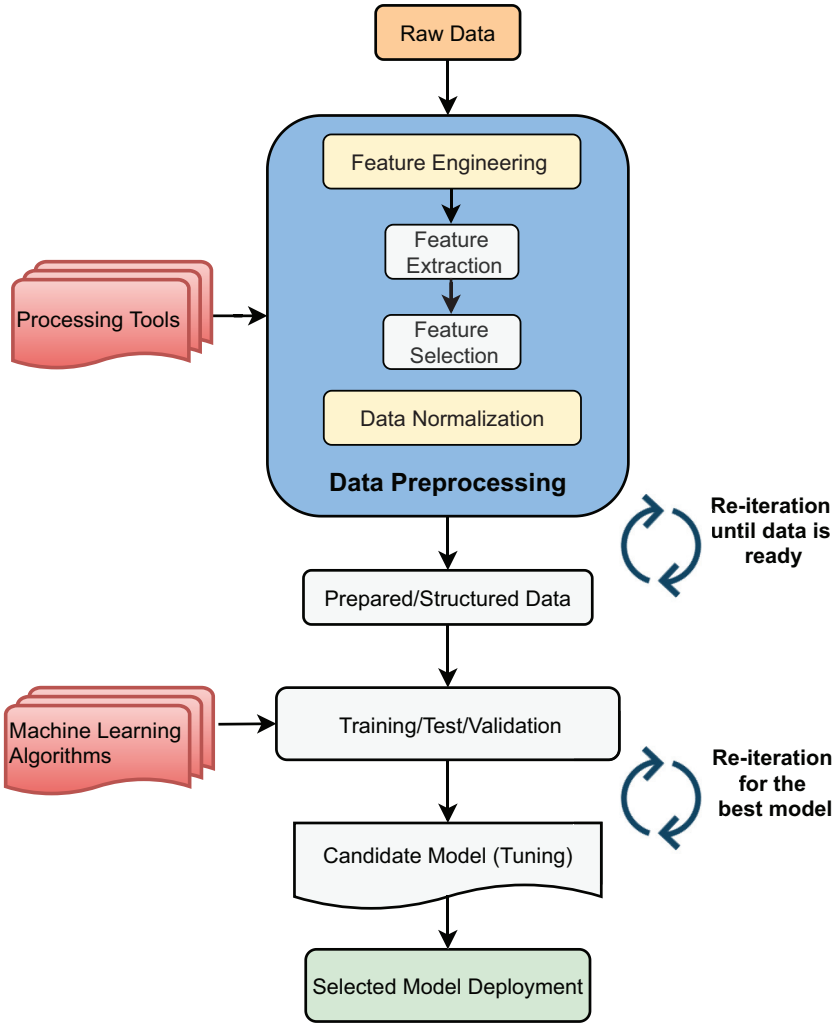


Figure 9.7 The common steps from raw data to ML model deployment.

data science workflow [26]. Since data scientists work mostly with data, the accuracy of the models becomes essential. Although newer approaches like DL and meta-heuristics are assisting in automated ML, every problem is narrow in scope, and better features (matched to the problem) usually determine the system's effectiveness. Those who deal with data science spend a large part of their time with feature engineering to prepare data. Unlike the data preparation part, the model construction covers a smaller amount of time, contrary to what is known.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

The data collected and processed in manufacturing are generally force, acceleration, sound, and image. When the data are considered, it is necessary to digitize all data used in ML models. The input type, the data can be continuous or discrete. According to its content, it can be nominal (orderless, unrelated to each other) and ordinal (there is a hierarchical relationship between them). The data types are nonsequential, sequential, and time series according to their relationship. Feature engineering can be further discussed under two subheadings, feature extraction, and feature selection.

9.3.1 Feature extraction

Feature extraction is a procedure that reduces a large collection of raw data into smaller groups for analysis. The massive number of variables within those large data sets requires generally a lot of computational resources. Extraction of features refers to strategies for selecting and/or combining parameters into features to reduce the quantity of data that must be analyzed while properly and thoroughly characterizing the original data set. Vibration signals from machinery are frequently studied to extract fault patterns, which are then identified using classifiers such as artificial neural networks (ANNs) and support vector machines (SVMs) [27]. The visualization of the feature extraction process is shown in Fig. 9.8. *XY* and *ZY* directions indicate extracted features. Many leading feature extraction methods such as the principal component analysis (PCA), t-distributed stochastic neighbor embedding (t-SNE), and autoencoder have been applied to tool wear [28–32], chatter detection [33–35], energy consumption [36,37], and sustainability [38,39].

The principal component analysis is a method for analyzing a data frame in which observations are characterized by interconnected quantitative dependent variables. Its purpose is to extract the key data and interpret it as a collection of new orthogonal variables known as principal components [40]. A novel tool monitoring approach based on the multiscale principal component analysis (MSPCA) was provided by Wang et al. [28]. Based on PCA, statistical indicators and accompanying control limits were created to monitor wear rate. According to the study, the accuracy and durability of the tool wear monitoring approach can be considerably enhanced by combining PCA with wavelet transform. Fig. 9.9 summarizes the wear monitoring approach. In this approach, test data were preprocessed. Next, feature extraction and discrete wavelet transform operations were performed, followed by the application of PCA at each scale. After the signals were reconstructed, and PCA was applied again, the system decided the tool condition.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

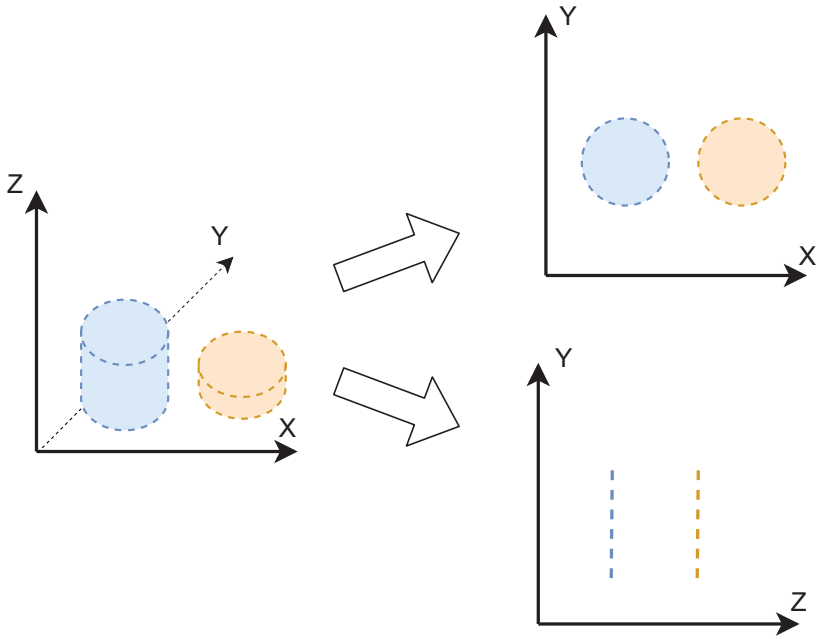


Figure 9.8 Conceptual visualization of feature extraction.

t-Distributed stochastic neighbor embedding effectively captures most of the spatial correlation of high-dimensional data while exposing the global structure, such as clustering existence at multiple scales. When constructing a single map that displays structures of several sizes, t-SNE outperforms previous approaches. This is especially crucial for high-dimensional data, which is spread across numerous low-dimensional manifolds yet to be connected, such as photographs of items from various classes seen from varied perspectives [41]. Due to multifeature datasets needing large dimensions of space and beging challenging to depict directly, Wan et al. used the t-SNE approach to lower the dimension of the data set in milling [34]; their t-SNE's outcomes are displayed in Fig. 9.10. Although t-SNE is computationally complex, it can handle nonlinear data. For example, since PCA has a linear algorithm, it may not be reliable when data are nonlinear. Another important advantage of t-SNE is that it maintains the global and local structure.

Training a multilayer neural network with a modest core layer to reconstruct high-dimensional inputs can be used to transform high-dimensional data into low-dimensional codes [42]. Gradient descent may be used to fine-tune the weights in the autoencoder networks, but only when the

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

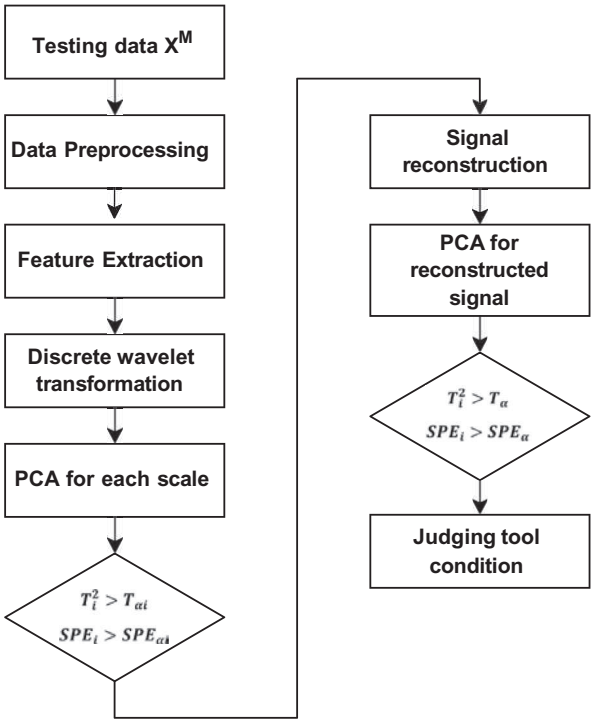


Figure 9.9 Multiscale principal component analysis (MSPCA) method for tool wear monitoring [28].

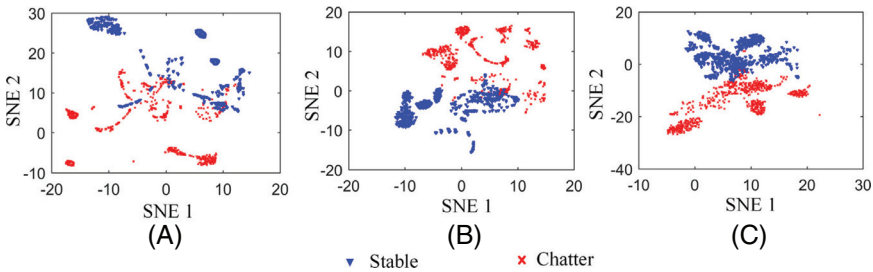


Figure 9.10 A chatter detection dataset representation using t-distributed stochastic neighbor embedding (t-SNE) [34].

beginning weights are near a satisfactory solution. The autoencoder may be thought of as a tool for changing representation. It may be generated with a compressed input representation by decreasing the number of hidden layer nodes, *m*, to be smaller than the number of original input nodes, *n*. This accomplishes the desired dimensionality reduction effect [43]. Fig. 9.11 presents a chatter detection dataset with autoencoder applied (each axis

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

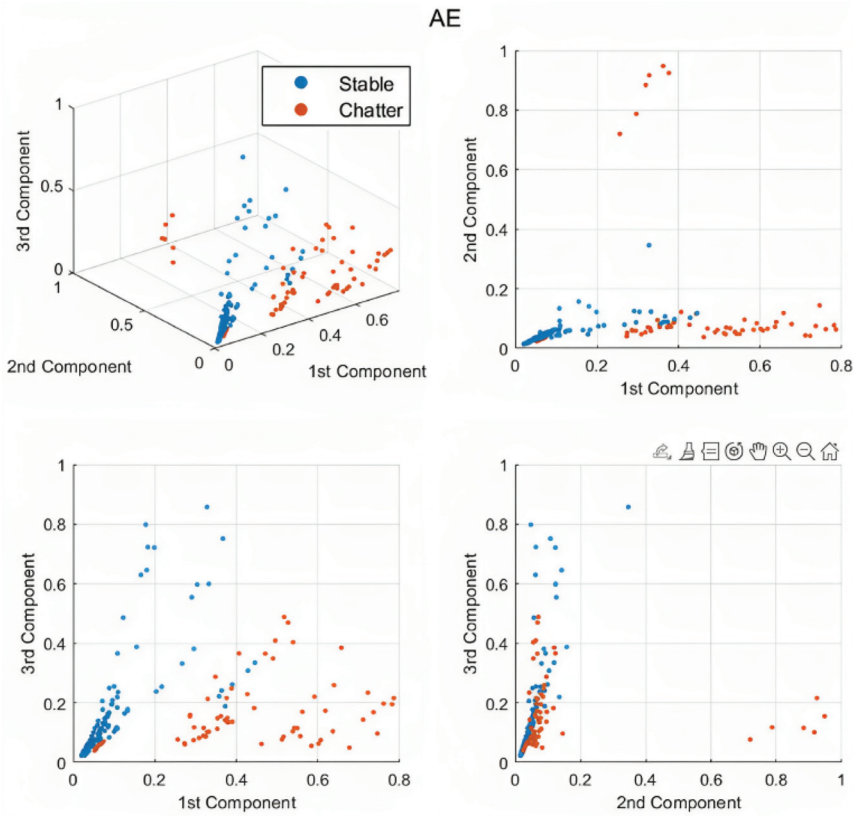


Figure 9.11 A chatter detection dataset representation using an autoencoder.

represents a component). Here, three principal components were obtained for chatter detection by compressing multidimensional data in hidden space with the help of an autoencoder.

9.3.2 Feature selection

Feature selection has shown to be an efficient and productive approach for preparing data for various data mining and ML models. Building simple and more precise models, enhancing data-mining speed, and producing clean, comprehensible data are all goals of feature selection [44]. In addition, because of its clarity, scalability, and empirical effectiveness, many variable selection methods employ variable ranking as a primary or auxiliary feature selection method [45]. Feature selection methods can be unsupervised or supervised. Correlations can be examined with supervised ones. An example is the Pearson correlation matrix, which is frequently used in engineering.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

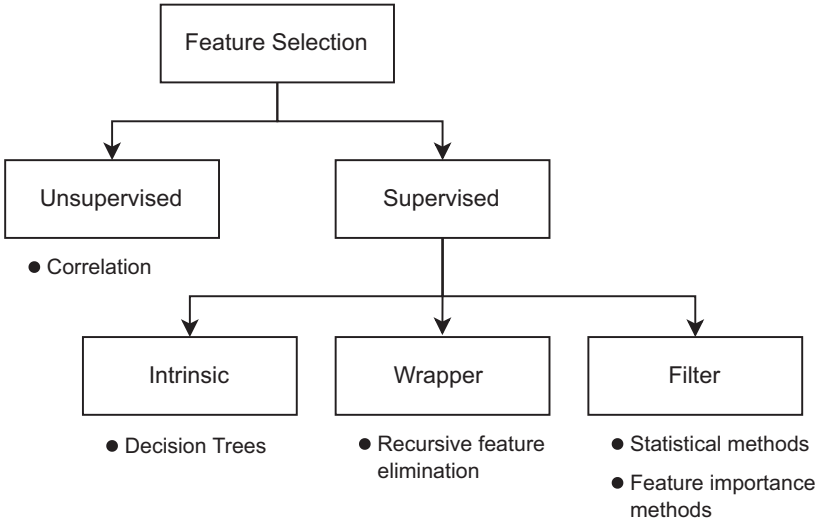


Figure 9.12 Feature selection methods.

Supervised feature selection methods include intrinsic, wrapper, and filter methods (Fig. 9.12).

Pearson's correlation matrix is an example method for selecting milling features such as surface roughness prediction [46] and chatter detection [47]. Pearson's correlation coefficient is a statistical measure of the statistical link between different variables. It provides information on the amount and direction of the correlation. This coefficient indicates whether or not characteristics and labels are properly associated. Eq. (9.1) gives the Pearson correlation formula (cov, X, and Y are the covariance and standard deviation of X and Y, respectively.). An example correlation matrix is presented in Fig. 9.13.

$$\rho_{X,Y} = \frac{\mathrm{cov}(X, Y)}{\sigma_X \sigma_Y} \tag{9.1}$$

9.3.3 Machine learning models

As in many data science fields, statistical features are frequently used in tool condition monitoring [49–52] and chatter detection [53–55]. Various measurements can be made more meaningful and valuable with these features. Statistical features frequently used in manufacturing are root mean square (RMS), peak value, crest factor, impulse factor, variance, dispersion index, skewness, kurtosis, clearance factor, and shape factor. Table 9.2 includes

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

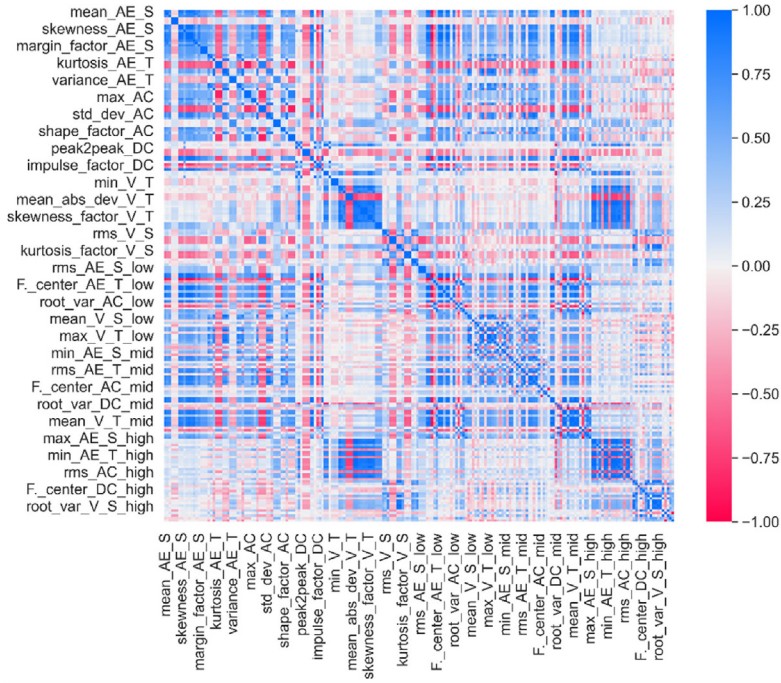


Figure 9.13 An example Pearson's feature correlation matrix [48].

equations describing these statistical features (x represents a vector, σ stands for the standard deviation of x, and $\bar{x}$ denotes the mean of x).

Machine learning algorithms can immediately realize structured data, which are well organized and often fall under the category of quantitative data. Querying and processing of ML data are made easier by structured data's precise and orderly architecture.

Machine learning aims to answer the question of how to create machines that learn [32]. It is at the crossroads of computer science and statistics, at the heart of AI and data science, and is undisputedly today's rapidly advancing technology topic. The development of novel learning algorithms and theory, as well as the progressive boom in the availability of online data and low-cost processing, has fueled recent advances in ML. Machine learning models are classified under three general categories. These are supervised learning, unsupervised learning, and reinforcement learning. The main categories and specialized ML algorithms used mostly in MPM are presented in Fig. 9.14.

In supervised learning, the output variable (or dependent variable) is predicted from a collection of predictors (independent variables). A function that maps inputs to expected output using this collection of variables is

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Table 9.2 Statistical features.

Statistical features	Formula
Root mean square (*RMS*)	$\sqrt{\frac{1}{N}\sum_{n=1}^{N} x_n^2}$
Peak value (P_v)	$0.5[\max(x_n) - \min(x_n)]$
Crest factor	$\sqrt{\frac{P_v}{RMS}}$
Impulse factor	$\frac{P_v}{\frac{1}{N}\sum_{n=1}^{N} \lvert x_n \rvert}$
Variance	$\frac{1}{N}\sum_{n=1}^{N} (x_n - \bar{x})^2$
Index of dispersion	$\frac{\sigma}{\frac{1}{N}\sum_{n=1}^{N} x_n}$
Skewness	$\frac{\frac{1}{N}\sum_{n=1}^{N} (x_n - \bar{x})^3}{\sigma^{3/2}}$
Kurtosis	$\frac{\frac{1}{N}\sum_{n=1}^{N} (x_n - \bar{x})^4}{\sigma^2}$
Clearance factor	$\frac{P_v}{\left(\frac{1}{N}\sum_{n=1}^{N} \lvert x_n \rvert\right)^2}$
Shape Factor	$\frac{RMS}{\frac{1}{N}\sum_{n=1}^{N} \lvert x_n \rvert}$

modeled and created. The model is trained until it reaches the appropriate degree of accuracy on the training dataset. Supervised learning can be examined under two subtitles. These are classification and regression. Many classification methods such as ANNs, multilayer perceptrons (MLPs), decision trees, random forests, and SVMs have been applied to chatter detection [47,53,56–58], tool wear [59–63], surface roughness estimation [64–68], and energy estimation [69,70].

One of the most successful methods used in surface roughness estimation is ANNs, inspired by biology [71]. A neural network representation is:

$$\mathrm{Z} = \mathrm{Bias} + \mathrm{W}_1\mathrm{X}_1 + \mathrm{W}_2\mathrm{X}_2 + \cdots + \mathrm{W}_\mathrm{n}\mathrm{X}_\mathrm{n} \tag{9.2}$$

where, *W*'s are weights, *X*'s are inputs, and *Z* is the output. Estimating surface roughness is critical for increasing productivity. A study by Khorasani and Yazdi aims to develop a universal dynamic monitoring system for

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

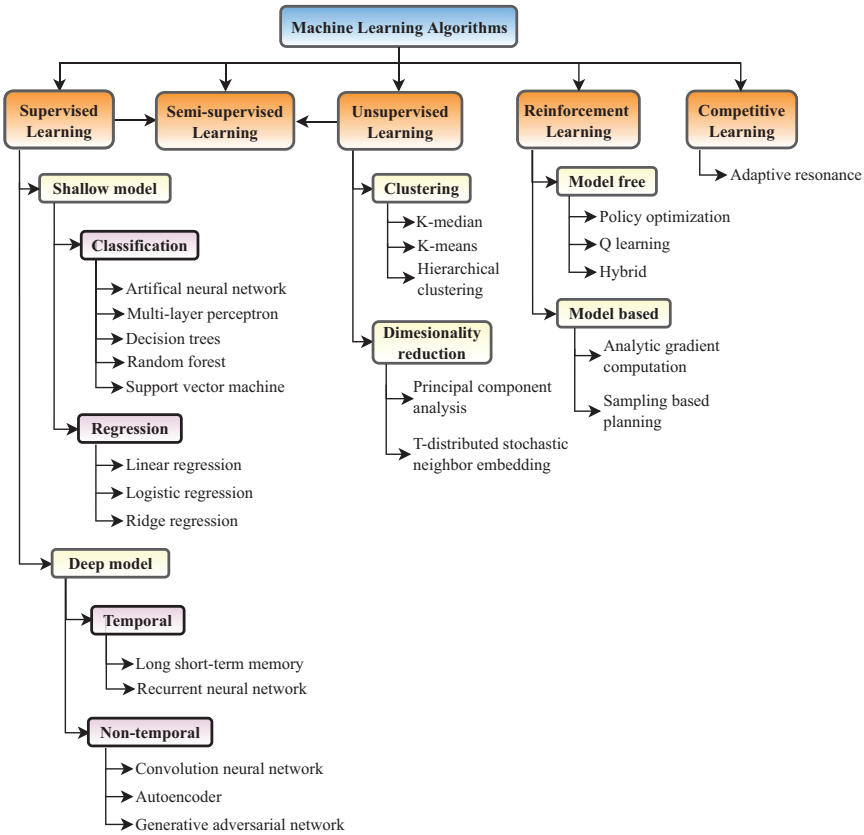


Figure 9.14 ML algorithms used in machining process monitoring.

surface roughness in milling [72]. A complete factorial design simulates dynamic surface roughness monitoring during milling for two substances in their investigation. Dynamic surface roughness monitoring using ANN is demonstrated in Fig. 9.15.

In process monitoring, classifiers are also often used [73]. Multilayer perceptrons are a popular ML approach that uses neural networks to create a classifier. By adjusting the combination of the inputs' weights, a perceptron may produce a single output from several inputs. Following that, the output of a perceptron is passed via activation functions like tanh or rectified linear unit (reLU). If enough data are utilized for training the model, this gives it the capacity to estimate any nonlinear function [47]. A typical MLP architecture is presented in Fig. 9.16.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

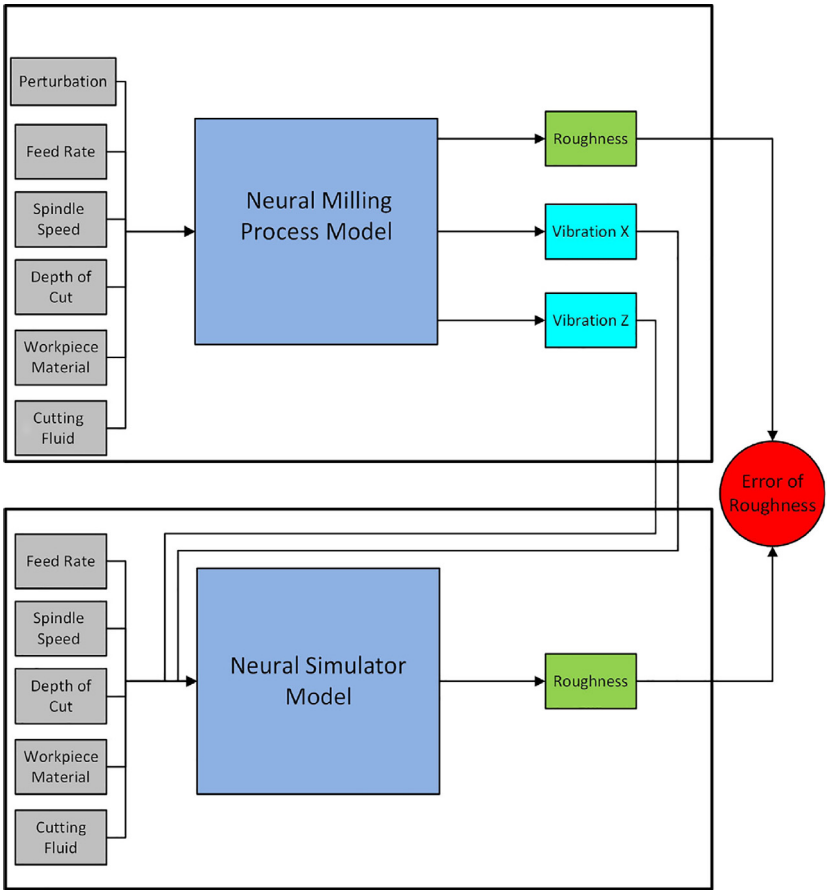


Figure 9.15 Surface roughness estimation using an ANN [72].

When a high material removal rate is required in industrial applications, milling is an essential machining process [75]. However, the chatter phenomenon often limits machining efficiency in milling. Multilayer perceptrons using statistical features as inputs are often used in chatter detection. Fig. 9.17 provides a sample model architecture. This work uses a self-organizer map-based MLP classifier to classify multifeatures. The model is verified with end milling tests demonstrating successful online detection at the onset of chatter.

One of the most studied methods for tool wear estimation is SVM methods. Support vector machines are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis. A support vector machine simulates scenarios by building a feature space,

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

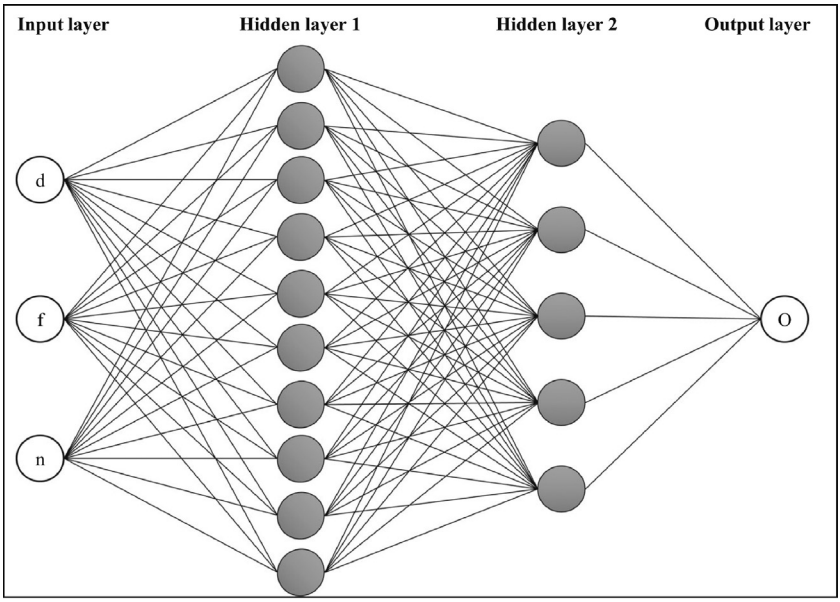


Figure 9.16 An MLP architecture [74].

that is, a finite-dimensional vector space whereby each dimension reflects a "feature" of a certain item. Every "feature" in the domain of content categorization is the frequency or significance of a certain term. The SVM's purpose is to build a classifier that categorizes novel objects that have not been seen before. Fig. 9.18 presents a dataset classified with SVM and SVM components (support vectors, hyperplane, maximum margin, and margin). If a linear classifier is defined as follows:

$$h(x) = \text{sign}\left(w^T x + b\right) \tag{9.3}$$

Examples of different hyperplane equations separating data are:

$$\begin{aligned} \left(w^T x_i\right) + b > 0 \qquad & \text{if } y_i = 1 \\ \left(w^T x_i\right) + b < 0 \qquad & \text{if } y_i = -1 \end{aligned} \tag{9.4}$$

The best hyperplane is the hyperplane with the maximum margin. Results from various types of SVMs are shown in Fig. 9.19.

Monitoring tool wear status during milling efficiently and correctly leads to optimizing process parameters to ensure milling stability and quality [79]. Feature extraction, feature selection, and prediction are the three components that make up the approach presented in Fig. 9.20. Time-domain analysis, frequency domain analysis, and wavelet decomposition

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

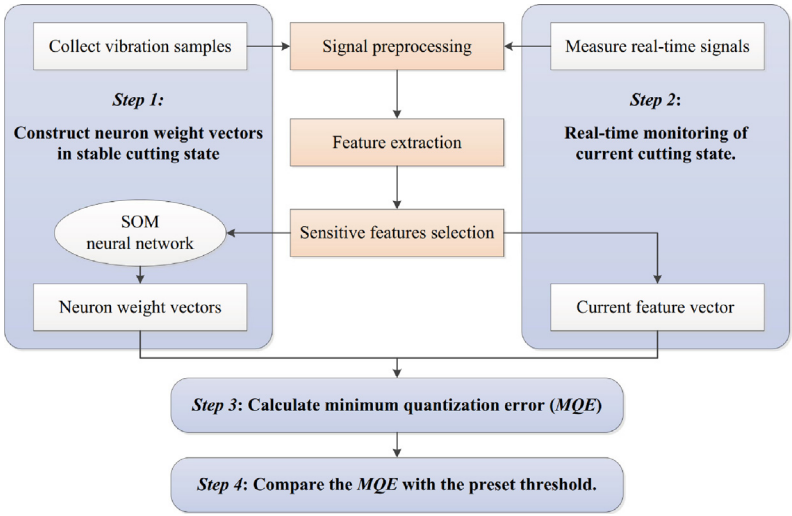


Figure 9.17 Chatter detection using statistical features and a multilayer perceptron [76].

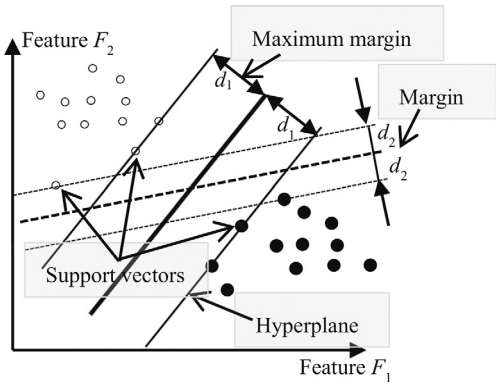


Figure 9.18 A two-class case support vector machine [77].

extract features. In order to reduce the complexity of the model, feature selection utilizes a genetic algorithm. To forecast tool wear state, the authors presented an SVM model improved using gray wolf optimization (GWO) for comprehensiveness. The outcomes of a comparison between the suggested approach and an SVM model improved using popular optimization methods show that the genetic algorithm (GA)–GWO–SVM model performs better than others (particle swarm optimization [PSO]–SVM, GA–PSO–SVM, GWO–SVM) in terms of accuracy and time.

Furthermore, regression methods such as linear regression and logistic regression have been applied to chatter detection [80], tool wear [81], surface

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

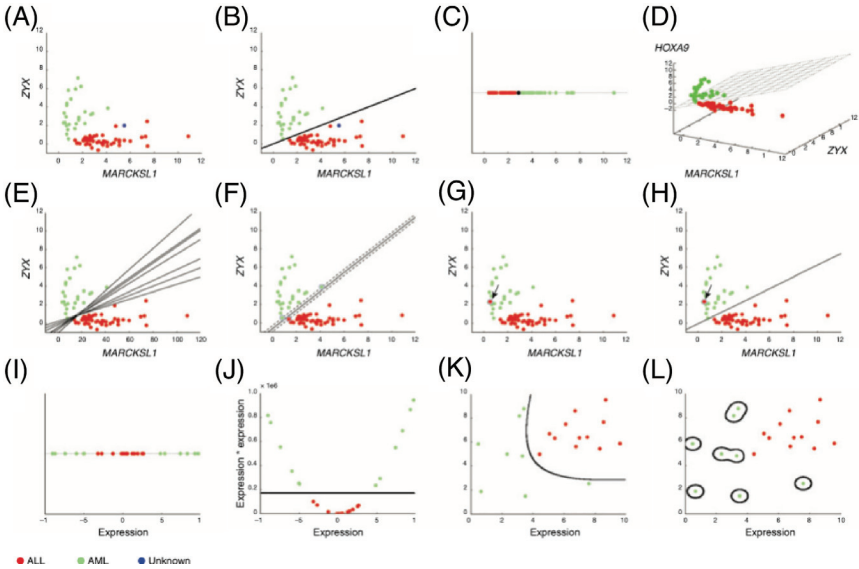


Figure 9.19 Support vector machines at work [78].

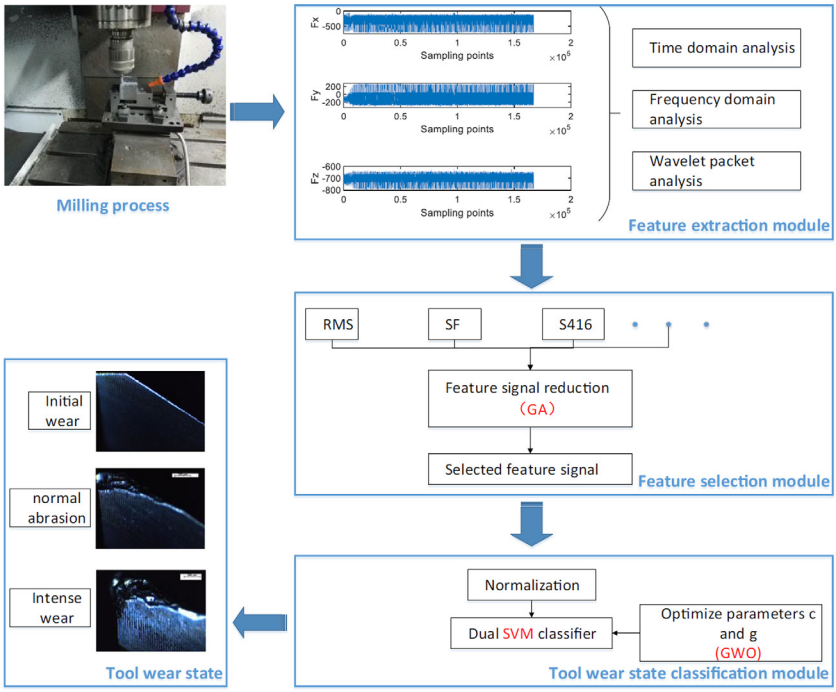


Figure 9.20 Tool wear state monitoring system using a support vector machine [79].

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

roughness estimation [82], and energy estimation [83]. Logistic regression is frequently used to identify an effective linear decision boundary as a distinct difference between two labels owing to its low computing cost and ease of implementation [80].

For unsupervised learning category, also known as clustering, there are no objective or result variables to predict or evaluate. It is commonly used to segment clients into separate groups for specialized action, and it is utilized to cluster populations into groups. Some unsupervised learning algorithm examples are k-nearest neighbors (kNN), k-means, and hierarchical clustering. Some of these unsupervised learning algorithms have been applied to tool wear [84,85], chatter detection [86,87], surface roughness estimation [88,89], and energy estimation [90].

Semi-supervised learning is a learning paradigm that examines how computers and natural systems like people learn in the presence of both labeled and unlabeled input [91]. Very few semi-supervised algorithms have been applied to MPM for chatter detection [92], tool wear [93], and surface roughness estimation [94].

In reinforcement learning, machine intelligence is taught to make certain judgments. The ML model is run in an environment where it should constantly teach itself via trial and error. The algorithm learns from previous experiences and attempts to acquire the most relevant information to make better decisions. Markov's decision process is the best-known example of reinforcement learning. Markov models are based on two assumptions, the limited horizon assumption Eq. (9.5) and the stationary process assumption Eq. (9.6).

$$P(z_t, z_{t-1}, z_{t-2}, \ldots z_1) = P(z_t/z_{t-1}) \tag{9.5}$$

$$P(z_t/z_{t-1}) = P(z_2/z_1) \rightarrow t \in 2 \cdots \mathrm{T} \tag{9.6}$$

The hidden Markov models (HMMs) are a strong framework for the periodic analysis of features taken from signals. A generalized HMM strategy proposed by Xie et al. [95] for chatter monitoring outperformed conventional HMM in terms of accuracy. Fig. 9.21 demonstrates a sample architecture. Hidden Markov models are well suited for modeling dynamic time-series data. It also has good pattern categorization capabilities; hence, it can process lengthy, random sequences, proving it a good choice for the monitoring of chatter state.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

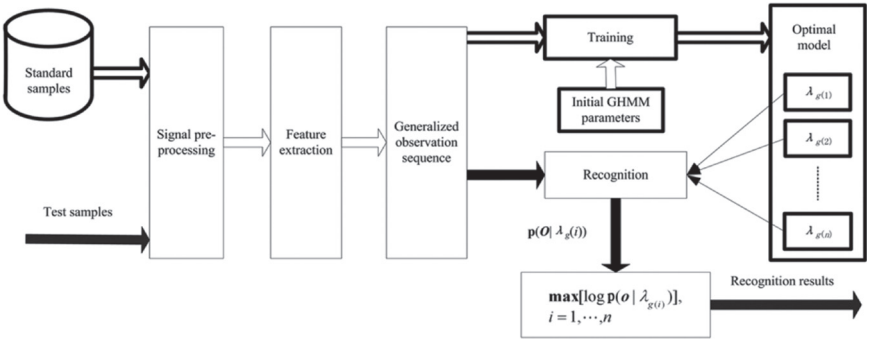


Figure 9.21 A HMM architecture for chatter detection [95].

9.3.4 Open-source datasets used in machining process monitoring

Since researchers often need to compare the performance of new algorithms using a benchmark dataset, it is worthwhile here to mention a commonly utilized open-source database of machining data; i.e., the 2010 Prognostics and Health Management (PHM) dataset. This database which include dynamometer, accelerometer, and noise emission sensor data collected during dry cutting to predict tool wear and life in CNC cutters. Six distinct cutters (named c1, c2, c3, c4, c5, and c6) were used to capture sensor data for the 2010 PHM dataset, although only three of them have data on pertinent tool wear (Fig. 9.22).

The NASA Ames dataset contains milling machine tests carried out under various operating settings, including 16 examples with a range of run counts. The Matsuura MC-510 V was used for the trials. The tool was a face-mill with six inserts. The experimental setup is illustrated in Fig. 9.23.

9.4 Signal decomposition methods

The consistency and visibility of information in collected signals from a machining environment are critical for learners' performance. Especially with the increasing research attention to DL methods, utilization of more advanced methods for signal decomposition has been advancing. However, the information collected from machining is nonstationary and nonlinear. For instance, as the tool wears out, vibration amplitude increases, or when the depth of cut increases, machining state can enter into chatter condition. Thus, only time domain or frequency domain analysis will not perform sufficiently under varying machining conditions.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

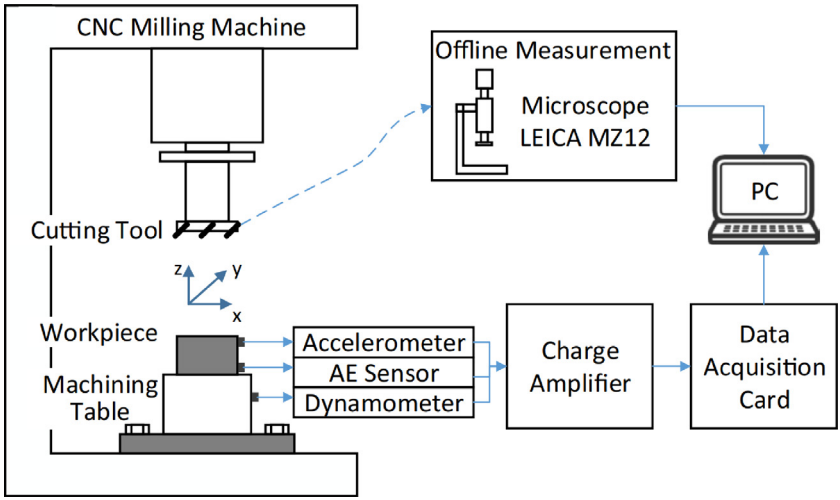


Figure 9.22 Schematic representation of the setup for the 2010 PHM dataset [96].

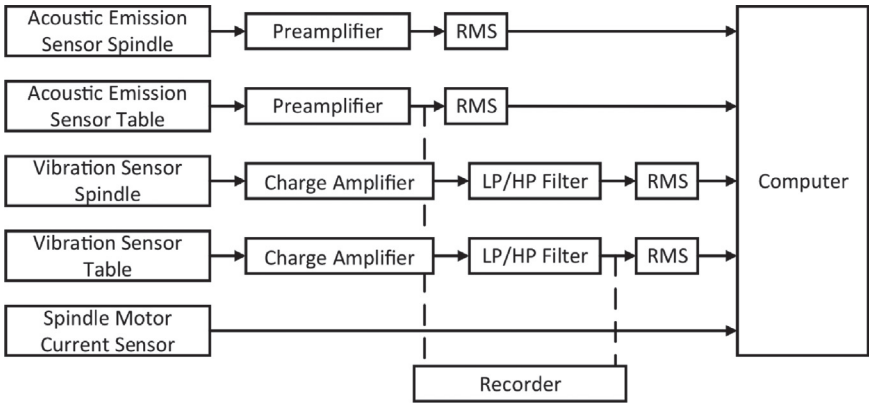


Figure 9.23 A schematic representation of the setup for NASA Ames dataset [96].

Recently, advanced time-frequency-based signal decomposition techniques have been used by many researchers. These techniques have been frequently implemented for chatter monitoring since the frequency bands where chatter occurs should be extracted from ambiguous signals with noise and other irrelevant information. These are short-time Fourier transform (STFT) [97], wavelet decomposition (WD) [98], empirical mode decomposition (EMD) [99], ensemble empirical mode decomposition (EEMD) [100] and variational mode decomposition (VMD) [101]. In most of these

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

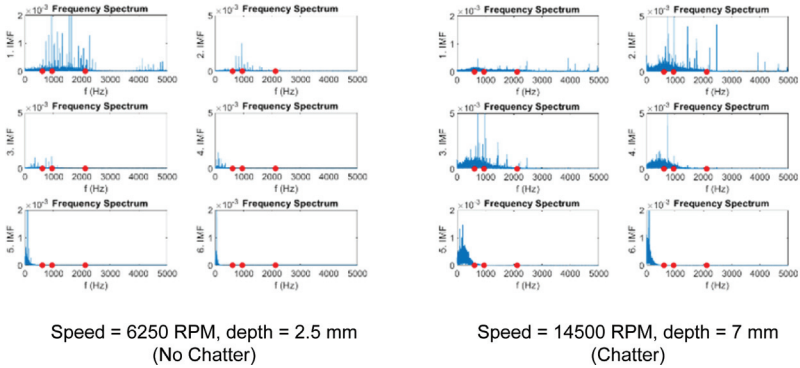


Figure 9.24 Intrinsic mode functions of two slot milling cuts (A) no chatter (B) chatter.

studies, used sensors are accelerometers for measurement of vibration and in some cases, dynamometers for measurement of cutting force, where measurements may be noisy and contain harmonics due to other vibration sources in a machine tool.

Huang et al. [102] first suggested nonlinear empirical analysis of non-stationary complex signals, referred to as empirical mode decomposition (EMD). Most typically, it is combined by Hilbert spectral analysis, referred to as Hilbert–Huang transform (HHT), which decomposes any nonstationary time series into the sum of its intrinsic mode functions (IMFs). Each IMF represents a zero-mean amplitude and frequency modulated component. Later on, an improved version of EMD which is known as EEMD is proposed by Wu and Huang, which resolves a frequent problem of EMD; mode-mixing in intermittent signals [103].

The first stage of HHT is decomposing the signal with one of the methods described above. A sample EEMD results is presented in Fig. 9.24, during slot milling of Al-7075 blocks. The original vibration signal has six frequent spectrum plots for each IMF. The red dots on the frequency axis indicate the modal frequencies of the tool-holder-spindle system at 613, 953, and 2113 Hz, respectively. When no chatter and chatter cuts are compared, the intensity increase around modal frequencies is clear in second, third, and fourth IMFs.

The second stage applies Hilbert spectral transform, which calculates the conjugate pair of each IMF. Fig. 9.25 provides the image representations of the two milling cuts. The color intensification in chatter cut is close to first and second modal frequencies. Although these Hilbert transformations are

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

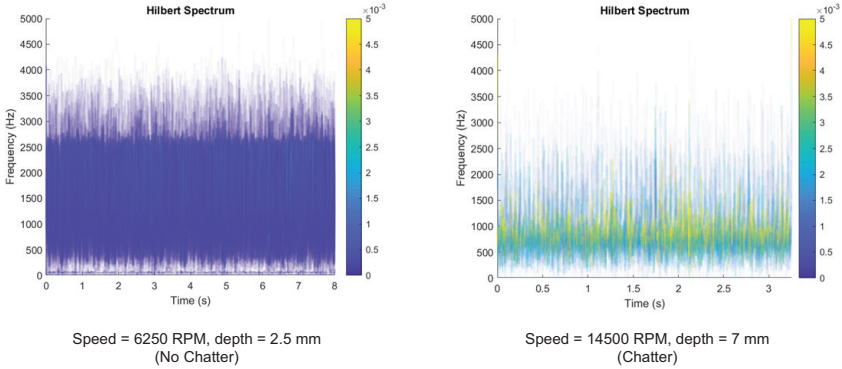


Figure 9.25 Hilbert–Huang transformations of two slot milling cuts (A) no chatter (B) chatter.

performed on all IMFs, selectively using the IMF which contains essential information for the learner often improves the performance.

9.5 Deep learning

The implementation of Industry 4.0 in manufacturing industries has been accompanied by the use of IoT systems more sensors, and higher capacity broad-band network systems, leading to the generation of "Big Data." For the cases that have large complex data, DL methods have been found to be appropriate [104]. Advantages of DL include: (1) its ability to deal with large data, (2) its satisfactory accuracy when the data are nonlinear, and (3) their implementation needing no tedious feature engineering tasks. Since the late 2010s, DL methods such as long short-term memory (LSTM) networks and convolutional neural networks (CNNs) have been applied to tool wear [105–107], chatter detection [108–110], surface roughness estimation [111–113], and energy estimation [114].

LSTM networks have been used widely in tool wear estimation, as they have shown high performance with sequentially constructed data in many applications. Long short-term memory networks are a particular type of recurrent neural networks (RNNs), in which a vanishing or exploding gradient problem is solved. These networks have a longer memory compared to other similar methods [115]. A typical architecture of LSTM is presented in Fig. 9.26.

Although an LSTM cell has the same inputs and outputs as an RNN cell, its forget gate gives the capability of learning long-term dependencies [96].

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

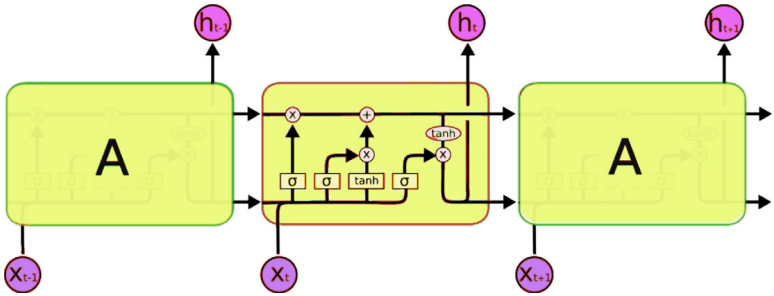


Figure 9.26 Long short-term memory network architecture.

The operations for each gate in an LSTM cell are given in Eqs. (9.7)–(9.11):

$$F_i^{(t)} = \sigma\left(b_i^F + \sum_j W_{ij}^F\left(x_j^{(t)} + h_j^{(t-1)}\right)\right) \tag{9.7}$$

$$I_i^{(t)} = \sigma\left(b_i^I + \sum_j W_{ij}^I\left(x_j^{(t)} + h_j^{(t-1)}\right)\right) \tag{9.8}$$

$$S_i^{(t)} = F_i^{(t)} S_i^{(t-1)} + I_i^{(t)} \tanh\left(b_i^I + \sum_j W_{ij}\left(x_j^{(t)} + h_j^{(t-1)}\right)\right) \tag{9.9}$$

$$O_i^{(t)} = \sigma\left(b_i^O + \sum_j W_{ij}^O\left(x_j^{(t)} + h_j^{(t-1)}\right)\right) \tag{9.10}$$

$$h_i^{(n)} = \tanh\left(S_i^{(n)}\right) \cdot O_i^{(n)} \tag{9.11}$$

where, (b^F and W^F), (b^I and W^I), and (b^O and W^O) denote the respective biases and weights of the forget gate, input gate, and output gate. σ is the sigmoid function that sets the forget gate to a value between 0 and 1. The forget gate updates the memory cell according to the current time input $x^{(t)}$ and the output of the last moment $h^{(t-1)}$, thus controlling the weight of the self-loop cell state as depicted in Eq. (9.7).

The input gate controls and updates the information incoming to the memory cell, per Eq. (9.8). The internal state of the LSTM cell is updated in Eq. (9.9). Using Eq. (9.10), the weights of the output cell are checked at the output gate. The output of the LSTM cell is calculated according to Eq. (9.11).

There are numerous tool wear monitoring and remaining useful life (RUL) frameworks developed based on coupling of LSTM networks with

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

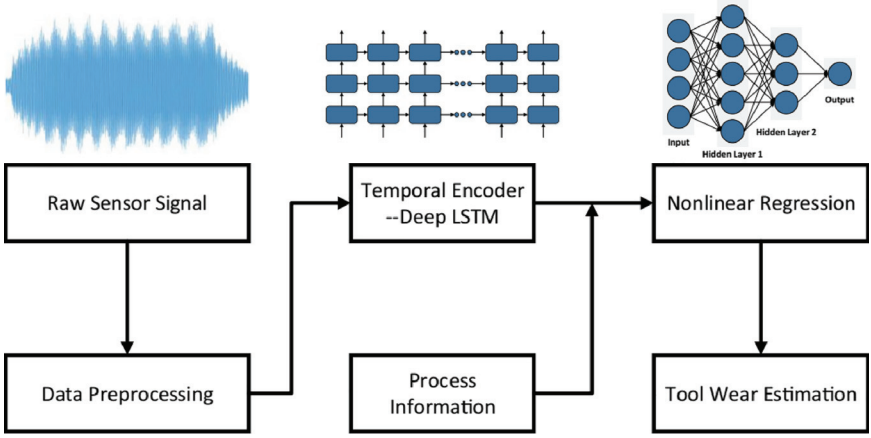


Figure 9.27 Long short-term memory network-based tool monitoring framework [96].

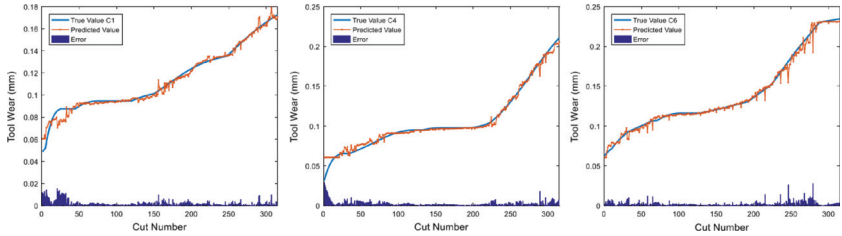


Figure 9.28 Results of long short-term memory network predictions using the 2010 PHM challenge dataset [96].

other networks such as BiLSTM [115], EMD–LSTM [116], LSTM–hidden Markov [117], wavelet pocket decomposition (WPD)–LSTM [118], Hurst exponent–CNN–LSTM [119], and residual–CNN–LSTM [120]. An earlier framework developed where an LSTM network is coupled by an ANN by Cai et al. [96] is provided in Fig. 9.27. As the number of studies using AI methods has increased, the need for benchmark data for comparing the performance of proposed methods has increased. Two frequently used datasets are Prognostics and Health Management (PHM) society 2010 challenge dataset [121] and NASA Prognostics Center of Excellence milling dataset [122]. Sample results of a LSTM–ANN framework utilizing C1, C4, and C6 tool wear measurements in the PHM 2010 dataset are shown in Fig. 9.28.

One of the most successful models used in chatter detection is CNN which has tremendous success for image classification tasks in many fields.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

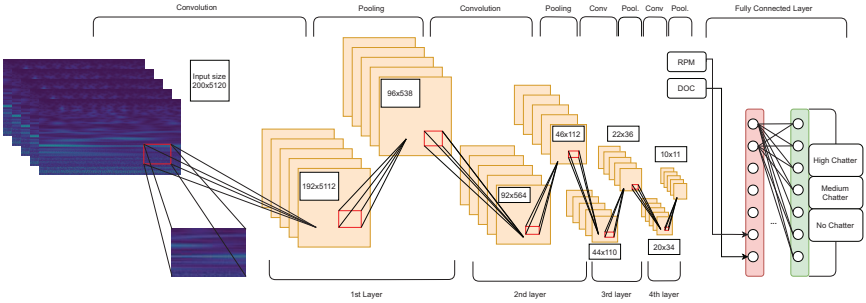


Figure 9.29 A convolutional neural network architecture for chatter detection [110].

A CNN model consists of three basic components: convolution layer, pooling layer, and fully connected layer. The convolutional layer is the backbone building block that uses several learnable convolution kernels to convolve the image to create a series of distinctive activation outputs [123]. The convolution operation can be expressed by the function given in Eq. (9.12):

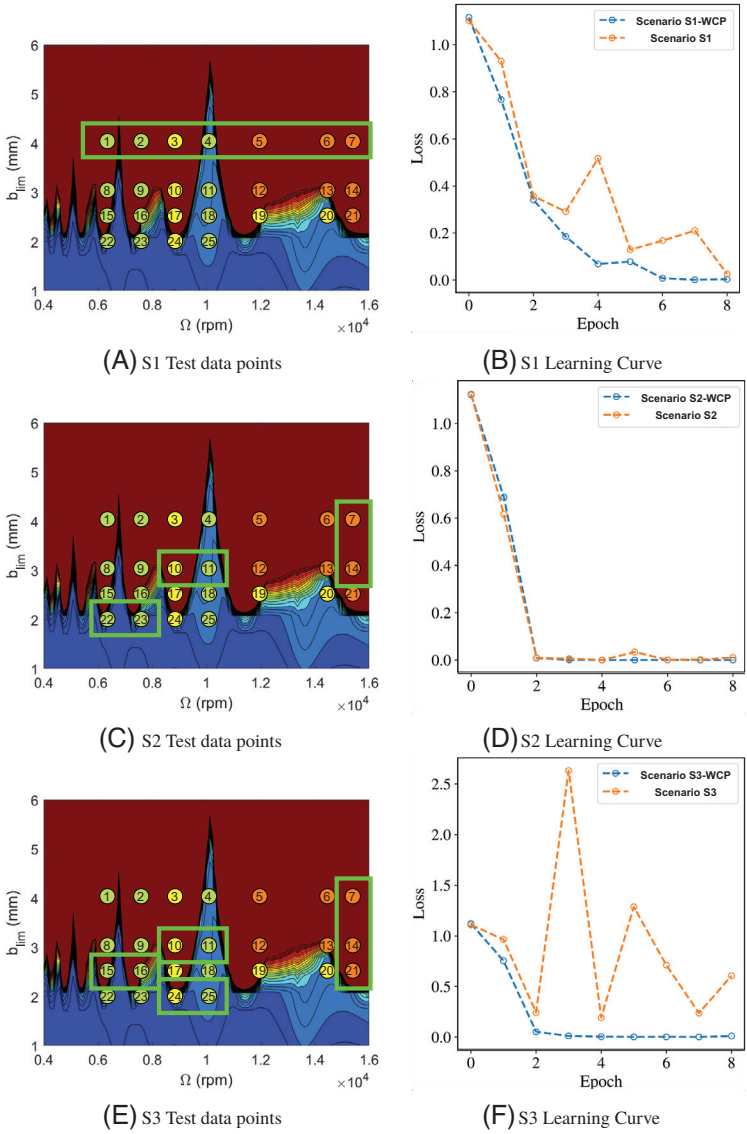
$$X_{p,t} = F\left(\sum_{q=1}^{Q} X^{\circ}_{q,t-1} K_{p,q,t} + B_{p,t}\right) \tag{9.12}$$

A custom CNN architecture using continuous wavelet transform (CWT) images generated by accelerometer signals as input to its front convolution layers, is shown in Fig. 9.29 [110]. In order to label input data, a hybrid approach was utilized by carrying a modal analysis on the tool-holder-spindle system and generating stability lobes. The labeling of validation points and training scenarios for three different training/test combinations are provided in Fig. 9.30. The architecture also uses cutting parameters, depth of cut, and spindle speed as the input to its fully connected layers, increasing its accuracy up to 99.88%. This approach also helped to reduce the loss in all cases rapidly. The cross-entropy loss function used in the training algorithm is given in Eq. (9.13).

$$L = -\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log\left(\hat{y}_n\right) + \left(1 - y_n\right)\log\left(1 - \hat{y}_n\right)\right] \tag{9.13}$$

where, $\hat{y}$ represents the predicted probability of the model and N represents the number of samples.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024



(A) S1 Test data points (B) S1 Learning Curve

(C) S2 Test data points (D) S2 Learning Curve

(E) S3 Test data points (F) S3 Learning Curve

Figure 9.30 Validation-loss results of three train/test scenarios including with cutting parameters [110].

9.6 Transfer learning

Machine learning and particularly DL models require extensive amount of data to perform sufficiently. This is an advantage in large-scale machine tool environments, but as each machining operation is different in

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

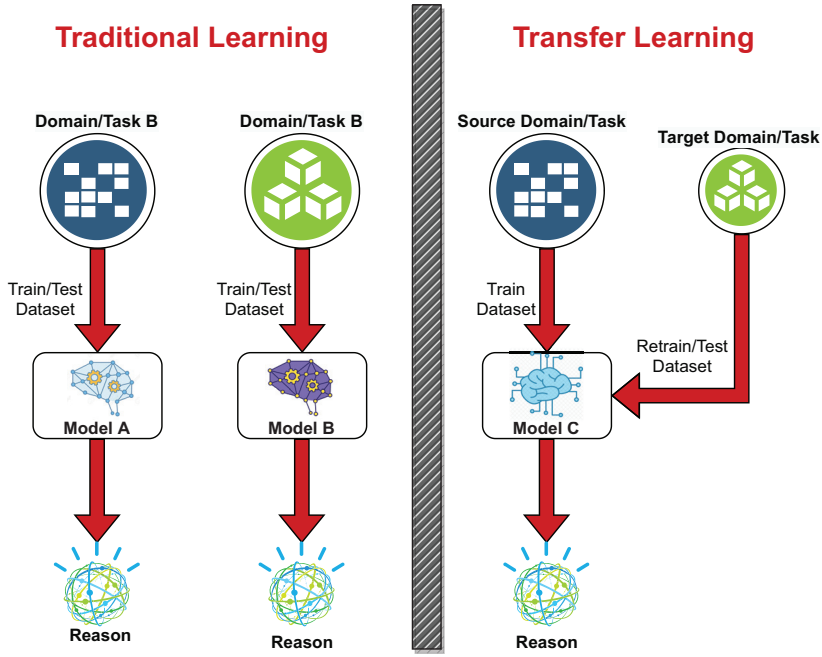


Figure 9.31 Traditional learning versus transfer learning.

many ways, this may become a liability. For instance, if milling is considered, the cutting tool, holder, and machine tool will vary depending on the complexity of a particular operation and the machinability, size, and weight of the workpiece. Hence, AI learner trained for one set of machining conditions will perform poorly in another set of conditions. The problem is having sufficient training data in one domain but having minimal or insufficient data in another domain [124]. Furthermore, labeling data in supervised learning is a major challenge when data size increases during the training phase.

An emerging approach for solution of variations between training and operating domains is transfer learning (TL) [125]. The core idea of TL is to train a model using data from a source domain/task where there is ample labeled data available and use this model for a target domain/task where data is limited or too costly to collect and label [126]. The Fig. 9.31 compares TL with conventional learning.

9.6.1 Cross-domain transfer learning

The most frequently used type of TL is cross-domain learning, where a model trained in a domain with sufficient labeled data is transferred to

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

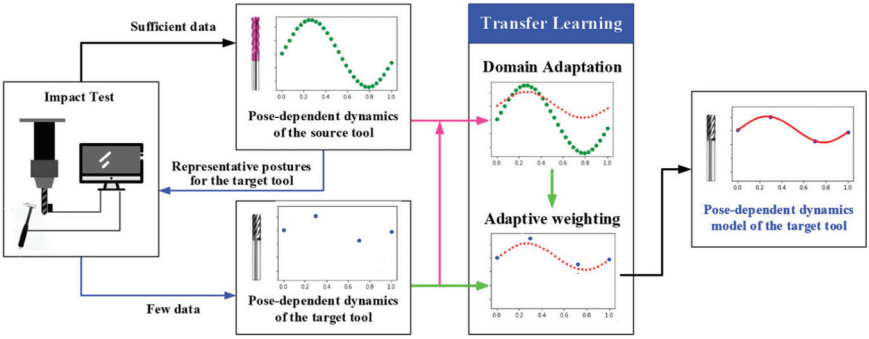


Figure 9.32 A tool tip dynamics domain adaptation framework [128].

another domain where labeled data is scarce. This type of transfer, also called domain adaption, is a broad concept. However, the algorithms in this technique can be classified under two categories: marginal distribution adaptation and conditional distribution adaptation [127]. A major challenge in stability monitoring of machining operation is modified tool-tip dynamic characteristics when parts or positions on a tool-holder-spindle system change. Chen et al. [128] applied domain adaptation to dynamic characteristics of a five-axis milling head. Fig. 9.32 depicts the pose-dependent varying characteristic of the milling head transferred to a new tool attached. As a result, a significant reduction in FRF measurement for the new tool was achieved, saving time-consuming tap tests.

In another study, chatter detection by measuring vibration at the spindle was achieved using SVM on varying operation characteristics in shoulder milling. The radial engagement in slot milling influences SLD of the system, hence modifying the chatter response. Two sets of measurement were used with 50% (DS#1 sufficient label data) and 70% (DS#2 few label data) radial engagements. The method used for testing the efficacy of the transfer is shown in Fig. 9.33. After the fine-tuning stage, with 70% radial engagement data, 92% accuracy was achieved.

9.6.2 Physics-guided transfer learning

Physics-guided TL is quite useful when the physics of the process can be modeled through physical principles with sufficient accuracy. The learner is trained using generated data through a physics-based system model and used in the learner's training. A recent studiy using this technique on chatter monitoring used few measured data for retraining the learner to fine-tune its classification capability in the new domain [130]. Another study used

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

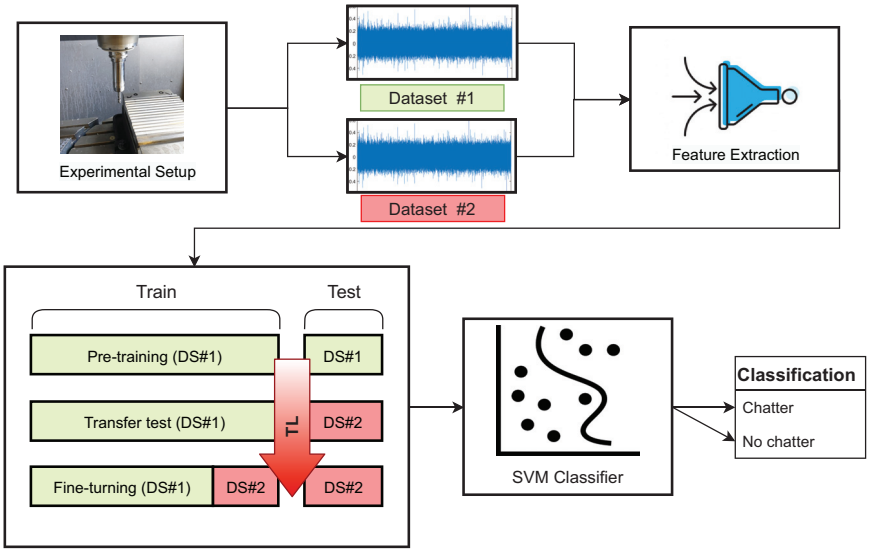


Figure 9.33 Cross-domain transfer learning for chatter detection [129].

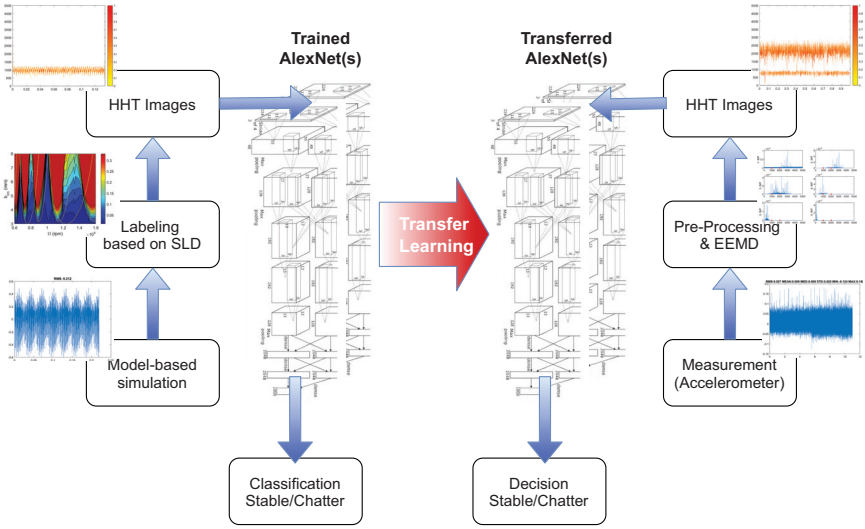


Figure 9.34 A physics-guided transfer learning framework for chatter detection [131].

AlexNet as the CNN classifier and HHT images as input and acheived accuracy higher than 90% without a fine-tuning stage with measured data [131].

Fig. 9.34 depicts the transfer learning of an AlexNet model [132,133] trained on acceleration signals generated by time-domain simulations of

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

end milling with inclined tooth cutter, a numerical computation method proposed by Smith and Tlusty [134]. The numerical generation of signals for each spindle speed and depth of cut combination across a grid provided a contour plot of SLD based on RMS of each signal. Based on the SLD automatic labeling of each signal as chatter or no chatter was enabled, virtually eliminating the time-consuming labeling process in supervised learning. Afterward each signal was decomposed into its IMF using an EEMD method, using only chatter-rich IMF bands HHT images were created as input to AlexNet. The measured data from the accelerometer during online monitoring was preprocessed with the same steps and then used to detect chatter with the trained AlexNet model.

9.7 Conclusion

This chapter put into perspective the state-of-the-art AI-based techniques and models used in MPM. Significant advances in ML during the last decade have opened up new frontiers in many aspects of manufacturing process monitoring. These advances are exemplified in machining monitoring and the current signal processing and decomposition methods, which are critical to a learner's performance.

Nevertheless, the ever-increasing flexibility and complexity of the machining process hinder the primary bottleneck of adaptation of AI-based monitoring and decision-making to the manufacturing industry. The concept of transfer learning seems to be a promising approach to overcoming this difficulty and should be examined on an industrial scale. Other issues are imbalance and correct labeling of Big Data since high performance supervised DL models require consistent, balanced, and labeled data. Synthetic data generation with generative models like variational auto encoders (VAE) and generative adversarial networks (GAN) can help alleviate the imbalance problem, and the use of hybrid models such as physical-driven TL can also eliminate the difficulty of labeling data.

There are many other opportunities for AI in MPM. Deep learning algorithms perform better in the cases with Big Data. However, obtaining sufficient data for training DL algorithms in manufacturing processes may be challenging for every possible fault scenario. A new generation of algorithms that combines physics-based models and machine learning models, often called physics-informed machine learning (PIML), can help cope with scarce and low fidelity data [135,136]. This class of ML algorithms promises the need

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

for minimum or no labeled data at all for training of deep neural networks [137].

Another opportunity for AI in manufacturing lies within the digital twin (DT) concept. Digital twins promise predicting and tracking the operational state and health of physical twin, simulating scenarios for improving efficiency without disrupting operations and real-time change in the course of actions when needed [138,139]. Hence AI and PIML have potential to play a major role in predicting and adapting to nonlinear behavior of physical systems.

On a final note, it is worth mentioning that as AI becomes more autonomous and ubiquitous and starts reasoning outside of data available to itself, with new opportunities to apply AI, more ethical issues will also arise. Hence, as in all major industries, the manufacturing industry also needs to focus on responsible, regulated, and trustworthy AI, which puts people first.

Acknowledgment

This study is supported by TUBITAK (The Scientific and Technological Research Council of Turkey) 1001 program (grant no: 118M414).

References

[1] R. Teti, K. Jemielniak, G. O'Donnell, D. Dornfeld, Advanced monitoring of machining operations, CIRP Ann. 59 (2) (2010) 717–739.
[2] P.J. Arrazola, T. Özel, D. Umbrello, M. Davies, I.S. Jawahir, Recent advances in modelling of metal machining processes, CIRP Ann. 62 (2) (2013) 695–718.
[3] W.A. Knight, G. Boothroyd, Fundamentals of Metal Machining and Machine Tools, 198, CRC Press, 2005.
[4] T.L. Schmitz, K.S. Smith, Machining Dynamics, Springer, 2014.
[5] H. Opitz, Investigation and calculation of the chatter behavior of: lathes and milling machines, CIRP Ann. Manuf. Technol. 18 (1979) 335–342.
[6] Y. Altintaş, E. Budak, Analytical prediction of stability lobes in milling, CIRP Ann. 44 (1) (1995) 357–362.
[7] E. Budak, Analytical models for high performance milling. Part II: process dynamics and stability, Int. J. Mach. Tools Manuf. 46 (12-13) (2006) 1489–1499.
[8] T. Kalsoom, S. Ahmed, P.M. Rafi-Ul-Shan, M. Azmat, P. Akhtar, Z. Pervez, M.A. Imran, M. Ur-Rehman, Impact of IoT on manufacturing Industry 4.0: a new triangular systematic review, Sustainability 13 (22) (2021) 12506.
[9] H.O. Unver, An ISA-95-based manufacturing intelligence system in support of lean initiatives, Int. J. Adv. Manuf. Technol. 65 (5) (2013) 853–866.
[10] J. Park, A.S.D.J. Park, S. Mackay, Practical Data Acquisition for Instrumentation and Control Systems, Newnes, 2003.
[11] Li Dan, J. Mathew, Tool wear and failure monitoring techniques for turning—a review, Int. J. Mach. Tools Manuf. 30 (4) (1990) 579–598.
[12] B. Sick, On-line and indirect tool wear monitoring in turning with artificial neural networks: a review of more than a decade of research, Mech. Syst. Sig. Process. 16 (4) (2002) 487–546.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[13] A. Siddhpura, R. Paurobally, A review of flank wear prediction methods for tool condition monitoring in a turning process, Int. J. Adv. Manuf. Technol. 65 (1-4) (2013) 371–393.
[14] G. Serin, B. Sener, A.M. Ozbayoglu, H.O. Unver, Review of tool condition monitoring in machining and opportunities for deep learning, Int. J. Adv. Manuf. Technol. 109 (2020) 953–974.
[15] S.Y. Wong, J.H. Chuah, H.J. Yap, Technical data-driven tool condition monitoring challenges for CNC milling: a review, Int. J. Adv. Manuf. Technol. 107 (11) (2020) 4837–4857.
[16] T. Mohanraj, S. Shankar, R. Rajasekar, N.R. Sakthivel, A. Pramanik, Tool condition monitoring techniques in milling process—a review, J. Mater. Res. Technol. 9 (1) (2020) 1032–1042.
[17] M. Kuntoğlu, E. Salur, M.K. Gupta, M. Sarıkaya, D.Y. Pimenov, A state-of-the-art review on sensors and signal processing systems in mechanical machining processes, Int. J. Adv. Manuf. Technol. 116 (9) (2021) 2711–2735.
[18] A.P. Kene, S.K. Choudhury, Analytical modeling of tool health monitoring system using multiple sensor data fusion approach in hard machining, Measurement 145 (2019) 118–129.
[19] M. Di Paolo Emilio, Data acquisition system, from fundamentals to applied design, 10, Springer, New York, 2013. https://link.springer.com/book/10.1007/978-1-4614-4214-1.
[20] O.B. Sezer, E. Dogdu, A.M. Ozbayoglu, Context-aware computing, learning, and Big Data in internet of things: a survey, IEEE Internet Things J. 5 (1) (2017) 1–27.
[21] B. Chen, J. Wan, A. Celesti, Di Li, H. Abbas, Q. Zhang, Edge computing in IoT-based manufacturing, IEEE Commun. Mag. 56 (9) (2018) 103–109.
[22] K. Cao, Y. Liu, G. Meng, Q. Sun, An overview on edge computing research, IEEE Access 8 (2020) 85714–85728.
[23] S. Sagiroglu, D. Sinanc, Big Data: a review, in: 2013 International Conference on Collaboration Technologies and Systems (CTS), IEEE, 2013, pp. 42–47.
[24] U. Sivarajah, M.M. Kamal, Z. Irani, V. Weerakkody, Critical analysis of Big Data challenges and analytical methods, J. Bus Res. 70 (2017) 263–286.
[25] A. Zheng, A. Casari, Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists, O'Reilly Media, Inc., 2018.
[26] U. Khurana, D. Turaga, H. Samulowitz, S. Parthasrathy, Cognito: automated feature engineering for supervised learning, in: 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), IEEE, 2016, pp. 1304–1307.
[27] W. Li, Z. Zhu, F. Jiang, G. Zhou, G. Chen, Fault diagnosis of rotating machinery with a novel statistical feature extraction and evaluation method, Mech. Syst. Sig. Process. 50 (2015) 414–426.
[28] G. Wang, Y. Zhang, C. Liu, Q. Xie, Y. Xu, A new tool wear monitoring method based on multi-scale PCA, J. Intell. Manuf. 30 (1) (2019) 113–122.
[29] K.P. Zhu, G.S. Hong, Y.S. Wong, A comparative study of feature selection for hidden Markov model-based micro-milling tool wear monitoring, Mach. Sci. Technol. 12 (3) (2008) 348–369.
[30] Y. Zhu, J. Wu, J. Wu, S. Liu, Dimensionality reduce-based for remaining useful life prediction of machining tools with multisensor fusion, Reliab. Eng. Syst. Saf. 218 (2022) 108179.
[31] T. Gittler, M. Glasder, E. Öztürk, M. Lüthi, L. Weiss, K. Wegener, International conference on advanced and competitive manufacturing technologies milling tool wear prediction using unsupervised machine learning, Int. J. Adv. Manuf. Technol. 117 (2021) 2213–2226.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[32] J. Ou, H. Li, G. Huang, Q. Zhou, A novel order analysis and stacked sparse auto-encoder feature learning method for milling tool wear condition monitoring, Sensors 20 (10) (2020) 2878.
[33] Y. Dun, L. Zhus, B. Yan, S. Wang, A chatter detection method in milling of thin-walled TC4 alloy workpiece based on auto-encoding and hybrid clustering, Mech. Syst. Sig. Process. 158 (2021) 107755.
[34] S. Wan, X. Li, Y. Yin, J. Hong, Milling chatter detection by multi-feature fusion and Adaboost-SVM, Mech. Syst. Sig. Process. 156 (2021) 107671.
[35] L. Wang, J. Pan, Y. Shao, Q. Zeng, X. Ding, Two new kurtosis-based similarity evaluation indicators for grinding chatter diagnosis under non-stationary working conditions, Measurement 176 (2021) 109215.
[36] J. Yuan, H. Shao, Y. Cai, X. Shi, Energy efficiency state identification of milling processing based on EEMD-PCA-ICA, Measurement 174 (2021) 109014.
[37] Y. He, P. Wu, Y. Li, Y. Wang, F. Tao, Y. Wang, A generic energy prediction model of machine tools using deep learning algorithms, Appl. Energy 275 (2020) 115402.
[38] X. Zhang, T. Yu, P. Xu, Ji Zhao, An intelligent sustainability evaluation system of micro milling, Rob. Comput. Integr. Manuf. 73 (2022) 102239.
[39] M.I. Qazi, M. Abas, R. Khan, W. Saleem, C.I. Pruncu, M. Omair, Experimental investigation and multi-response optimization of machinability of AA5005H34 using composite desirability coupled with PCA, Metals 11 (2) (2021) 235.
[40] H. Abdi, L.J. Williams, Principal component analysis, Wiley Interdiscip. Rev. Comput. Stat. 2 (4) (2010) 433–459.
[41] L.V. Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn Res. 9 (11) (2008) 2579–2605.
[42] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[43] Y. Wang, H. Yao, S. Zhao, Auto-encoder based dimensionality reduction, Neurocomputing 184 (2016) 232–242.
[44] J. Li, K. Cheng, S. Wang, F. Morstatter, R.P. Trevino, J. Tang, H. Liu, Feature selection: a data perspective, ACM Comput. Surveys (CSUR) 50 (6) (2017) 1–45.
[45] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, J. Mach. Learn Res. 3 (Mar) (2003) 1157–1182.
[46] T.Y. Wu, K.W. Lei, Prediction of surface roughness in milling process using vibration signal analysis and artificial neural network, Int. J. Adv. Manuf. Technol. 102 (1) (2019) 305–314.
[47] B. Sener, G. Serin, M. Ugur Gudelek, A. Murat Ozbayoglu, H.O. Unver, Intelligent chatter detection in milling using vibration data features and deep multi-layer perceptron, in: 2020 IEEE International Conference on Big Data (Big Data), IEEE, 2020, pp. 4759–4768.
[48] X. Zhang, S. Wang, W. Li, X. Lu, Heterogeneous sensors-based feature optimisation and deep learning for tool wear prediction, Int. J. Adv. Manuf. Technol. 114 (9) (2021) 2651–2675.
[49] T. Mohanraj, J. Yerchuru, H. Krishnan, R.S. Nithin Aravind, R. Yameni, Development of tool condition monitoring system in end milling process using wavelet features and Hoelder's exponent with machine learning algorithms, Measurement 173 (2021) 108671.
[50] C.K. Madhusudana, H. Kumar, S. Narendranath, Face milling tool condition monitoring using sound signal, Int. J. Syst. Assur. Eng. Manag. 8 (2) (2017) 1643–1653.
[51] K. Guo, J. Sun, B. Yang, J. Liu, Ge Song, C. Sun, Z. Jiang, et al., Tool condition monitoring in milling using a force singularity analysis approach, Int. J. Adv. Manuf. Technol. 107 (3) (2020) 1785–1792.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[52] S. Shankar, T. Mohanraj, A. Pramanik, Tool condition monitoring while using vegetable based cutting fluids during milling of Inconel 625, J. Adv. Manuf. Syst. 18 (04) (2019) 563–581.
[53] M.-Q. Tran, M.-K. Liu, M. Elsisi, Effective multi-sensor data fusion for chatter detection in milling process, ISA Trans. 125 (2022) 514–527.
[54] H. Cao, Y. Yue, X. Chen, X. Zhang, Chatter detection based on synchrosqueezing transform and statistical indicators in milling process, Int. J. Adv. Manuf. Technol. 95 (1) (2018) 961–972.
[55] Y. Chen, H. Li, L. Hou, J. Wang, X. Bu, An intelligent chatter detection method based on EEMD and feature selection with multi-channel vibration signals, Measurement 127 (2018) 356–365.
[56] M.C. Yesilli, F.A. Khasawneh, A. Otto, On transfer learning for chatter detection in turning using wavelet packet transform and ensemble empirical mode decomposition, CIRP J. Manuf. Sci. Technol. 28 (2020) 118–135.
[57] I. Oleaga, C. Pardo, J.J. Zulaika, A. Bustillo, A machine-learning based solution for chatter prediction in heavy-duty milling machines, Measurement 128 (2018) 34–44.
[58] G.S. Chen, Q.Z. Zheng, Online chatter detection of the end milling based on wavelet packet transform and support vector machine recursive feature elimination, Int. J. Adv. Manuf. Technol. 95 (1) (2018) 775–784.
[59] D. D'Addona, T. Segreto, A. Simeone, R. Teti, Ann tool wear modelling in the machining of nickel superalloy industrial products, CIRP J. Manuf. Sci. Technol. 4 (1) (2011) 33–37.
[60] P. Srinivasa Pai, T.N. Nagabhushana, P.K. Ramakrishna Rao, Tool wear estimation using resource allocation network, Int. J. Mach. Tools Manuf. 41 (5) (2001) 673–685.
[61] G. Li, Y. Wang, J. He, Q. Hao, H. Yang, J. Wei, Tool wear state recognition based on gradient boosting decision tree and hybrid classification RBM, Int. J. Adv. Manuf. Technol. 110 (1) (2020) 511–522.
[62] A. Farias, S.L.R. Almeida, S. Delijaicov, V. Seriacopi, Ed C. Bordinassi, Simple machine learning allied with data-driven methods for monitoring tool wear in machining processes, Int. J. Adv. Manuf. Technol. 109 (9) (2020) 2491–2501.
[63] C. Zhang, H. Zhang, Modelling and prediction of tool wear using LS-SVM in milling operation, Int. J. Computer Integr. Manuf. 29 (1) (2016) 76–91.
[64] A.M. Zain, H. Haron, S. Sharif, Prediction of surface roughness in the end milling machining using artificial neural network, Expert Syst. Appl. 37 (2) (2010) 1755–1768.
[65] M.R.S. Yazdi, S.Z. Chavoshi, Analysis and estimation of state variables in CNC face milling of al6061, Prod. Eng. 4 (6) (2010) 535–543.
[66] C.K. Madhusudana, H. Kumar, S. Narendranath, Fault diagnosis of face milling tool using decision tree and sound signal, Mater. Today Proc. 5 (5) (2018) 12035–12044.
[67] D. Yu Pimenov, A. Bustillo, T. Mikolajczyk, Artificial intelligence for automatic prediction of required surface roughness by monitoring wear on face mill teeth, J. Intell. Manuf. 29 (5) (2018) 1045–1061.
[68] K. Kadirgama, M.M. Noor, M.M. Rahman, Optimization of surface roughness in end milling using potential support vector machine, Arab. J. Sci. Eng. 37 (8) (2012) 2269–2275.
[69] R. Ak, M.M. Helu, S. Rachuri, Ensemble neural network model for predicting the energy consumption of a milling machine, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 57113, American Society of Mechanical Engineers, 2015 V004T05A056.
[70] A. Tohry, S.C. Chelgani, S.S. Matin, M. Noormohammadi, Power-draw prediction by random forest based on operating parameters for an industrial ball mill, Adv. Powder Technol. 31 (3) (2020) 967–972.
[71] Y.-S. Park, S. Lek, Artificial neural networks: multilayer perceptron for ecological modeling, Dev. Environ. Model. 28 (2016) 123–140.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[72] A.M. Khorasani, M.R.S. Yazdi, Development of a dynamic surface roughness monitoring system based on artificial neural networks (ANN) in milling operation, Int. J. Adv. Manuf. Technol. 93 (1) (2017) 141–151.
[73] V. Pourmostaghimi, M. Zadshakoyan, M.A. Badamchizadeh, Intelligent model-based optimization of cutting parameters for high quality turning of hardened AISI D2, AI EDAM 34 (3) (2020) 421–429.
[74] P. Gupta, B. Singh, Investigation of tool chatter using local mean decomposition and artificial neural network during turning of AL 6061, Soft Comput. 25 (16) (2021) 11151–11174.
[75] M. Lamraoui, M. Barakat, M. Thomas, M. El Badaoui, Chatter detection in milling machines by neural network classification and feature selection, J. Vib. Control 21 (7) (2015) 1251–1266.
[76] H. Cao, K. Zhou, X. Chen, X. Zhang, Early chatter detection in end milling based on multi-feature fusion and 3 σ criterion, Int. J. Adv. Manuf. Technol. 92 (9) (2017) 4387–4397.
[77] Y. Chen, H. Li, X. Jing, L. Hou, X. Bu, Intelligent chatter detection using image features and support vector machine, Int. J. Adv. Manuf. Technol. 102 (5) (2019) 1433–1442.
[78] W.S. Noble, What is a support vector machine? Nat. Biotechnol. 24 (12) (2006) 1565–1567.
[79] X. Liao, G. Zhou, Z. Zhang, J. Lu, J. Ma, Tool wear state recognition based on GWO–SVM with feature selection of genetic algorithm, Int. J. Adv. Manuf. Technol. 104 (1) (2019) 1051–1063.
[80] L. Ding, Y. Sun, Z. Xiong, Early chatter detection based on logistic regression with time and frequency domain features, in: 2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), IEEE, 2017, pp. 1052–1057.
[81] D. Kong, Y. Chen, N. Li, Gaussian process regression for tool wear prediction, Mech. Syst. Sig. Process. 104 (2018) 556–574.
[82] B. Lela, D. Bajić, S. Jozić, Regression analysis, support vector machines, and Bayesian neural network approaches to modeling surface roughness in face milling, Int. J. Adv. Manuf. Technol. 42 (11-12) (2009) 1082–1088.
[83] D.Y. Pimenov, A.T. Abbas, M.K. Gupta, I.N. Erdakov, M.S. Soliman, M.M. El Rayes, Investigations of surface quality and energy consumption associated with costs and material removal rate during face milling of AISI 1045 steel, Int. J. Adv. Manuf. Technol. 107 (7) (2020) 3511–3525.
[84] L. Fernández-Robles, L. Sánchez-González, J. Díez-González, M. Castejón- Limas, H. Pérez, Use of image processing to monitor tool wear in micro milling, Neurocomputing 452 (2021) 333–340.
[85] Z. Li, G. Wang, G. He, Milling tool wear state recognition based on partitioning around medoids (PAM) clustering, Int. J. Adv. Manuf. Technol. 88 (5-8) (2017) 1203–1213.
[86] Y. Chen, H. Li, L. Hou, X. Bu, S. Ye, D. Chen, Chatter detection for milling using novel p-leader multifractal features, J. Intell. Manuf. 33 (2020) 121–135.
[87] S. Tangjitsitcharoen, T. Saksri, S. Ratanakuakangwan, Advance in chatter detection in ball end milling process by utilizing wavelet transform, J. Intell. Manuf. 26 (3) (2015) 485–499.
[88] L. Xu, C. Huang, C. Li, J. Wang, H. Liu, X. Wang, An improved case based reasoning method and its application in estimation of surface quality toward intelligent machining, J. Intell. Manuf. 32 (1) (2021) 313–327.
[89] K. Shi, D. Zhang, J. Ren, Optimization of process parameters for surface roughness and microhardness in dry milling of magnesium alloy using Taguchi with grey relational analysis, Int. J. Adv. Manuf. Technol. 81 (1) (2015) 645–651.
[90] Y. Yang, Y. Wang, Q. Liao, J. Pan, J. Meng, H. Huang, CNC corner milling parameters optimization based on variable-fidelity metamodel and improved MOPSO regarding energy consumption, Int. J. Precis. Eng. Manuf. Green Technol. 9 (2021) 977–995.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[91] X. Zhu, A.B. Goldberg, Introduction to Semi-Supervised Learning, in: Synthesis Lectures on Artificial Intelligence and Machine Learning, 3, Cham, Springer, 2009, pp. 1–130.
[92] C. Qiu, K. Li, B. Li, X. Mao, S. He, C. Hao, L. Yin, Semi-supervised graph convolutional network to predict position-and speed-dependent tool tip dynamics with limited labeled data, Mech. Syst. Sig. Process. 164 (2022) 108225.
[93] J. Wang, Y. Li, R. Zhao, R.X. Gao, Physics guided neural network for machining tool wear prediction, J. Manuf. Syst. 57 (2020) 298–310.
[94] M. Grzenda, A. Bustillo, Semi-supervised roughness prediction with partly unlabeled vibration data streams, J. Intell. Manuf. 30 (2) (2019) 933–945.
[95] F.-Y. Xie, Y.-M. Hu, Bo Wu, Y. Wang, A generalized hidden Markov model and its applications in recognition of cutting states, Int. J. Precis. Eng. Manuf. 17 (11) (2016) 1471–1482.
[96] W. Cai, W. Zhang, X. Hu, Y. Liu, A hybrid information model based on long short-term memory network for tool condition monitoring, J. Intell. Manuf. 31 (6) (2020) 1497–1510.
[97] I. Yesilyurt, H. Ozturk, Tool condition monitoring in milling using vibration analysis, Int. J. Prod. Res. 45 (4) (2007) 1013–1028.
[98] H. Cao, Y. Lei, Z. He, Chatter identification in end milling process using wavelet packets and Hilbert–Huang transform, Int. J. Mach. Tools Manuf. 69 (2013) 11–19.
[99] Y. Fu, Y. Zhang, H. Zhou, D. Li, H. Liu, H. Qiao, X. Wang, Timely online chatter detection in end milling process, Mech. Syst. Sig. Process. 75 (2016) 668–688.
[100] H. Cao, K. Zhou, X. Chen, Chatter identification in end milling process based on EEMD and nonlinear dimensionless indicators, Int. J. Mach. Tools Manuf. 92 (2015) 52–59.
[101] C. Liu, L. Zhu, C. Ni, Chatter detection in milling process based on VMD and energy entropy, Mech. Syst. Sig. Process. 105 (2018) 169–182.
[102] N.E. Huang, Z. Shen, S.R. Long, M.C. Wu, H.H. Shih, Q. Zheng, N.-C. Yen, C.C. Tung, H.H. Liu, The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis, in: Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, 454, 1998, pp. 903–995.
[103] Z. Wu, N.E. Huang, Ensemble empirical mode decomposition: a noise-assisted data analysis method, Adv. Adapt. Data Anal. 1 (01) (2009) 1–41.
[104] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[105] Z. Huang, J. Zhu, J. Lei, X. Li, F. Tian, Tool wear predicting based on multi-domain feature fusion by deep convolutional neural network in milling operations, J. Intell. Manuf. 31 (4) (2020) 953–966.
[106] Y. Fu, Y. Zhang, Y. Gao, H. Gao, T. Mao, H. Zhou, D. Li, Machining vibration states monitoring based on image representation using convolutional neural networks, Eng. Appl. Artif. Intell. 65 (2017) 240–251.
[107] X.-C. Cao, B.-Q. Chen, B. Yao, W.-P. He, Combining translation-invariant wavelet frames and convolutional neural network for intelligent tool wear state identification, Comput. Ind. 106 (2019) 71–84.
[108] M.-Q. Tran, M.-K. Liu, Q.-V. Tran, Milling chatter detection using scalogram and deep convolutional neural network, Int. J. Adv. Manuf. Technol. 107 (3) (2020) 1505–1516.
[109] W. Zhu, J. Zhuang, B. Guo, W. Teng, F. Wu, An optimized convolutional neural network for chatter detection in the milling of thin-walled parts, Int. J. Adv. Manuf. Technol. 106 (9) (2020) 3881–3895.
[110] B. Sener, M.U. Gudelek, A.M. Ozbayoglu, H.O. Unver, A novel chatter detection method for milling using deep convolution neural networks, Measurement 182 (2021) 109689.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[111] A.P. Rifai, H. Aoyama, N.H. Tho, S.Z. Md Dawal, N.A. Masruroh, Evaluation of turned and milled surfaces roughness using convolutional neural network, Measurement 161 (2020) 107860.
[112] W. Sun, B. Yao, B. Chen, Y. He, X. Cao, T. Zhou, H. Liu, Noncontact surface roughness estimation using 2D complex wavelet enhanced ResNet for intelligent evaluation of milled metal surface quality, Appl. Sci. 8 (3) (2018) 381.
[113] P.-M. Huang, C.-H. Lee, Estimation of tool wear and surface roughness development using deep learning and sensors fusion, Sensors 21 (16) (2021) 5338.
[114] G. Serin, B. Sener, M. Ugur Gudelek, A. Murat Ozbayoglu, H.O. Unver, Deep multi-layer perceptron based prediction of energy efficiency and surface quality for milling in the era of sustainability and Big Data, Procedia Manuf. 51 (2020) 1166–1177.
[115] X. Wu, J. Li, Y. Jin, S. Zheng, Modeling and analysis of tool wear prediction based on SVD and BiLSTM, Int. J. Adv. Manuf. Technol. 106 (9) (2020) 4391–4399.
[116] J.-T. Zhou, Xu Zhao, J. Gao, Tool remaining useful life prediction method based on LSTM under variable working conditions, Int. J. Adv. Manuf. Technol. 104 (9) (2019) 4715–4726.
[117] Z. Tao, Q. An, G. Liu, M. Chen, A novel method for tool condition monitoring based on long short-term memory and hidden Markov model hybrid framework in high-speed milling Ti-6Al-4V, Int. J. Adv. Manuf. Technol. 105 (7) (2019) 3165–3182.
[118] H. Habbouche, T. Benkedjouh, N. Zerhouni, Intelligent prognostics of bearings based on bidirectional long short-term memory and wavelet packet decomposition, Int. J. Adv. Manuf. Technol. 114 (1) (2021) 145–157.
[119] X. Zhang, X. Lu, W. Li, S. Wang, Prediction of the remaining useful life of cutting tool using the Hurst exponent and CNN-LSTM, Int. J. Adv. Manuf. Technol. 112 (7) (2021) 2277–2299.
[120] H. Sun, J. Zhang, R. Mo, X. Zhang, In-process tool condition forecasting based on a deep learning method, Rob. Comput. Integr. Manuf. 64 (2020) 101924.
[121] 2010 PHM Society Conference Data Challenge PHM Society, May 18, 2010, [online] Available: https://phmsociety.org/phm_competition/2010-phm-society-conference-data-challenge.
[122] K.F. Goebel, Management of Uncertainty in Sensor Validation, Sensor Fusion, and Diagnosis of Mechanical Systems Using Soft Computing Techniques, University of California, Berkeley, 1996.
[123] Y. Guo, Y. Zhou, Z. Zhang, Fault diagnosis of multi-channel data by the CNN with the multilinear principal component analysis, Measurement 171 (2021) 108513.
[124] S.J. Pan, Q. Yang, A survey on transfer learning, IEEE Trans. Knowl. Data Eng. 22 (10) (2009) 1345–1359.
[125] C. Li, S. Zhang, Yi Qin, E. Estupinan, A systematic review of deep transfer learning for machinery fault diagnosis, Neurocomputing 407 (2020) 121–135.
[126] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, Q. He, A comprehensive survey on transfer learning, Proc. IEEE 109 (1) (2020) 43–76.
[127] Z. Wan, R. Yang, M. Huang, N. Zeng, X. Liu, A review on transfer learning in EEG signal analysis, Neurocomputing 421 (2021) 1–14.
[128] G. Chen, Y. Li, Xu Liu, Pose-dependent tool tip dynamics prediction using transfer learning, Int. J. Mach. Tools Manuf. 137 (2019) 30–41.
[129] H.O. Unver, B. Sener, Exploring the potential of transfer learning for chatter detection, Procedia Comput. Sci. 200 (2022) 151–159.
[130] M. Postel, B. Bugdayci, K. Wegener, Ensemble transfer learning for refining stability predictions in milling using experimental stability states, Int. J. Adv. Manuf. Technol. 107 (9) (2020) 4123–4139.
[131] H.O. Unver, B. Sener, A novel transfer learning framework for chatter detection using convolutional neural networks, J. Intell. Manuf. 34 (2023) 1105–1124.
[132] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. 25 (2012) 1097–1105.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

[133] A. Krizhevsky. One weird trick for parallelizing convolutional neural networks. arXiv preprint arXiv:1404.5997, (2014).
[134] S. Smith, J. Tlusty, An overview of modeling and simulation of the milling process, J. Manuf. Sci. Eng. 113 (2) (1991) 169–175.
[135] G.C.Y. Peng, M. Alber, A.B. Tepole, W.R. Cannon, S. De, S. Dura-Bernal, K. Garikipati, G. Karniadakis, W.W. Lytton, P. Perdikaris, et al., Multiscale modeling meets machine learning: what can we learn? Arch. Comput. Meth. Eng. 28 (3) (2021) 1017–1037.
[136] Y. Guo, X. Cao, B. Liu, M. Gao, Solving partial differential equations using deep learning and physical constraints, Appl. Sci. 10 (17) (2020) 5917.
[137] G. Em Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (6) (2021) 422–440.
[138] M. Singh, E. Fuenmayor, E.P. Hinchy, Y. Qiao, N. Murray, D. Devine, Digital twin: origin to future, Appl. Syst. Innov. 4 (2) (2021) 36.
[139] E. VanDerHorn, S. Mahadevan, Digital twin: generalization, characterization and implementation, Decision Support Syst. 145 (2021) 113524.

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Index

Page numbers followed by "*f*" and "*t*" indicate figures and tables respectively.

A

Activation functions, 51, 73
Active learning, 16*f*
Adaptive radial basis function (ARBF), 149
Algebraic models for optimization (ALAMO), 46, 47*f*
Algebra learning via elastic net (ALVEN), 53, 59
  method, 48, 49*f*
  model-fitting algorithm, 48
Algorithmic perspective, 13
Anomaly detection, 6
ARGONAUT, 148
  algorithm, 151, 154
  framework, 154
Artificial intelligence methods, 265, 267*f*
Artificial neural network (ANN), 7, 50, 203, 285, 287
  architectures, 55
Asymmetric quadratic functions, 246
Asymptotic properties, 188
Autocorrelation function (ACF), 58
Autoencoders, 7, 8*f*
Auto-Keras, 56
Automated learning of algebraic models for optimization (ALAMO), 118
  framework, 148
Automated machine learning (AutoML), 55, 56, 60
  approach, 60
  literature, 56
  methods, 56
  packages, 56
  software, 56
  strategy, 55
Autoregressive integrated moving average (ARIMA) model, 54
Auto-Sklearn, 56
AutoWEKA, 56

B

Backpropagation, 51
Bagging, 44
Bayesian classifiers, 281
Bayesian information criterion (BIC), 47
Bayesian optimization
  methods, 55
  techniques, 106
Best linear unbiased estimator (BLUE), 43
Bi-level optimization problems, 160
Black-box models, 40, 44
Blended sparse identification methods, 118
Burgers equation, 111, 112

C

Canonical variate analysis (CVA), 58
Cationic surfactants, 86
CDSP algorithm, 257
Chatter detection, 318
Chemical and biochemical processes, 185
Classification algorithms, 3
CNN architecture optimization, 76
CNN sensor, 75
COBALT algorithm, 146
Computing equipment performance, 103
Constrained ELM parameter learning, 256
Constrained optimization problem, 252
Contracting dynamical systems primitives (CDSP), 248
Contraction analysis constraints, 251
Contractive strategies, 116
Controller parameters, 214
Convexification, 30
Convexified convolutional neural network, 32
Convexified deep neural network, 31
Convex optimization methods, 30
Convex quadratic program, 48
Convolutional neural network (CNN), 22, 71

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY
Copyright Elsevier 2024

architecture, 72*f*
architecture pipeline, 23*f*
Convolution blocks, 74
CORS algorithm, 147
CORS–RBF framework, 147
Cross-domain transfer learning, 339
CutPRO software, 315

D

DALVEN, 58
DAQ hardware, 313
Data
acquisition systems, 311
analytics triangle, 58
augmentation, 75
based models, 186
driven algorithms, 136
driven methods, 131
driven model, 55
driven modeling algorithms, 60
driven multiobjective optimization, 154
driven optimization, 135, 152
Decision Tree algorithm, 280
Decision Trees model, 277
Deductive inference, 12
Deep learning, 308
methods, 242, 287
networks, 51
Deep neural network (DNN), 16*f*
Deep reinforcement learning (DRL), 223
Derivative-free optimization (DFO) algorithm, 145
Detection of dimethyl methylphosphonate (DMMP), 81
Discrete-time model, 186
Distilling governing equations, 105
DOMINO framework, 153
Dynamical systems, 246
Dynamic movement primitives (DMP), 245
formulation models, 245
Dynamics, ridge regression, 59
Dynamic system models, 243, 258*f*

E

EGO algorithm, 146
Elastic net, 48
Element-by-element multiplication, 52, 52
Encoders, 7
EndoNet, 79
architecture, 80*f*
Ensemble learning technique, 18, 19*f*
ENTMOOT framework, 150
Epoch, 197
Error vector, 42
Extreme learning machines (ELM), 246, 250
network, 258
neurons, 250

F

Fast Bayesian Optimization for Data Analytic Pipelines, 55
Feature extraction procedure, 328*f*
Feed-forward artificial neural networks, 203
Feed forward neural networks, 75
Fick's laws, 40
Filtering-based approaches, 295
First-principles, 40
Flow cytometry, 78
Fourier's laws, 40
Fully connected network (FCN), 21

G

Gated recurrent unit (GRU), 27, 52
Gaussian mixture model (GMM), 246, 250
model parameters, 252
parameter, 251, 257
parameters, 252
Gaussian mixture modeling (GMM), 285
Gaussian process, 55, 183
classifier, 89
Gauss-Markov Theorem, 43
GCN model, 87
Generalized least squares (GLS), 43
Generalized Pattern Search (GPS), 139
algorithm, 140
Generative adversarial network (GAN) architecture overview, 5*f*
Generic action-reward feedback loop, 11*f*
Genetic and evolutionary algorithms, 152
Genetic programming algorithms, 114
Global and Local Optimization using Direct Search (GLODS), 141
Global-constrained data-driven optimization algorithm, 154

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Global optimization techniques, 141, 146
GNN architectures, 73
Google searches, 40
Governing equations, 104
Gramian angular difference fields (GADF), 87
Gramian angular fields (GAF), 87
Gramian angular summation fields (GASF), 87
Graph, 68
Graphical processing units (GPUs), 22
Graph neural networks, 25
Graphs, 65
Grey-box modeling, 40
GSS algorithms, 143

H

Hadamard product, 52
Hidden Markov Models (HMMs), 293
Hidden state, 50
High-fidelity plant models, 209
Hilbert spectral transform, 333
HoloLens, 245*f*
Human-robot collaborative task, 243*f*
Human vision and auditory system, 67
Human vision system, 66, 69
Hybrid
  learning, 8
  learning architecture, 9*f*
  models, 186
  techniques, 143
HyNet, 25
Hyperplanes, 45

I

Image-based feedback control, 91
ImageNet, 23
ImgAug, 75
Inductive learning, 12
Input-variables values, 2
Internal model control, 217
Inverse model-based methods, 217
Inverse optimal control (IOC), 247
Inverse reinforcement learning (IRL), 247
Irregular grid, 68
Iterative learning control (ILC), 226

L

LAB space, 70
LASA dataset, 259
LASA human handwriting dataset, 257
LC sensor, 81
Learning models, 258
  holistic view of, 2
Learning task-level movement primitives, 249
Least absolute shrinkage and selection operator (LASSO), 116, 118
  regression, 116
Least-squares methods, 42
Linear and nonlinear regression tools, 112
Linear regression, 41
Local optimization techniques, 143
Long short-term memory (LSTM)
  architecture, 335*f*
  networks, 21
Long–short-term memory (LSTM), 51
  networks, 52
  units, 52
Low-dimensional word embeddings, 27
LSVM classification, 83
Lyapunov and barrier constraints, 254
Lyapunov-based LfD approaches, 248
Lyapunov constraints, 249, 255, 256
Lyapunov function-based dynamic movements, 246
Lyapunov stability tools, 249

M

Machine learning (ML), 39, 63, 181, 193, 241, 308, 335*f*
  algorithms, 316
  applications, 182
  framework, 84*f*
  methods, 18, 33
  models, 186
  optimization algorithms, 30
  techniques, 83
  tools, 1
MADS algorithm, 140
Manufacturing data, 41
Manufacturing processes, 111, 190
Markov decision processes, 10
Markov models, 245

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY
Copyright Elsevier 2024

Mathematical objects, 65
Mathematical programming, 118
Mean squared residual, 111
Mesh, 139
Meta-learning procedure, 248
Mixed-integer linear programming (MILP), 118
ML–Oracle block, 193
ML techniques, 84
ML technologies, 64
MNIST dataset, 22
Model-based algorithms, 136, 143
Model-based control methods, 227
Model-based estimation and control theory, 118
Model parsimony, 106
Model predictive controller, 193, 216
Modern automation systems, 63
Modern chemical processes, 190
Movement primitives (MP), 245
Multichannel data representation approach, 79
Multicollinearity, 58
Multi-instance learning, 9
Multilayer perceptrons (MLPs), 20
Multiple linear regression, 42
Multiple techniques, 42
Multispectral and hyperspectral cameras, 70
Multistage distillation process, 74
Multitask learning, 14
  architecture overview of, 15*f*, 16*f*
Multivariable output error state space (MOESP), 58
Multivariate process monitoring, 91

N

Named entity recognition, 26
National Institute of Standards and Technology (MNIST) database, 23
Natural language generation (NLG), 25
Natural language processing (NLP), 9, 25
Natural language understanding (NLU), 25
Navier–Stokes equation, 111
Nelder–Mead algorithm, 137
Nelder–Mead approach, 137
Neural architecture search (NAS), 76
Neurally-imprinted Lyapunov candidate (NILC), 246
Neurally-imprinted vector fields (NiVF), 246
Neural network (NN), 246
Neural networks, 4
  approximation, 111
Neuron, 51
Newton's laws, 40
NEWUOA algorithm, 144
NOMAD software, 140
Nonisothermal batch reactor system, 124, 125*f*
Nonlinear autoregressive model with exogenous inputs (NARX), 118
Nonlinear blackbox modeling method, 44
Nonlinear regression methods, 44, 45
Nonlinear regression model, 46
NOWPAC algorithm, 145
NP-hard optimization, 46

O

Observational teaching, 243
Olfactory signals (data), 63
Online learning, 16
Optimization-based methods, 46
Optimization formulation, 48
Ordinary least squares (OLS), 43
Ordinary least squares (OLS) regression, 116
Overfitting, 8

P

Parameter estimation, 108, 193
Partial least squares (PLS), 59
Particle Swarm Optimization (PSO), 141
  algorithm, 141
Pattern matching technique, 69*f*
Pearson correlation coefficient, 57, 124
Percentage least squares (PLS), 43
PHM dataset, 331
Physics-guided transfer learning, 311
Physics-informed machine learning, 108
PINN architecture, 108
Plant dynamics, 185
Powell's methods, 144
Pressure swing adsorption (PSA) systems, 151
Principal component analysis (PCA), 19, 268
Priori, 108
Probabilistic road maps, 247

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY Copyright Elsevier 2024

Probably approximate correct (PAC) learning, 32
Process data, 40, 56
Processes operating modes, 188

Q

QSPR models, 86
Quadratic program (QP), 256

R

Radial basis function (RBF), 45
Random forest, 18, 44
  models, 45
  regression, 44*f*
Real-valued nonnegative function, 254
Rectified linear unit (ReLU), 74
Recurrent neural networks (RNNs), 21, 22*f*, 205
Recycling process streams, 188, 190
Regression, 3
  algorithms, 4
  based approaches, 112
  visualization of, 4*f*
Reinforcement learning (RL), 10, 223, 247
Representation learning, 66
Residual block, 24*f*
Residual networks, 23
RGB-D sensor, 242
RGB space, 70
RNN-based model identification, 54
Robotics researchers, 241
Root mean squared error (RMSE), 4
Runge-Kutta integration schemes, 118

S

SEDS algorithm, 260
Self-supervised learning, 9
Semi-supervised learning, 8, 330
Sequential methods, 200
Sigmoid functions, 45
Signal decomposition methods, 331
Simple linear regression, 41
Simplex-based search methodology, 137
Single input single-output (SISO) process, 74
Smart process analytics, 56, 57
SMP algorithm, 257, 259
Space-time grid point, 66
Spatial color patterns, 82
Spectral imaging, 70
Stability lobe diagrams (SLD), 315
State observability, 106
State-space autoregressive exogenous model (SSARX), 58
Static data
  approaches, 41
Statistical models, 247
Stochastic gradient descent (SGD), 76
Substitute conventional controllers, 221
Sum-of squared-error (SSE), 42, 76
Supervised learning model, 2, 3*f*, 4
Support vector machines (SVMs), 45, 148, 268
Support vector regression (SVR), 45
Support vectors, 45
Surrogate modeling, 106
Symbolic regression, 46, 114
System identification, 106
System model, 249

T

Tennessee Eastman process, 91
Tensors, 65, 66
  flow, 75
  representations, 66
Total least squares (TLS), 43
Training CNN procedures, 76
Training process, 77
Trajectory data, 257*f*
Transductive learning, 12
Transition model, 10
Tree-based classifiers, 277

U

Unsupervised machine learning, 5, 284

V

Variable inflation factor (VIF), 58
Variational autoencoders (VAEs), 7

W

WordNet, 23

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY
Copyright Elsevier 2024

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY

Copyright Elsevier 2024

ARTIFICIAL INTELLIGENCE IN MANUFACTURING

Concepts and Methods

Artificial Intelligence in Manufacturing: Concepts and Methods explains the most successful emerging techniques for applying artificial intelligence (AI) to engineering problems.

AI is increasingly being applied to all engineering disciplines, producing more insight into how we understand the world and allowing us to discover and create products in new ways and more efficiently. This book unlocks the advantages of this technology for manufacturing by drawing on work by leading researchers who have successfully developed AI methods that can apply to a range of engineering applications. It also provides detailed technical instructions for the implementation of these methods. Drawing on research in computer science and a range of engineering disciplines, this book tackles the interdisciplinary challenges of the subject to introduce new thinking to important manufacturing problems.

Key Features

- Presents AI concepts from the computer science field using language and examples designed to inspire engineering graduates
- Worked examples are used throughout to help readers fully engage with the methods described
- Readers are supported with definitions of key terms and chapter summaries

About the Editors

Masoud Soroush is a Professor of Chemical and Biological Engineering at Drexel University. His research incorporates manufacturing, nanomaterials, polymers, functional safety, and energy systems. His national awards include the American Institute of Chemical Engineers (AIChE) 2021 Institute Award for Excellence in Industrial Gases Technology, an NSF CAREER award, and the O. Hugo Schuck Best Paper Award of the American Automatic Control Council. He received his B.S. in Chemical Engineering from the Abadan Institute of Technology, Iran, and his M.S. and Ph.D. in Chemical Engineering and M.S. in Electrical Engineering from the University of Michigan, Ann Arbor.

Richard D. Braatz is the Edwin R. Gilliland Professor at the Massachusetts Institute of Technology (MIT) where he does research in advanced manufacturing systems. He received an M.S. and Ph.D. from the California Institute of Technology and was a Professor at the University of Illinois at Urbana-Champaign and a Visiting Scholar at Harvard University before moving to MIT. He was elected to the U.S. National Academy of Engineering in 2019.

ELSEVIER

ACADEMIC PRESS
An imprint of Elsevier
elsevier.com/books-and-journals

ISBN 978-0-323-99134-6
9 780323 991346

This book belongs to Central Library - BADJI Mokhtar-ANNABA UNIVERSITY

Copyright Elsevier 2024