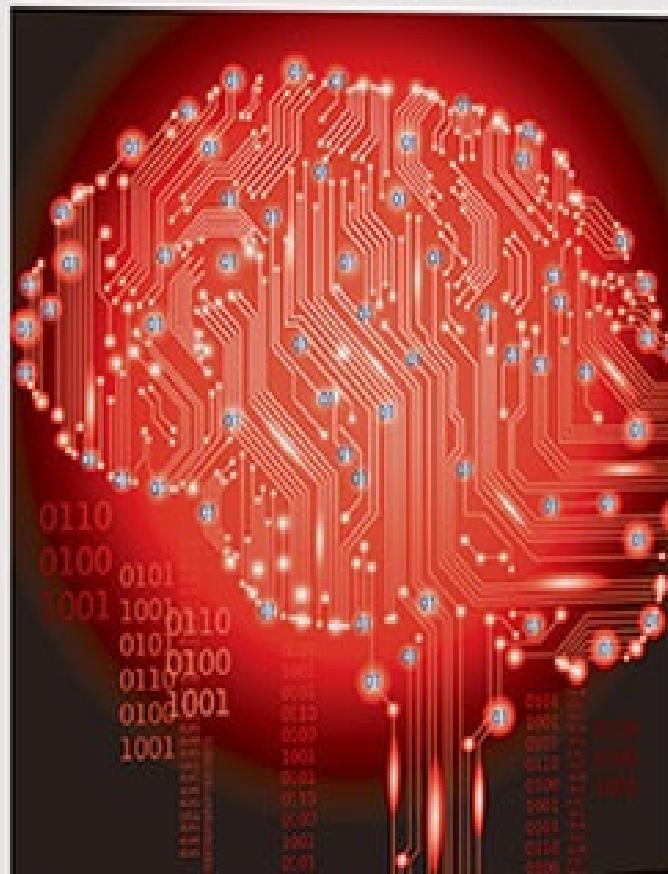


HANDBOOK ON COMPUTATIONAL INTELLIGENCE

Volume 1: Fuzzy Logic, Systems, Artificial Neural Networks, and Learning Systems

Plamen Parvanov Angelov Editor



HANDBOOK ON COMPUTATIONAL INTELLIGENCE

Volume 1: Fuzzy Logic, Systems, Artificial Neural Networks, and Learning Systems

Angelov

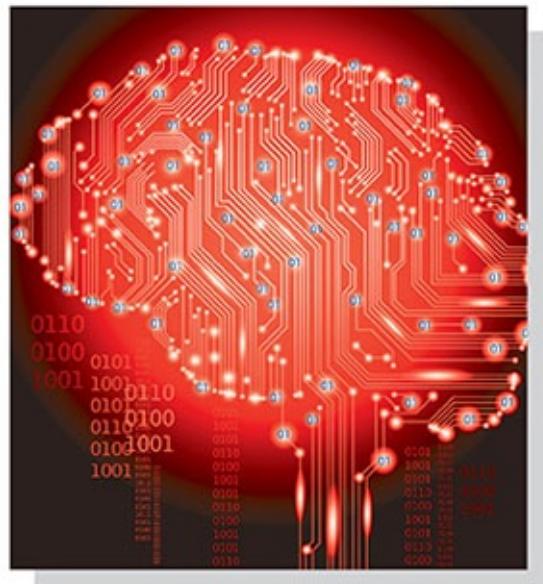


 World Scientific

HANDBOOK ON COMPUTATIONAL INTELLIGENCE

Volume 1

Plamen Parvanov Angelov Editor



HANDBOOK ON COMPUTATIONAL INTELLIGENCE

**Volume 1: Fuzzy Logic, Systems, Artificial
Neural Networks, and Learning Systems**

HANDBOOK ON COMPUTATIONAL INTELLIGENCE

**Volume 1: Fuzzy Logic, Systems, Artificial
Neural Networks, and Learning Systems**

Editor

Plamen Parvanov Angelov

Lancaster University, UK



NEW JERSEY • LONDON • SINGAPORE • BEIJING • SHANGHAI • HONG KONG • TAIPEI • CHENNAI • TOKYO

Published by

World Scientific Publishing Co. Pte. Ltd.

5 Toh Tuck Link, Singapore 596224

USA office: 27 Warren Street, Suite 401-402, Hackensack, NJ 07601

UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library.

HANDBOOK ON COMPUTATIONAL INTELLIGENCE

In 2 Volumes

Volume 1: Fuzzy Logic, Systems, Artificial Neural Networks, and Learning Systems

Volume 2: Evolutionary Computation, Hybrid Systems, and Applications

Copyright © 2016 by World Scientific Publishing Co. Pte. Ltd.

All rights reserved. This book, or parts thereof, may not be reproduced in any form or by any means, electronic or mechanical, including photocopying, recording or any information storage and retrieval system now known or to be invented, without written permission from the publisher

For photocopying of material in this volume, please pay a copying fee through the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, USA. In this case permission to photocopy is not required from the publisher.

ISBN 978-981-4675-00-0 (Set)

ISBN 978-981-4675-03-1 (Vol. 1)

ISBN 978-981-4675-04-8 (Vol. 2)

In-house Editors: Dipasri Sardar/Amanda Yun

Typeset by Stallion Press

Email: enquiries@stallionpress.com

Printed in Singapore

Dedication

Computational Intelligence is only possible if we have a *Computer*. This work is dedicated to the inventor of the digital computer, John Vincent Atanasoff.

Contents

Introduction by the Editor

About the Editor

Acknowledgments

Prologue

Volume 1: Fuzzy Logic, Systems, Artificial Neural Networks, and Learning Systems

Part I: Fuzzy Logic and Systems

1. Fundamentals of Fuzzy Set Theory

Fernando Gomide

2. Granular Computing

Andrzej Bargiela and Witold Pedrycz

3. Evolving Fuzzy Systems — Fundamentals, Reliability, Interpretability, Usability, Applications

Edwin Lughofer

4. Modeling Fuzzy Rule-Based Systems

Rashmi Dutta Baruah and Diganta Baruah

5. Fuzzy Classifiers

Abdelhamid Bouchachia

6. Fuzzy Model-Based Control — Predictive and Adaptive Approaches

Igor Škrjanc and Sašo Blažič

7. Fuzzy Fault Detection and Diagnosis

Bruno Sielly Jales Costa

Part II: Artificial Neural Networks and Learning Systems

8. The ANN and Learning Systems in Brains and Machines

Leonid Perlovsky

9. Introduction to Cognitive Systems

Péter Érdi and Mihály Bányai

10. A New View on Economics with Recurrent Neural Networks

Hans-Georg Zimmermann, Ralph Grothmann and Christoph Tietz

11. Evolving Connectionist Systems for Adaptive Learning and Pattern Recognition: From

Neuro-Fuzzy-, to Spiking- and Neurogenetic

Nikola Kasabov

12. Reinforcement Learning with Applications in Automation Decision and Feedback Control

Kyriakos G. Vamvoudakis, Frank L. Lewis and Draguna Vrabie

13. Kernel Models and Support Vector Machines

Denis Kolev, Mikhail Suvorov and Dmitry Kangin

Volume 2: Evolutionary Computation, Hybrid Systems, and Applications

Part III: Evolutionary Computation

14. History and Philosophy of Evolutionary Computation

Carlos A. Coello Coello, Carlos Segura and Gara Miranda

15. A Survey of Recent Works in Artificial Immune Systems

Guilherme Costa Silva and Dipankar Dasgupta

16. Swarm Intelligence: An Introduction, History and Applications

Fevrier Valdez

17. Memetic Algorithms

Qiangfu Zhao and Yong Liu

Part IV: Hybrid Systems

18. Multi-Objective Evolutionary Design of Fuzzy Rule-Based Systems

Michela Antonelli, Pietro Ducange and Francesco Marcelloni

19. Bio-Inspired Optimization of Interval Type-2 Fuzzy Controllers

Oscar Castillo

20. Nature-Inspired Optimization of Fuzzy Controllers and Fuzzy Models

Radu-Emil Precup and Radu-Codruț David

21. Genetic Optimization of Modular Neural Networks for Pattern Recognition with a Granular Approach

Patricia Melin

22. Hybrid Evolutionary-, Constructive- and Evolving Fuzzy Neural Networks

Michael J. Watts and Nikola Kasabov

Part V: Applications

23. Applications of Computational Intelligence to Decision-Making: Modeling Human Reasoning/Agreement

Simon Miller, Christian Wagner and Jonathan Garibaldi

24. Applications of Computational Intelligence to Process Industry

Jose Macias Hernández

25. Applications of Computational Intelligence to Robotics and Autonomous Systems

Adham Atyabi and Samia Nefti-Meziani

26. Selected Automotive Applications of Computational Intelligence

Mahmoud Abou-Nasr, Fazal Syed and Dimitar Filev

Index

Introduction by the Editor

The term *Computational Intelligence* was coined more recently (at the end of the last century when a series of high profile conferences were organized by the Institute of Electrical and Electronics Engineers (IEEE) leading to the formation of the Computational Intelligence Society within the IEEE), however, the disciplines and problems it covers have been in existence for a much longer period of time. The very idea of developing systems, devices, algorithms, techniques that possess characteristics of “*intelligence*” and are computational (not just conceptual) dates back to the middle of the 20th century or even earlier and is broadly associated with the so-called “*artificial intelligence*”. However, “*artificial intelligence*” is nowadays rather linked with logic, cognition, natural language processing, induction and so on, while “*computational intelligence*” has been developed in a direction that can be described as “*nature-inspired*” alternatives to the conventional/traditional computing approaches. This includes, but is not limited to:

- Fuzzy logic (as a more human-oriented approach to reasoning);
- Artificial neural networks (mimicking the human brain);
- Evolutionary algorithms (mimicking the population-based genetic evolution), and
- Dynamically evolving systems based on the above.

Some authors also attribute other areas of research such as belief-based Dempster–Shafer theory, chaos theory, swarm and collective intelligence, etc. on the margins of *Computational Intelligence*. It is also often the case that the application areas such as pattern recognition, image processing, business and video analytics and so on are also attributed or linked closely to *Computational Intelligence*; areas of research that are closer to Statistical Learning (e.g., Support Vector Machines), probability theory, Bayesian, Markov models etc. are also sometimes considered to be a part of *Computational Intelligence*.

In this handbook, while not closing the door to all possible methods and alternatives, we keep clear the picture of *Computational Intelligence* as a distinct area of research that is based on the above mentioned pillars and we assume that the other areas are either applications of *Computational Intelligence* methods, techniques or approaches or research areas that gravitate around Statistical Learning.

The primary goal of the area of *Computational Intelligence* is to provide efficient computational solutions to the existing open problems from theoretical and application points of view in understanding, representation, modeling, visualization, reasoning, decision, prediction, classification, analysis and control of physical objects, environmental or social processes and phenomena to which the traditional methods, techniques and

theories (primarily, so-called “first principles”, deterministic, often expressed as differential equations and stemming from mass-and energy-balance) cannot provide a valid or useful/practical solution.

Another specific feature of *Computational Intelligence* is that it offers solutions that bear characteristics of “intelligence” which is usually attributed to humans only. This has to be considered broadly rather than literally as in the area of “artificial intelligence”. For example, fuzzy logic systems can make decisions like humans do. This is in a stark contrast with the deterministic type expert systems as well as with the probabilistic associative rules. For artificial neural networks, one can argue that they process the data in a manner which is more like what human Brain does. In evolutionary computation, the population of candidate solutions “evolves” towards the optimum in a manner similar to the way species and living organisms evolve in Nature. Finally, in dynamically evolving systems, self-development is very much like in the real life, where we, humans, learn individually from experience in a supervised (from parents, teachers, peers etc.) or unsupervised (self-learning) manner. Learning from experience, we can develop a rule-base starting from “scratch” and constantly update this rule-base by adding new rules or removing the outdated ones; or similarly, the strengths of the links between neurons in our brain can dynamically evolve and new areas of the brain can be activated or deactivated (thus, dynamically evolving neural networks).

In this handbook, a mix of leading academics and practitioners in this area as well as some of the younger generation researchers contributed to cover the main aspects of the exciting area of *Computational Intelligence*. The overall structure and invitations to a much broader circle of contributors were set up initially. After a careful peer review process, a selected set of contributions was organized in a coherent end product aiming to cover the main areas of *Computational Intelligence*; it also aims to cover both, the theoretical basis and, at the same time, to give a flavor of the possible efficient applications.

This handbook is composed of two volumes and five parts which contain 26 chapters.

[Volume 1](#) includes [Part I](#) (Fuzzy Logic) and [Part II](#) (Artificial Neural Networks and Learning Systems).

[Volume 2](#) includes [Part III](#) (Evolutionary Computation), [Part IV](#) (Hybrid Systems) and [Part V](#) (Applications).

In [Part I](#), the readers can find seven chapters, including:

- Fundamentals of Fuzzy Set Theory

This chapter sets the tone with a thorough, step by step introduction to the theory of fuzzy sets. It is written by one of the foremost experts in this area, Dr. Fernando Gomide, Professor at University of Campinas, Campinas, Brazil.

- Granular Computing

Granular computing became a cornerstone of *Computational Intelligence* and the chapter offers a thorough review of the problems and solutions. It is written by two of the leading experts in this area, Dr. Andrzej Bargiela, Professor at Nottingham University, UK (now in Christchurch University, New Zealand) and Dr. Witold Pedrycz, Professor at University of Alberta, Canada.

- Evolving Fuzzy Systems — Fundamentals, Reliability, Interpretability, Usability, Applications

Since its introduction around the turn of the centuries by the Editor, the area of evolving fuzzy systems is constantly developing and this chapter offers a review of the problems and some of the solutions. It is written by Dr. Edwin Lughofer, Key Researcher at Johannes Kepler University, Linz, Austria, who quickly became one of the leading experts in this area following an exchange of research visits with Lancaster University, UK.

- Modeling of Fuzzy Rule-based Systems

This chapter covers the important problem of designing fuzzy systems from data. It is written by Dr. Rashmi Dutta Baruah of Indian Institute of Technology, Guwahati, India and Diganta Baruah of Sikkim Manipal Institute of Technology, Sikkim, India. Rashmi recently obtained a PhD degree from Lancaster University, UK in the area of evolving fuzzy systems.

- Fuzzy Classifiers

This chapter covers the very important problem of fuzzy rule-based classifiers and is written by the expert in the field, Dr. Hamid Bouchachia, Associate Professor at Bournemouth University, UK.

- Fuzzy Model based Control: Predictive and Adaptive Approach

This chapter covers the problems of fuzzy control and is written by the experts in the area, Dr. Igor Škrjanc, Professor and Dr. Sašo Blažič, Professor at the University of Ljubljana, Slovenia.

- Fuzzy Fault Detection and Diagnosis

This chapter is written by Dr. Bruno Costa, Professor at IFRN, Natal, Brazil who specialized recently in Lancaster, UK.

Part II consists of six chapters, which cover:

- The ANN and Learning Systems in Brains and Machines

This chapter is written by Dr. Leonid Perlovsky from Harvard University, USA.

- Introduction to Cognitive Systems

This chapter is written by Dr. Peter Erdi, Professor at Kalamazoo College, USA, co-authored by Dr. Mihaly Banyai (leading author) from Wigner RCP, Hungarian Academy of Sciences.

- A New View on Economics with Recurrent Neural Networks

This chapter offers a rather specific view on the recurrent neural networks from the point of view of their importance for modeling economic processes and is written by a team of industry-based researchers from Siemens, Germany including Drs. Hans Georg Zimmermann, Ralph Grothmann and Christoph Tietz.

- Evolving Connectionist Systems for Adaptive Learning and Pattern Recognition: From Neuro-Fuzzy, to Spiking and Neurogenetic

This chapter offers a review of one of the cornerstones of *Computational Intelligence*, namely, the evolving connectionist systems, and is written by the pioneer in this area Dr. Nikola Kasabov, Professor at Auckland University of Technology, New Zealand.

- Reinforcement Learning with Applications in Automation Decision and Feedback Control

This chapter offers a thorough and advanced analysis of the reinforcement learning from the perspective of decision-making and control. It is written by one of the world's leading experts in this area, Dr. Frank L. Lewis, Professor at The University of Texas, co-authored by Dr. Kyriakos Vamvoudakis from the same University (the leading author) and Dr. Draguna Vrabie from the United Technologies Research Centre, USA.

- Kernel Models and Support Vector Machines

This chapter offers a very skilful review of one of the hottest topics in research and applications linked to classification and related problems. It is written by a team of young Russian researchers who are finishing their PhD studies at Lancaster University, UK (Denis Kolev and Dmitry Kangin) and by Mikhail Suvorov. All three graduated from leading Moscow Universities (Moscow State University and Bauman Moscow State Technical University).

Part III consists of four chapters:

- History and Philosophy of the Evolutionary Computation

This chapter lays the basis for one of the pillars of *Computational Intelligence*, covering its history and basic principles. It is written by one of the well-known experts in this area, Dr. Carlos A. Coello-Coello from CINVESTAV, Mexico and co-authored by Dr. Carlos Segura from the Centre of Research in Mathematics, Mexico and Dr. Gara Miranda from the University of La Laguna, Tenerife, Spain.

- A Survey of Recent Works in Artificial Immune Systems

This chapter covers one of the important aspects of *Computational Intelligence* which is

associated with the Evolutionary Computation. It is written by the pioneer in the area Dr. Dipankar Dasgupta, Professor at The University of Memphis, USA and is co-authored by Dr. Guilherme Costa Silva from the same University who is the leading author.

- **Swarm Intelligence: An Introduction, History and Applications**

This chapter covers another important aspect of Evolutionary Computation and is written by Dr. Fevrier Valdez from The Institute of Technology, Tijuana, Mexico.

- **Memetic Algorithms**

This chapter reviews another important type of methods and algorithms which are associated with the Evolutionary Computation and is written by a team of authors from the University of Aizu, Japan led by Dr. Qiangfu Zhao who is a well-known expert in the area of *Computational Intelligence*. The team also includes Drs. Yong Liu and Yan Pei.

Part IV consists of five chapters:

- **Multi-objective Evolutionary Design of Fuzzy Rule-Based Systems**

This chapter covers one of the areas of hybridization where Evolutionary Computation is used as an optimization tool for automatic design of fuzzy rule-based systems from data. It is written by the well-known expert in this area, Dr. Francesco Marcelloni, Professor at the University of Pisa, Italy, supported by Dr. Michaela Antonelli and Dr. Pietro Ducange from the same University.

- **Bio-inspired Optimization of Type-2 Fuzzy Controllers**

The chapter offers a hybrid system where a fuzzy controller of the so-called type-2 is being optimized using a bio-inspired approach. It is written by one of the leading experts in type-2 fuzzy systems, Dr. Oscar Castillo, Professor at The Institute of Technology, Tijuana Mexico.

- **Nature-inspired Optimization of Fuzzy Controllers and Fuzzy Models This**

chapter also offers a hybrid system in which fuzzy models and controllers are being optimized using nature-inspired optimization methods. It is written by the well-known expert in the area of fuzzy control, Dr. Radu-Emil Precup, Professor at The Polytechnic University of Timisoara, Romania and co-authored by Dr. Radu Codrut David.

- **Genetic Optimization of Modular Neural Networks for Pattern Recognition with a Granular Approach**

This chapter describes a hybrid system whereby modular neural networks using a granular approach are optimized by a genetic algorithm and applied for pattern recognition. It is written by Dr. Patricia Melin, Professor at The Institute of Technology, Tijuana, Mexico who is well-known through her work in the area of hybrid systems.

- Hybrid Evolutionary-, Constructive-, and Evolving Fuzzy Neural Networks

This is another chapter by the pioneer of evolving neural networks, Professor Dr. Nikola Kasabov, co-authored by Dr. Michael Watts (leading author), both from Auckland, New Zealand.

Part V includes four chapters:

- Applications of Computational Intelligence to Decision-Making: Modeling Human Reasoning/Agreement

This chapter covers the use of *Computational Intelligence* in decision-making applications, in particular, modeling human reasoning and agreement. It is authored by the leading expert in this field, Dr. Jonathan Garibaldi, Professor at The Nottingham University, UK and co-authored by Drs. Simon Miller (leading author) and Christian Wagner from the same University.

- Applications of Computational Intelligence to Process Industry

This chapter offers the industry-based researcher's point of view. Dr. Jose Juan Macias Hernandez is leading a busy Department of Process Control at the largest oil refinery on the Canary Islands, Spain and is also Associate Professor at the local University of La Laguna, Tenerife.

- Applications of Computational Intelligence to Robotics and Autonomous Systems

This chapter describes applications of *Computational Intelligence* to the area of Robotics and Autonomous Systems and is written by Dr. Adham Atyabi and Professor Dr. Samia Nefti-Meziani, both from Salford University, UK.

- Selected Automotive Applications of Computational Intelligence

Last, but not least, the chapter by the pioneer of fuzzy systems area Dr. Dimitar Filev, co-authored by his colleagues, Dr. Mahmoud Abou-Nasr (leading author) and Dr. Fazal Sayed (all based at Ford Motors Co., Dearborn, MI, USA) offers the industry-based leaders' point of view.

In conclusion, this Handbook is composed with care aiming to cover all main aspects of *Computational Intelligence* area of research offering solid background knowledge as well as end-point applications from the leaders in the area supported by younger researchers in the field. It is designed to be a one-stop-shop for interested readers, but by no means aims to completely replace all other sources in this dynamically evolving area of research.

Enjoy reading it.

Plamen Angelov

Lancaster, UK

About the Editor



Professor Plamen Angelov (www.lancs.ac.uk/staff/angelov) holds a Personal Chair in Intelligent Systems and leads the Data Science Group at Lancaster University, UK. He has PhD (1993) and Doctor of Sciences (DSc, 2015) degrees and is a Fellow of both the IEEE and IET, as well as a Senior Member of the International Neural Networks Society (INNS). He is also a member of the Boards of Governors of both bodies for the period 2014–2017. He also chairs the Technical Committee (TC) on Evolving Intelligent Systems within the Systems, Man and Cybernetics Society, IEEE and is a member of the TCs on Neural Networks and on Fuzzy Systems within the Computational Intelligence Society, IEEE. He has authored or co-authored over 200 peer-reviewed publications in leading journals, peer-reviewed conference proceedings, five patents and a dozen books, including two research monographs by Springer (2002) and Wiley (2012), respectively. He has an active research portfolio in the area of data science, computational intelligence and autonomous machine learning and internationally recognized results into online and evolving learning and algorithms for knowledge extraction in the form of human-intelligible fuzzy rule-based systems. Prof. Angelov leads numerous projects funded by UK research councils, EU, industry, UK Ministry of Defence, The Royal Society, etc. His research was recognized by ‘The Engineer Innovation and Technology 2008 Special Award’ and ‘For outstanding Services’ (2013) by IEEE and INNS. In 2014, he was awarded a Chair of Excellence at Carlos III University, Spain sponsored by Santander Bank. Prof. Angelov is the founding Editor-in-Chief of Springer’s journal on *Evolving Systems* and Associate Editor of the leading international scientific journals in this area, including *IEEE Transactions on Cybernetics*, *IEEE Transactions on Fuzzy Systems* and half a dozen others. He was General, Program or Technical co-Chair of prime IEEE conferences (IJCNN-2013, Dallas; SSCI2014, Orlando, WCCI2014, Beijing; IS’14, Warsaw; IJCNN2015, Killarney; IJCNN/ WCCI2016, Vancouver; UKCI 2016, Lancaster; WCCI2018, Rio de Janeiro) and founding General co-Chair of a series of annual IEEE conferences on Evolving and Adaptive Intelligent Systems. He was a Visiting Professor in Brazil, Germany, Spain, France, Bulgaria. Prof. Angelov regularly gives invited and plenary talks at leading conferences, universities and companies.

Acknowledgments

The Editor would like to acknowledge the support of the Chair of Excellence programme of Carlos III University, Madrid, Spain.

The Editor would also like to acknowledge the unwavering support of his family (Rosi, Lachezar and Mariela).

Prologue

The *Handbook on Computational Intelligence* aims to be a one-stop-shop for the various aspects of the broad research area of Computational Intelligence. The *Handbook* is organized into five parts over two volumes:

- (1) Fuzzy Sets and Systems (Vol. 1)
- (2) Artificial Neural Networks and Learning Systems (Vol. 1)
- (3) Evolutionary Computation (Vol. 2)
- (4) Hybrid Systems (Vol. 2)
- (5) Applications (Vol. 2)

In total, 26 chapters detail various aspects of the theory, methodology and applications of *Computational Intelligence*. The authors of the different chapters are leading researchers in their respective fields or “rising stars” (promising early career researchers). This mix of experience and energy provides an invaluable source of information that is easy to read, rich in detail, and wide in spectrum. In total, over 50 authors from 16 different countries including USA, UK, Japan, Germany, Canada, Italy, Spain, Austria, Bulgaria, Brazil, Russia, India, New Zealand, Hungary, Slovenia, Mexico, and Romania contributed to this collaborative effort. The scope of the *Handbook* covers practically all important aspects of the topic of Computational Intelligence and has several chapters dedicated to particular applications written by leading industry-based or industry-linked researchers.

Preparing, compiling and editing this Handbook was an enjoyable and inspirational experience. I hope you will also enjoy reading it and will find answers to your questions and will use this book in your everyday work.

Plamen Angelov
Lancaster, UK

Volume 1

Part I

Fuzzy Logic and Systems

Chapter 1

Fundamentals of Fuzzy Set Theory

Fernando Gomide

The goal of this chapter is to offer a comprehensive, systematic, updated, and self-contained tutorial-like introduction to fuzzy set theory. The notions and concepts addressed here cover the spectrum that contains, we believe, the material deemed relevant for computational intelligence and intelligent systems theory and applications. It starts by reviewing the very basic idea of sets, introduces the notion of a fuzzy set, and gives the main insights and interpretations to help intuition. It proceeds with characterization of fuzzy sets, operations and their generalizations, and ends discussing the issue of information granulation and its key constituents.

1.1. Sets

A set is a fundamental concept in mathematics and science. Classically, a set is defined as “any multiplicity which can be thought of as one and any totality of definite elements which can be bound up into a whole by means of a law” or being more descriptive “any collection into a whole M of definite and separate objects m of our intuition or our thought” (Cantor, 1883, 1895).

Intuitively, a set may be viewed as the class M of all objects m satisfying any particular property or defining condition. Alternatively, a set can be characterized by an assignment scheme to define the objects of a domain that satisfy the intended property. For instance, an indicator function or a characteristic function is a function defined on a domain X that indicates membership of an object of X in a set A on X , having the value 1 for all elements of A and the value 0 for all elements of X not in A . The domain can be either continuous or discrete. For instance, the closed interval $[3, 7]$ constitutes a continuous and bounded domain whereas the set $N = \{0, 1, 2, \dots\}$ of natural numbers is discrete and countable, but with no bound.

In general, a characteristic function of a set A defined in X assumes the following form

$$A(x) = \begin{cases} 1, & \text{if } x \in A \\ 0, & \text{if } x \notin A \end{cases}. \quad (1)$$

The empty set \emptyset has a characteristic function that is identically equal to zero, $\emptyset(x) = 0$ for all x in X . The domain X itself has a characteristic function that is identically equal to one, $X(x) = 1$ for all x in X . Also, a singleton $A = \{a\}$, a set with only a single element, has the characteristic function $A(x) = 1$ if $x = a$ and $A(x) = 0$ otherwise.

Characteristic functions $A : X \rightarrow \{0, 1\}$ induce a constraint with well-defined boundaries on the elements of the domain X that can be assigned to a set A .

1.2. Fuzzy Sets

The fundamental idea of a fuzzy set is to relax the rigid boundaries of the constraints induced by characteristic functions by admitting intermediate values of class membership. The idea is to allow assignments of intermediate values between 0 and 1 to quantify our perception on how compatible the objects of a domain are with the class, with 0 meaning incompatibility, complete exclusion, and 1 compatibility, complete membership. Membership values thus express the degrees to which each object of a domain is compatible with the properties distinctive to the class. Intermediate membership values mean that no natural threshold exists and that elements of a universe can be a member of a class and at the same time belong to other classes with different degrees. Gradual, less strict membership degrees is the essence of fuzzy sets.

Formally, a fuzzy set A is described by a membership function mapping the elements of a domain X to the unit interval $[0, 1]$ (Zadeh, 1965)

$$A : X \rightarrow [0, 1]. \quad (2)$$

Membership functions fully define fuzzy sets. Membership functions generalize characteristic functions in the same way as fuzzy sets generalize sets. Fuzzy sets can be also be seen as a set of ordered pairs of the form $\{x, A(x)\}$ where x is an object of X and $A(x)$ is its corresponding degree of membership. For a finite domain $X = \{x_1, x_2, \dots, x_n\}$, A can be represented by an n -dimensional vector $A = (a_1, a_2, \dots, a_n)$ with each component $a_i = A(x_i)$.

Being more illustrative, we may view fuzzy sets as elastic constraints imposed on the elements of a universe. Fuzzy sets deal primarily with the concepts of elasticity, graduality, or absence of sharply defined boundaries. In contrast, sets are concerned with rigid boundaries, lack of graded belongingness, and sharp binary constraints. Gradual membership means that no natural boundary exists and that some elements of the domain can, contrary to sets, coexist (belong) to different fuzzy sets with different degrees of membership. For instance, in [Figure 1.1](#), $x_1 = 1.5$ is compatible with the concept of *short* and $x_2 = 2.0$ belongs to the category of *tall* people, when assuming the model of sets, but x_1 simultaneously is 0.8 *short* and 0.2 *tall* and x_2 simultaneously is 0.2 *short* and 0.8 *tall* under the perspective of fuzzy sets.

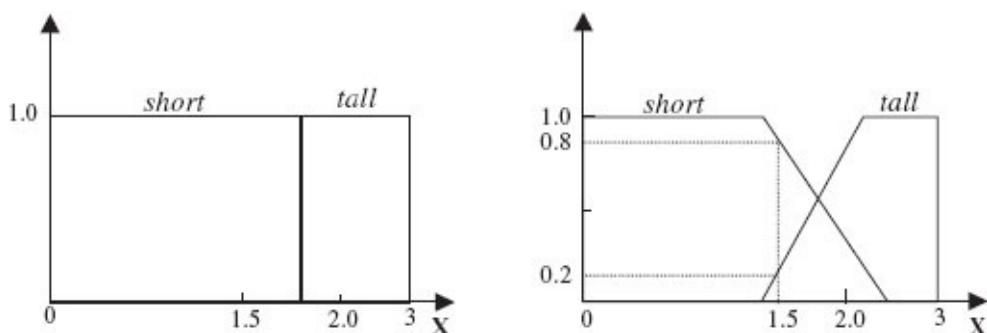


Figure 1.1: Two-valued membership in characteristic functions (sets) and gradual membership represented by membership functions (fuzzy sets).

1.2.1. Interpretation of Fuzzy Sets

In fuzzy set theory, fuzziness has a precise meaning. Fuzziness primarily means lack of precise boundaries of a collection of objects and, as such, it is a manifestation of imprecision and a particular type of uncertainty.

First, it is worth noting that fuzziness is both conceptually and formally different from the fundamental concept of probability. In general, it is difficult to foresee the result of tossing a fair coin as it is impossible to know if either head or tail will occur for certain. We may, at most, say that there is a 50% chance to have a head or tail, but as soon as the coin falls, uncertainty vanishes. On the contrary, when we say that a person is tall we are not being precise, and imprecision remains independently of any event. Formally, probability is a set function, a mapping whose universe is a set of subsets of a domain. In contrast, fuzzy sets are membership functions, mappings from some given universe of discourse to the unit interval.

Secondly, fuzziness, generality, and ambiguity are distinct notions. A notion is general when it applies to a multiplicity of objects and keeps only a common essential property. An ambiguous notion stands for several unrelated objects. Therefore, from this point of view, fuzziness does not mean neither generality nor ambiguity and applications of fuzzy sets exclude these categories. Fuzzy set theory assumes that the universe is well defined and has its elements assigned to classes by means of a numerical scale.

Applications of fuzzy set in areas such as data analysis, reasoning under uncertainty, and decision-making suggest different interpretations of membership grades in terms of similarity, uncertainty, and preference (Dubois and Prade, 1997, 1998). Membership value $A(x)$ from the point of view of similarity means the degree of compatibility of an element $x \in X$ with representative elements of A . This is the primary and most intuitive interpretation of a fuzzy set, one that is particularly suitable for data analysis. An example is the case when we question on how to qualify an environment as *comfortable* when we know that current temperature is 25°C. Such quantification is a matter of degree. For instance, assuming a domain $X = [0, 40]$ and choosing 20°C as representative of *comfortable* temperature, we note, in [Figure 1.2](#), that 25°C is comfortable to the degree of 0.2. In the example, we have adopted piecewise linearly decreasing functions of the distance between temperature values and the representative value 20°C to determine the corresponding membership degree.

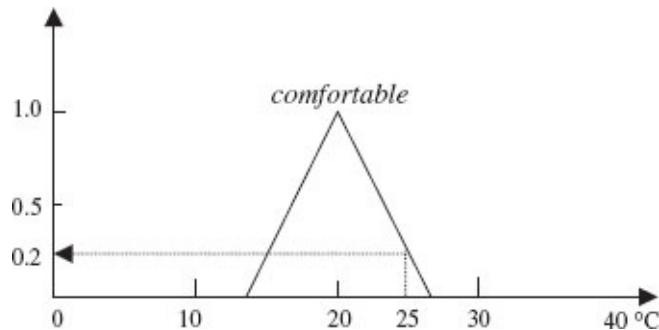


Figure 1.2: Membership function for a fuzzy set of *comfortable* temperature.

Now, assume that the values of a variable x is such that $A(x) > 0$. Then given a value v of X , $A(v)$ expresses a possibility that $x = v$ given that x is in A is all that is known. In this situation, the membership degree of a given tentative value v to the class A reflects the degree of plausibility that this value is the same as x . This idea reflects a type of uncertainty because if the membership degree is high, our confidence about the value of x may still be low, but if the degree is low, then the tentative value may be rejected as an implausible candidate. The variable labeled by the class A is uncontrollable. This allows assignment of fuzzy sets to possibility distributions as suggested in possibility theory (Zadeh, 1978). For instance, suppose someone said he felt comfortable in an environment. In this situation the membership degree of a given tentative temperature value, say 25°C , reflects the degree of plausibility that this value of temperature is the same as the one under which the individual felt comfortable. Note that the actual value of the temperature value is unknown, but there is no question if that value of temperature did occur or not. Possibility concerns whether an event may occur and with what degree. On the contrary, probability concerns whether an event will occur.

Finally, assume that A reflects a preference on the values of a variable x in X . For instance, x can be a decision variable and fuzzy set A , a flexible constraint characterizing feasible values and decision-maker preferences. In this case $A(v)$ denotes the grade of preference in favor of v as the value of x . This interpretation prevails in fuzzy optimization and decision analysis. For instance, we may be interested in finding a comfortable value of temperature. The membership degree of a candidate temperature value v reflects our degree of satisfaction with the particular temperature value chosen. In this situation, the choice of the value is controllable in the sense that the value being adopted depends on our choice.

1.2.2. *Rationale for Membership Functions*

Generally speaking, any function $A : X \rightarrow [0, 1]$ is qualified to serve as a membership function describing the corresponding fuzzy set. In practice, the form of the membership functions should reflect the environment of the problem at hand for which we construct fuzzy sets. They should mirror our perception of the concept to be modeled and used in problem solving, the level of detail we intend to capture, and the context in which the fuzzy set are going to be used. It is essential to assess the type of fuzzy set from the

standpoint of its suitability when handling the design and optimization issues. Given these reasons in mind, we review the most commonly used categories of membership functions. All of them are defined in the universe of real numbers, that is $X = \mathbf{R}$.

1.2.2.1. Triangular membership function

It is described by piecewise linear segments of the form

$$A(x, a, m, b) = \begin{cases} 0 & \text{if } x \leq a \\ \frac{x-a}{m-a} & \text{if } x \in [a, m] \\ \frac{b-x}{b-m} & \text{if } x \in [m, b] \\ 0 & \text{if } x \geq b \end{cases}.$$

Using more concise notation, the above expression can be written down in the form $A(x, a, m, b) = \max\{\min[(x - a)/(m - a), (b - x)/(b - m)], 0\}$, as in [Figure 1.3](#). The meaning of the parameters is straightforward: m is the modal (typical) value of the fuzzy set while a and b are the lower and upper bounds, respectively. They could be sought as those elements of the domain that delineate the elements belonging to A with non-zero membership degrees.

Triangular fuzzy sets (membership functions) are the simplest possible models of grades of membership as they are fully defined by only three parameters. The semantics of triangular fuzzy sets reflects the knowledge of the typical value of the concept and its spread. The linear change in the membership grades is the simplest possible model of membership one could think of. If the derivative of the triangular membership function could be sought as a measure of sensitivity of A , then its sensitivity is constant for each of the linear segments of the fuzzy set.

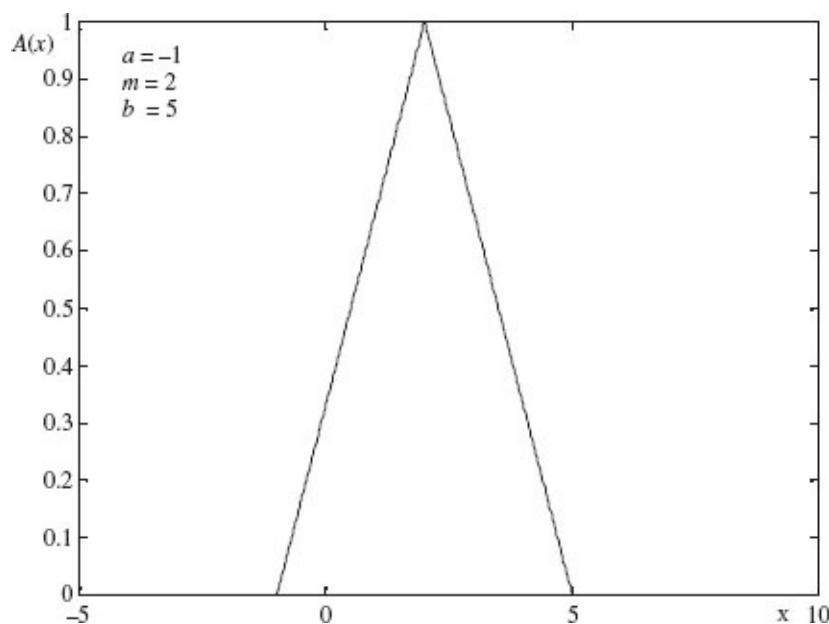


Figure 1.3: Triangular membership function.

1.2.2.2. Trapezoidal membership function

A piecewise linear function characterized by four parameters, a , m , n , and b each of which defines one of the four linear parts of the membership function, as in [Figure 1.4](#). It has the following form

$$A(x) = \begin{cases} 0 & \text{if } x < a \\ \frac{x-a}{m-a} & \text{if } x \in [a, m] \\ 1 & \text{if } x \in [m, n] \\ \frac{b-x}{b-n} & \text{if } x \in [n, b] \\ 0 & \text{if } x > b \end{cases}$$

We can rewrite A using an equivalent notation as follows

$$A(x, a, m, n, b) = \max\{\min[(x - a)/(m - a), 1, (b - x)/(b - n)], 0\}.$$

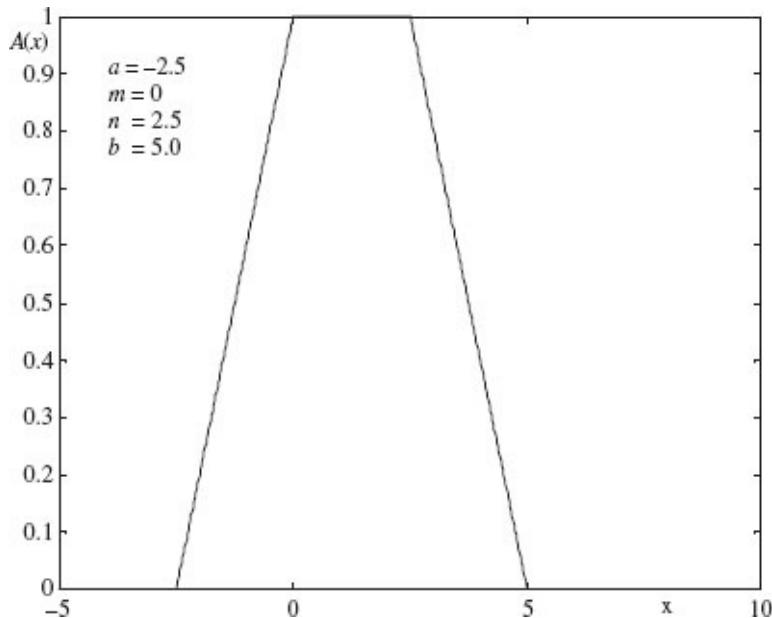


Figure 1.4: Trapezoidal membership function.

1.2.2.3. Γ -membership function

This function has the form

$$A(x) = \begin{cases} 0 & \text{if } x \leq a \\ 1 - e^{-k(x-a)^2} & \text{if } x > a \end{cases} \quad \text{or} \quad A(x) = \begin{cases} 0 & \text{if } x \leq a \\ \frac{k(x-a)^2}{1+k(x-a)^2} & \text{if } x > a \end{cases},$$

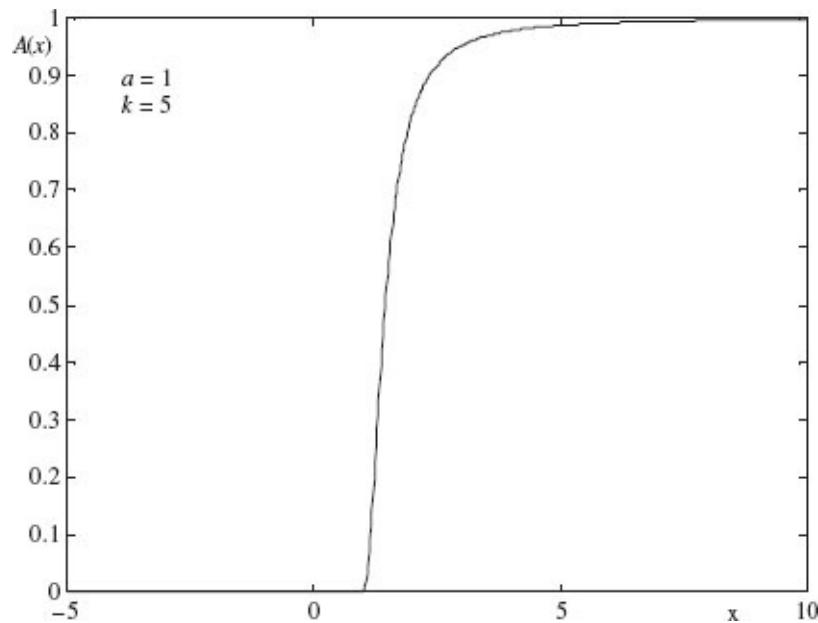
where $k > 0$, as in [Figure 1.5](#).

1.2.2.4. S-membership function

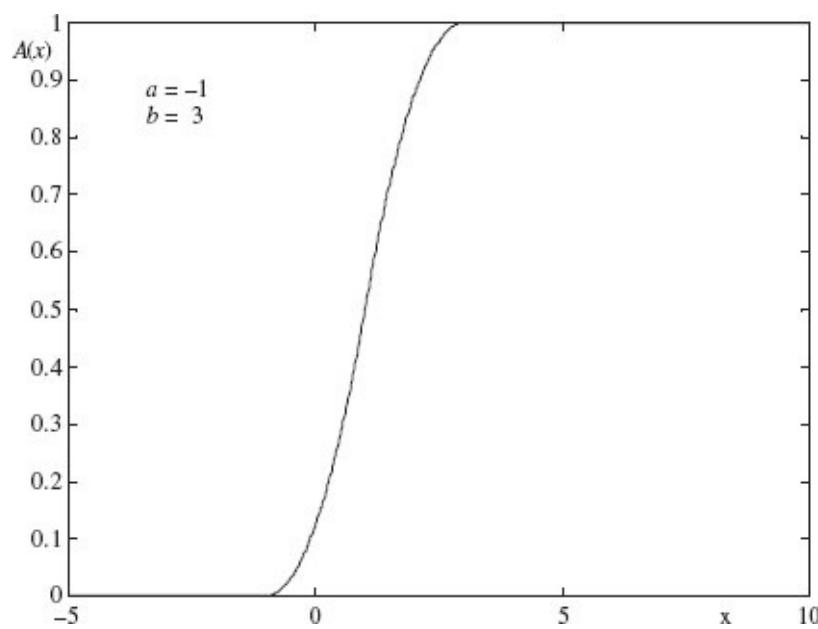
The function is expressed by

$$A(x) = \begin{cases} 0 & \text{if } x \leq a \\ 2 \left(\frac{x-a}{b-a} \right)^2 & \text{if } x \in [a, m] \\ 1 - 2 \left(\frac{x-b}{b-a} \right)^2 & \text{if } x \in [m, b] \\ 1 & \text{if } x > b \end{cases}.$$

The point $m = (a + b)/2$ is the crossover point of the S-function, as in [Figure 1.6](#).



[Figure 1.5:](#) Γ -membership function.



[Figure 1.6:](#) S-membership function.

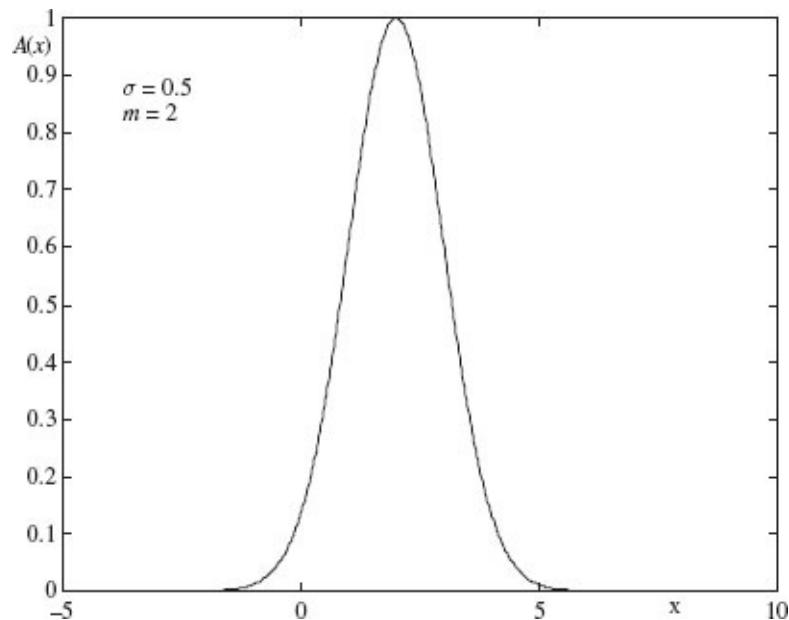


Figure 1.7: Gaussian membership function.

1.2.2.5. Gaussian membership function

This membership function is described by the following relationship

$$A(x, m, \sigma) = \exp\left(-\frac{(x - m)^2}{\sigma^2}\right).$$

An example of the membership function is shown in [Figure 1.7](#). Gaussian membership functions have two important parameters. The modal value m represents the typical element of A while σ denotes a spread of A . Higher values of σ corresponds to larger spreads of the fuzzy sets.

1.2.2.6. Exponential-like membership function

It has the following form, as in [Figure 1.8](#).

$$A(x) = \frac{1}{1 + k(x - m)^2} \quad k > 0.$$

The spread of the exponential-like membership function increases as the value of k gets lower.

1.3. Characteristics of Fuzzy Sets

Given the diversity of potentially useful and semantically sound membership functions, there are certain common characteristics or descriptors that are conceptually and operationally useful to capture the essence of fuzzy sets. We provide next a list of the descriptors commonly encountered in practice.

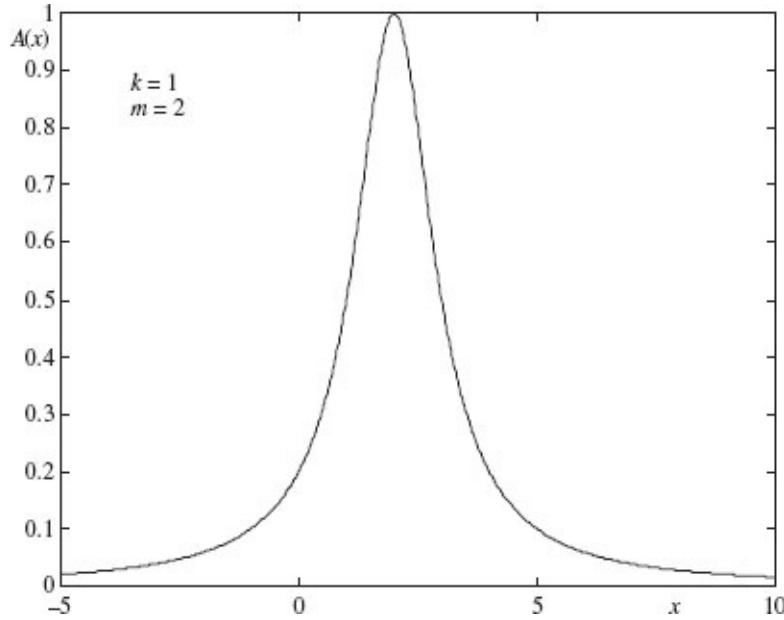


Figure 1.8: Example of the exponential-like membership function.

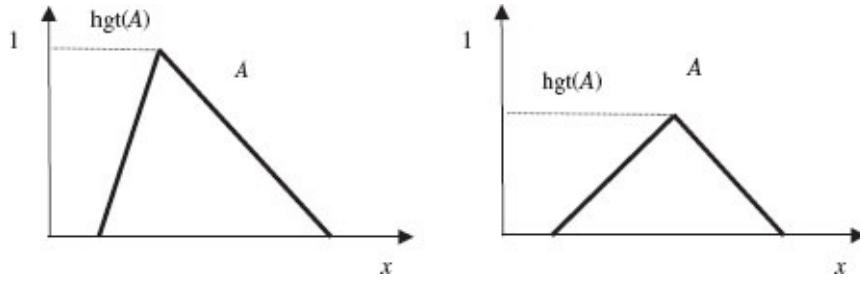


Figure 1.9: Examples of normal and subnormal fuzzy sets.

1.3.1. Normality

We say that the fuzzy set A is *normal* if its membership function attains 1, that is,

$$\sup_{x \in X} A(x) = 1. \quad (3)$$

If this property does not hold, we call the fuzzy set *subnormal*. An illustration of the corresponding fuzzy set is shown in Figure 1.9. The supremum (\sup) in the above expression is also referred to as a height of the fuzzy set A . Thus, the fuzzy set is normal if $hgt(A) = 1$. The normality of A has a simple interpretation: by determining the height of the fuzzy set, we identify an element of the domain whose membership degree is the highest. The value of the height being equal to 1 states that there is at least one element in X fully typical with respect to A and which could be sought as entirely compatible with the semantic category presented by A . A subnormal fuzzy set has height lower than 1, viz.

$hgt(A) < 1$, and the degree of typicality of elements in this fuzzy set is somewhat lower (weaker) and we cannot identify any element in X which is fully compatible with the underlying concept. In practice, while forming a fuzzy set we expect its normality.

1.3.2. Normalization

The normalization, denoted by $Norm(\cdot)$, is a mechanism to convert a subnormal non-empty fuzzy set A into its normal counterpart. This is done by dividing the original membership function by its height

$$Norm(A) = \frac{A(x)}{hgt(A)}. \quad (4)$$

While the height describes the global property of the membership grades, the following notions offer an interesting characterization of the elements of X regarding their membership degrees.

1.3.3. Support

Support of a fuzzy set A , $Supp(A)$, is a set of all elements of X with non-zero membership degrees in A

$$Supp(A) = \{x \in X \mid A(x) > 0\}. \quad (5)$$

In other words, support identifies all elements of X that exhibit some association with the fuzzy set under consideration (by being allocated to A with non-zero membership degrees).

1.3.4. Core

The core of a fuzzy set A , $Core(A)$, is a set of all elements of the universe that are typical to A : they come with unit membership grades.

$$Core(A) = \{x \in X \mid A(x) = 1\}. \quad (6)$$

The support and core are related in the sense that they identify and collect elements belonging to the fuzzy set, yet at two different levels of membership. Given the character of the core and support, we note that all elements of the core of A are subsumed by the elements of the support of this fuzzy set. Note that both support and core, are sets, not fuzzy sets. In [Figure 1.10](#), they are intervals. We refer to them as the set-based characterizations of fuzzy sets.

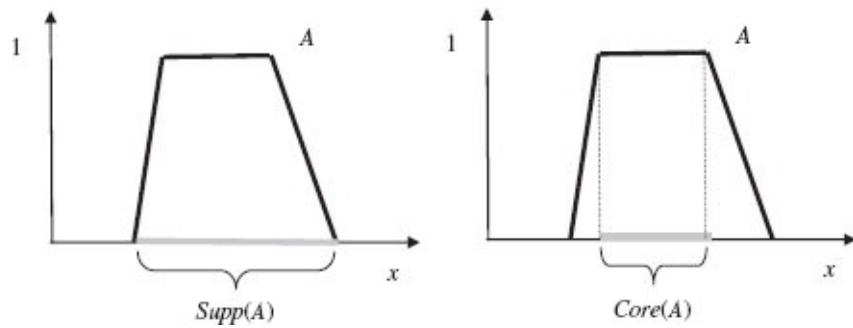


Figure 1.10: Support and core of A .

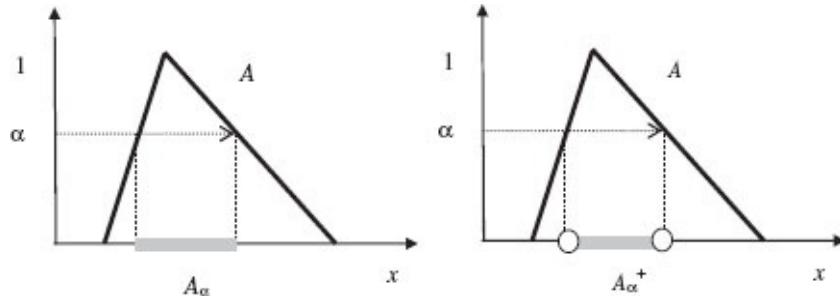


Figure 1.11: Example of α -cut and of strong α -cut.

While core and support are somewhat extreme, in the sense that they identify the elements of A that exhibit the strongest and the weakest links with A , we may be also interested in characterizing sets of elements that come with some intermediate membership degrees. The notion of **α -cut** offers here an interesting insight into the nature of fuzzy sets.

1.3.5. **α -Cut**

The **α -cut** of a fuzzy set A , denoted by A_α , is a set consisting of the elements of the domain whose membership values are equal to or exceed a certain threshold level $\alpha \in [0, 1]$. Formally $A_\alpha = \{x \in X \mid A(x) \geq \alpha\}$. A strong **α -cut** identifies all elements in X for which $A_\alpha^+ = \{x \in X \mid A(x) > \alpha\}$. Figure 1.11 illustrates the notion of **α -cut** and strong **α -cut**. Both support and core are limit cases of **α -cuts** and strong **α -cuts**. From $\alpha = 0$ and the strong **α -cut**, we arrive at the concept of the support of A . The value $\alpha = 1$ means that the corresponding **α -cut** is the core of A .

1.3.6. **Representation Theorem**

Any fuzzy set can be viewed as a family of fuzzy sets. This is the essence of a result known as the representation theorem. The representation theorem states that any fuzzy set A can be decomposed into a family of **α -cuts**,

$$A = \bigcup_{\alpha \in [0,1]} \alpha A_\alpha$$

or, equivalently in terms of membership functions,

$$A(x) = \sup_{\alpha \in [0,1]} \alpha A_\alpha(x).$$

1.3.7. Convexity

We say that a fuzzy set is convex if its membership function satisfies the following condition:

$$A[\lambda x_1 + (1 - \lambda)x_2] \geq \min[A(x_1), A(x_2)], \quad \forall x_1, x_2 \in X, \lambda \in [0, 1]. \quad (7)$$

Relationship (7) says that, whenever we choose a point x on a line segment between x_1 and x_2 , the point $(x, A(x))$ is always located above or on the line passing through the two points $(x_1, A(x_1))$ and $(x_2, A(x_2))$, as in [Figure 1.12](#). Note that the membership function is not a convex function in the conventional sense (Klir and Yuan, 1995).

The set S is convex if, for all $x_1, x_2 \in S$, then $x = \lambda x_1 + (1 - \lambda)x_2 \in S$ for all $\lambda \in [0, 1]$. Convexity means that any line segment identified by any two points in S is contained in S . For instance, intervals of real numbers are convex sets. Therefore, if a fuzzy set is convex, then all of its **α -cuts** are convex, and conversely, if a fuzzy set has all its **α -cuts** convex, then it is a convex fuzzy set, as in [Figure 1.13](#). Thus we may say that a fuzzy set is convex if all its **α -cuts** are convex.

Fuzzy sets can be characterized by counting their elements and using a single numeric quantity as a descriptor of the count. While in case of sets this sounds clear, with fuzzy sets we have to consider different membership grades. In the simplest form this counting comes under the name of cardinality.

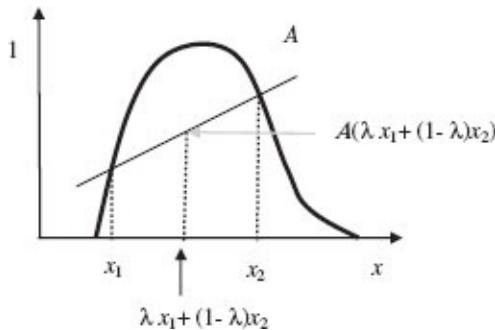


Figure 1.12: Convex fuzzy set A .

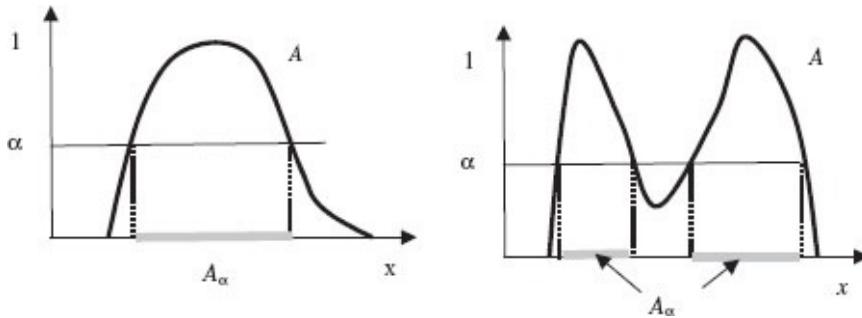


Figure 1.13: Convex and non-convex fuzzy sets.

1.3.8. Cardinality

Given a fuzzy set A defined in a finite or countable universe X , its cardinality, denoted by $\text{Card}(A)$ is expressed as the following sum

$$\text{Card}(A) = \sum_{x \in X} A(x) \quad (8)$$

or, alternatively, as the integral

$$\text{Card}(A) = \int_X A(x) dx, \quad (9)$$

assuming that the integral is well-defined. We also use the alternative notation $\text{Card}(A) = |A|$ and refer to it as a sigma count (σ -count).

The cardinality of fuzzy sets is explicitly associated with the concept of granularity of information granules realized in this manner. More descriptively, the more the elements of A we encounter, the higher the level of abstraction supported by A and the lower the granularity of the construct. Higher values of cardinality come with the higher level of abstraction (generalization) and the lower values of granularity (specificity).

So far we discussed properties of a single fuzzy set. Next we look at the characterizations of relationships between two fuzzy sets.

1.3.9. Equality

We say that two fuzzy sets A and B defined in X are equal if and only if their membership functions are identical, that is

$$A(x) = B(x) \quad \forall x \in X. \quad (10)$$

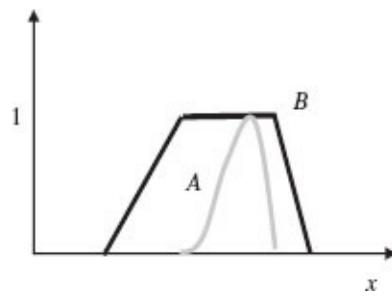


Figure 1.14: Inclusion $A \subseteq B$.

1.3.10. Inclusion

Fuzzy set A is a subset of B (A is included in B), $A \subseteq B$, if and only if every element of A also is an element of B . This property expressed in terms of membership degrees means that the following inequality is satisfied

$$A(x) \leq B(x) \quad \forall x \in X. \quad (11)$$

An illustration of these two relationships in the case of sets is shown in [Figure 1.14](#). To satisfy the relationship of inclusion, we require that the characteristic functions adhere to Equation (11) for all elements of X . If the inclusion is not satisfied even for a single point of X , the inclusion property does not hold. See (Pedrycz and Gomide, 2007) for alternative notion of inclusion that captures the idea of degree of inclusion.

1.3.11. Specificity

Often, we face the issue to quantify how much a single element of a domain could be viewed as a representative of a fuzzy set. If this fuzzy set is a singleton, then

$$A(x) = \begin{cases} 1 & \text{if } x = x_0 \\ 0 & \text{if } x \neq x_0 \end{cases}, \quad (12)$$

and there is no hesitation in selecting x_0 as the sole representative of A . We say that A is very *specific* and its choice comes with no hesitation. On the other extreme, if A covers the entire domain X and has all elements with the membership grade equal to 1, the choice of the only one representative of A becomes more problematic once it is not clear which element to choose. These two extreme situations are shown in [Figure 1.15](#). Intuitively, we see that the specificity is a concept that relates with the cardinality of a set. The higher the cardinality of the set (the more evident its abstraction) is, the lower its specificity.

One approach to quantify the notion of specificity of a fuzzy set is as follows (Yager, 1983). The specificity of a fuzzy set A defined in X , denoted by $Spec(A)$, is a mapping from a family of normal fuzzy sets in X into nonnegative numbers such that the following conditions are satisfied, as in [Figure 1.16](#).

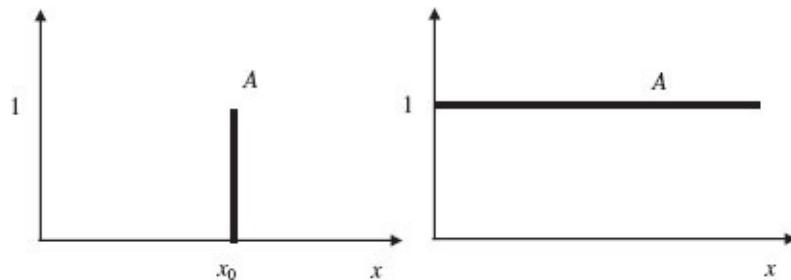


Figure 1.15: Two extreme cases of sets with distinct levels of specificity.

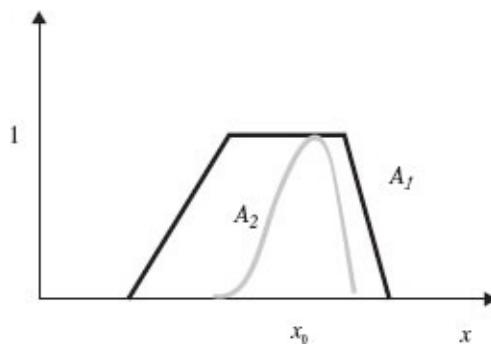


Figure 1.16: Specificity of fuzzy sets: Fuzzy set A_1 is less specific than A_2 .

1. $Spec(A) = 1$ if and only if there exists only one element x_0 of X for which $A(x_0) = 1$ and

$$A(x) = 0 \quad \forall x \neq x_o;$$

2. $Spec(A) = 0$ if and only if $A(x) = 0 \quad \forall x \in X$;
3. $Spec(A_1) \leq Spec(A_2)$ if $A_1 \supset A_2$.

A particular instance of specificity measure is (Yager, 1983)

$$Spec(A) = \int_0^{\alpha_{\max}} \frac{1}{Card(A_\alpha)} d\alpha,$$

where $\alpha_{\max} = hgt(A)$. For finite domains, the integration is replaced by the sum

$$Spec(A) = \sum_{i=1}^m \frac{1}{Card(A_{\alpha_i})} \Delta \alpha_i,$$

where $\Delta \alpha_i = \alpha_i - \alpha_{i-1}$ with $\alpha_0 = 0$; m stands for the number of the membership grades of A .

1.4. Operations with Fuzzy Sets

As in set theory, we may combine fuzzy sets to produce new fuzzy sets. Generally, combination must possess properties to match intuition, to comply with the semantics of the intended operation, and to be flexible to fit application requirements. Next we provide an overview of the main operations and their generalizations, interpretations, and examples of realizations.

1.4.1. Standard Operations on Sets and Fuzzy Sets

To start, we review the familiar operations of intersection, union and complement of set theory. Consider two sets $A = \{x \in \mathbf{R} | 1 \leq x \leq 3\}$ and $B = \{x \in \mathbf{R} | 2 \leq x \leq 4\}$, closed intervals of the real line. Their intersection is the set $A \cap B = \{x \in \mathbf{R} | 2 \leq x \leq 3\}$. [Figure 1.17](#) shows the intersection operation in terms of the characteristic functions of A and B . Looking at the values of the characteristic function of $A \cap B$ that results when comparing the individual values of $A(x)$ and $B(x)$ for each $x \in \mathbf{R}$, we note that they correspond to the minimum between the values of $A(x)$ and $B(x)$.

In general, given the characteristic functions of A and B , the characteristic function of their intersection $A \cap B$ is computed using

$$(A \cap B)(x) = \min[A(x), B(x)] \quad \forall x \in X, \quad (13)$$

where $(A \cap B)(x)$ denotes the characteristic function of the set $A \cap B$.

The union of sets A and B in terms of the characteristic functions proceeds similarly. If A and B are the same intervals as above, then $A \cup B = \{x \in \mathbf{R} | 1 \leq x \leq 4\}$. In this case the value of the characteristic function of the union is the maximum of corresponding values of the characteristic functions $A(x)$ and $B(x)$ taken point wise, as in [Figure 1.18](#).

Therefore, given the characteristic functions of A and B , we determine the characteristic function of the union as

$$(A \cup B)(x) = \max[A(x), B(x)] \quad \forall x \in X, \quad (14)$$

where $(A \cup B)(x)$ denotes the characteristic function of the set $A \cup B$.

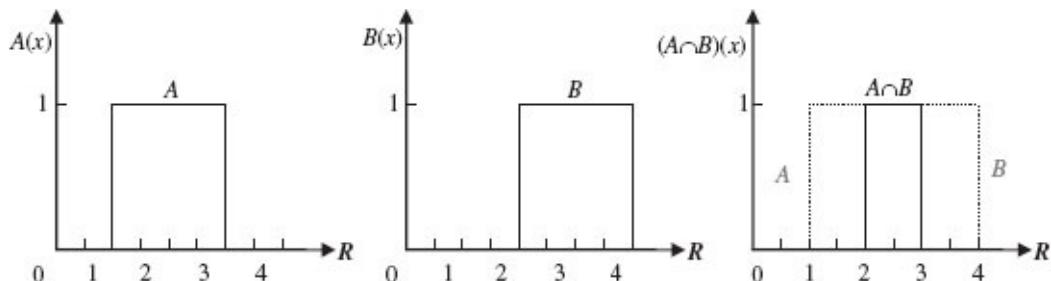


Figure 1.17: Intersection of sets in terms of their characteristic functions.

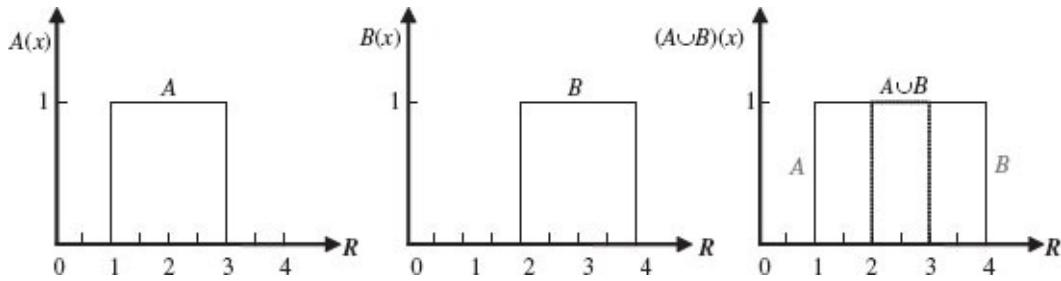


Figure 1.18: Union of two sets in terms of their characteristic functions.

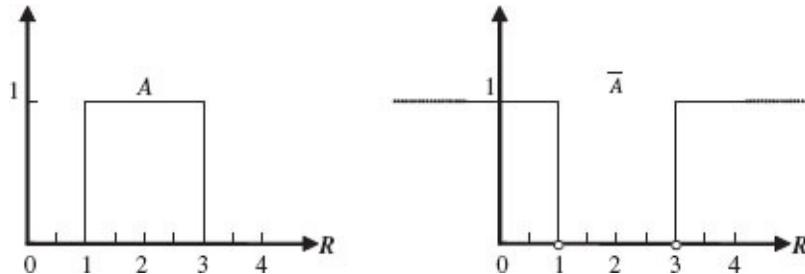


Figure 1.19: Complement of a set in terms of its characteristic function.

Likewise, as Figure 1.19 suggests, the complement \bar{A} of A , expressed in terms of its characteristic function, is the one-complement of the characteristic function of A . For instance, if $A = \{x \in \mathbf{R} | 1 \leq x \leq 3\}$, then $\bar{A} = \{x \in \mathbf{R} | 4 < x < 1\}$.

Thus, the characteristic function of the complement of a set A is

$$\bar{A}(x) = 1 - A(x), \quad \forall x \in X. \quad (15)$$

Because sets are particular instances of fuzzy sets, the operations of intersection, union and complement should equally well apply to fuzzy sets. Indeed, when we use membership functions in expressions (13)–(15), these formulae serve as standard definitions of intersection, union, and complement of fuzzy sets. Examples are shown in Figure 1.20. Standard set and fuzzy set operations fulfill the properties of Table 1.1.

Figures 1.19 and 1.20 show that the laws of non-contradiction and excluded middle hold for sets, but not for fuzzy sets under the standard operations. See also Table 1.2. Particularly worth noting is a violation of the non-contradiction law once it shows the issue of fuzziness from the point of view of the coexistence of a class and its complement, one of the main source of fuzziness. This coexistence is impossible in set theory and means a contradiction in conventional logic. Interestingly, if we consider a particular subnormal fuzzy set A whose membership function is constant and equal to 0.5 for all elements of the universe, then from Equations (13)–(15) we see that $A = A \cup \bar{A} = A \cap \bar{A} = \bar{A}$, a situation in which there is no way to distinguish the fuzzy set from its complement and any fuzzy set that results from standard operations with them. The value 0.5 is a crossover point representing a balance between membership and non-membership at which we attain the highest level of fuzziness. The fuzzy set and its complement are indiscernible.

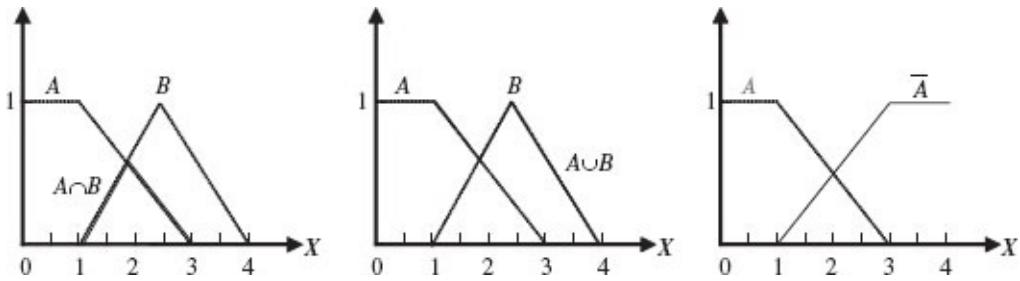


Figure 1.20: Standard operations on fuzzy sets.

Table 1.1: Properties of operations with sets and fuzzy sets.

(1) Commutativity	$A \cup B = B \cup A$ $A \cap B = B \cap A$
(2) Associativity	$A \cup (B \cup C) = (A \cup B) \cup C$ $A \cap (B \cap C) = (A \cap B) \cap C$
(3) Distributivity	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
(4) Idempotency	$A \cup A = A$ $A \cap A = A$
(5) Boundary conditions	$A \cup \phi = A$ and $A \cup X = X$ $A \cap \phi = \phi$ and $A \cap X = A$
(6) Involution	$\bar{\bar{A}} = A$
(7) Transitivity	if $A \subset B$ and $B \subset C$, then $A \subset C$

Table 1.2: Non-contradiction and excluded middle for standard operations.

	Sets	Fuzzy sets
8-Non-contradiction	$A \cap \bar{A} = \phi$	$A \cap \bar{A} \neq \phi$
9-Excluded middle	$A \cup \bar{A} = X$	$A \cup \bar{A} \neq X$

As we will see next, there are types of operations for which non-contradiction and excluded middle are recovered. While for sets these types produce the same result as the standard operators, this is not the case with fuzzy sets. Also, $A = A \cup \bar{A} = A \cap \bar{A} = \bar{A}$ does not hold for any choice of intersection, union and complement operators.

Operations on fuzzy sets concern manipulation of their membership functions. Therefore, they are domain dependent and different contexts may require their different realizations. For instance, since operations provide ways to combine information, they can be performed differently in image processing, control, and diagnostic systems applications for example. When developing realizations of intersection and union of fuzzy sets it is useful to require commutativity, associativity, monotonicity, and identity. The last requirement (identity) has different forms depending on the operation. For instance, the intersection of any fuzzy set with domain X should return the fuzzy set itself. For the union, identity implies that the union of any fuzzy set and an empty fuzzy set returns the fuzzy set itself. Thus, in principle, any two place operator $[0, 1] \times [0, 1] \rightarrow [0, 1]$ which satisfies the collection of the requirements can be regarded as a potential candidate to

realize the intersection or union of fuzzy sets, identity acting as boundary conditions meaning. In general, idempotency is not strictly required, however the realizations of union and intersection could be idempotent as this happens with the minimum and maximum operators ($\min[a, a] = a$ and $\max[a, a] = a$).

1.4.2. Triangular Norms and Conorms

Triangular norms and conorms constitute general forms of operations for intersection and union. While t-norms generalize intersection of fuzzy sets, t-conorms (or s-norms) generalize the union of fuzzy sets (Klement *et al.*, 2000).

A triangular norm is a two place operation $t: [0, 1] \times [0, 1] \rightarrow [0, 1]$ that satisfies the following properties

1. Commutativity: $a \, t \, b = b \, t \, a$,
2. Associativity: $a \, t \, (b \, t \, c) = (a \, t \, b) \, t \, c$,
3. Monotonicity: if $b \leq c$ then $a \, t \, b \leq a \, t \, c$,
4. Boundary conditions: $a \, t \, 1 = a$ $a \, t \, 0 = 0$,

where $a, b, c \in [0, 1]$.

There is a one-to-one correspondence between the general requirements outlined above and the properties of t-norms. The first three reflect the general character of set operations. Boundary conditions stress the fact that all t-norms attain the same values at boundaries of the unit square $[0, 1] \times [0, 1]$. Thus, for sets, any t-norm produces the same result.

Examples of t-norms are

$$\text{Minimum: } a \, t_m \, b = \min(a, b) = a \wedge b,$$

$$\text{Product: } a \, t_p \, b = a \, b,$$

$$\text{Lukasiewicz: } a \, t_l \, b = \max(a + b - 1, 0),$$

$$\text{Drastic product: } a \, t_d \, b = \begin{cases} a & \text{if } b = 1 \\ b & \text{if } a = 1 \\ 0 & \text{otherwise} \end{cases}.$$

The minimum(t_m), product(t_p), Lukasiewicz(t_l), and drastic product(t_d) operators are shown in [Figure 1.21](#), with examples of the union of the triangular fuzzy sets on $X = [0, 8]$, $A = (x, 1, 3, 6)$ and $B = (x, 2.5, 5, 7)$.

Triangular conorms are functions $s: [0, 1] \times [0, 1] \rightarrow [0, 1]$ that serve as generic realizations of the union operator on fuzzy sets.

One can show that $s: [0, 1] \times [0, 1] \rightarrow [0, 1]$ is a t-conorm if and only if there exists a t-norm, called dual t-norm, such that for $\forall a, b \in [0, 1]$ we have

$$a \text{ s } b = 1 - (1 - a)t(1 - b). \quad (16)$$

For the corresponding dual t-norm, we have

$$a \text{ t } b = 1 - (1 - a)s(1 - b). \quad (17)$$

The duality expressed by Equations (16) and (17) can be viewed as alternative definition of t-conorms. Duality allows us to deduce the properties of t-conorms on the basis of the analogous properties of t-norms. From Equations (16) and (17), we get

$$(1 - a)t(1 - b) = 1 - a \text{ s } b$$

and

$$(1 - a)s(1 - b) = 1 - a \text{ t } b.$$

These two relationships can be expressed symbolically as

$$\begin{aligned} \bar{A} \cap \bar{B} &= \overline{A \cup B}, \\ \bar{A} \cup \bar{B} &= \overline{A \cap B}, \end{aligned}$$

which are the De Morgan laws. Commonly used t-conorms include

Maximum: $as_m b = \max(a, b) = a \vee b$,

Algebraic sum: $as_p b = a + b - a b$,

Lukasiewicz: $as_l b = \min(a + b, 1)$,

Drastic sum: $as_d b = \begin{cases} a & \text{if } b = 0 \\ b & \text{if } a = 0 \\ 1 & \text{otherwise} \end{cases}$

The maximum(s_m), algebraic sum(s_p), Lukasiewicz(s_l), and drastic sum(s_d) operators are shown in [Figure 1.22](#), which also include the union of the triangular fuzzy sets on $[0, 8]$, $A = (x, 1, 3, 6)$ and $B = (x, 2.5, 5, 7)$.

The properties $A \cup \bar{A} = X$ and the excluded middle hold for the drastic sum.

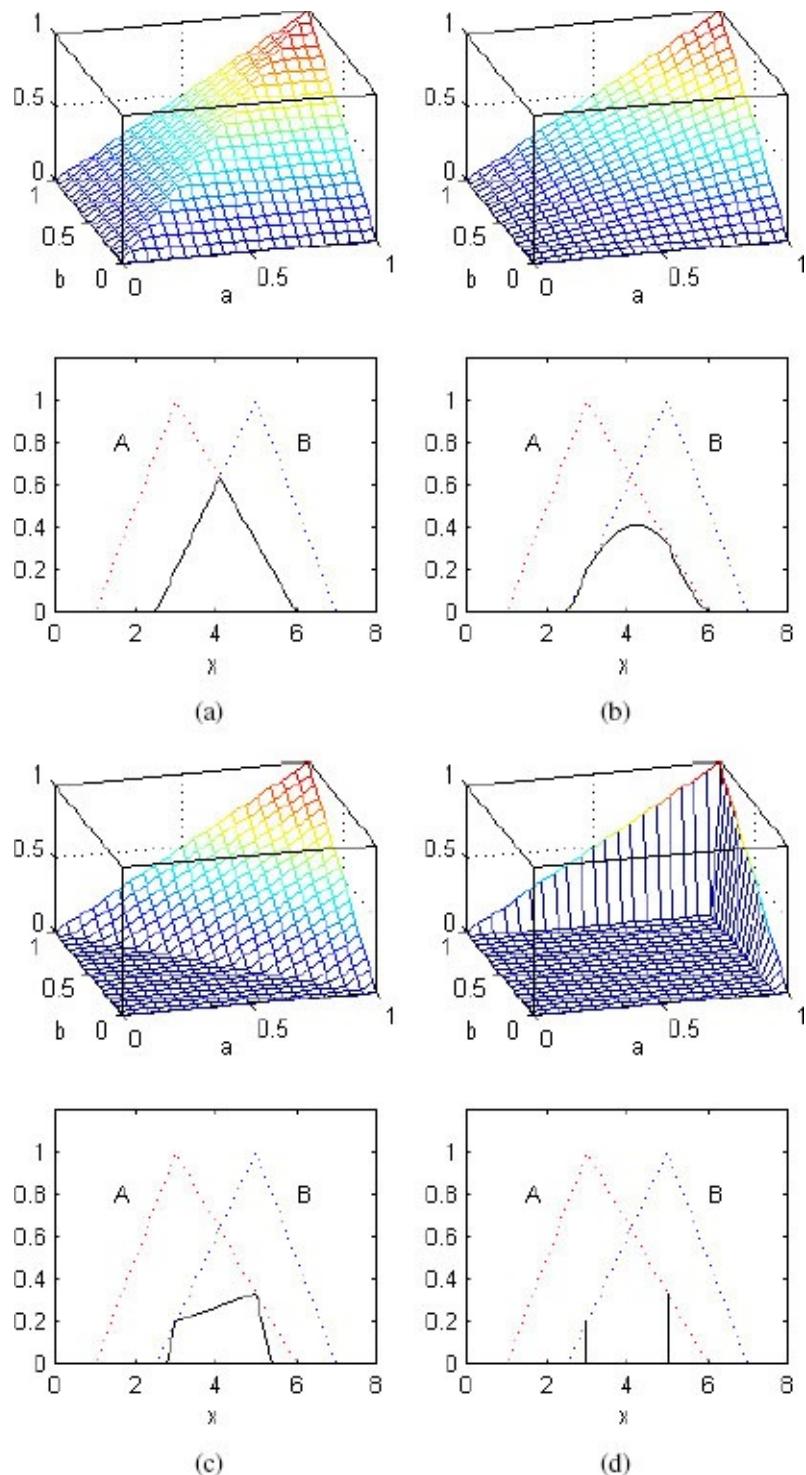


Figure 1.21: The (a) minimum, (b) product, (c) Lukasiewicz, (d) drastic product t-norms and the intersection of fuzzy sets A and B .

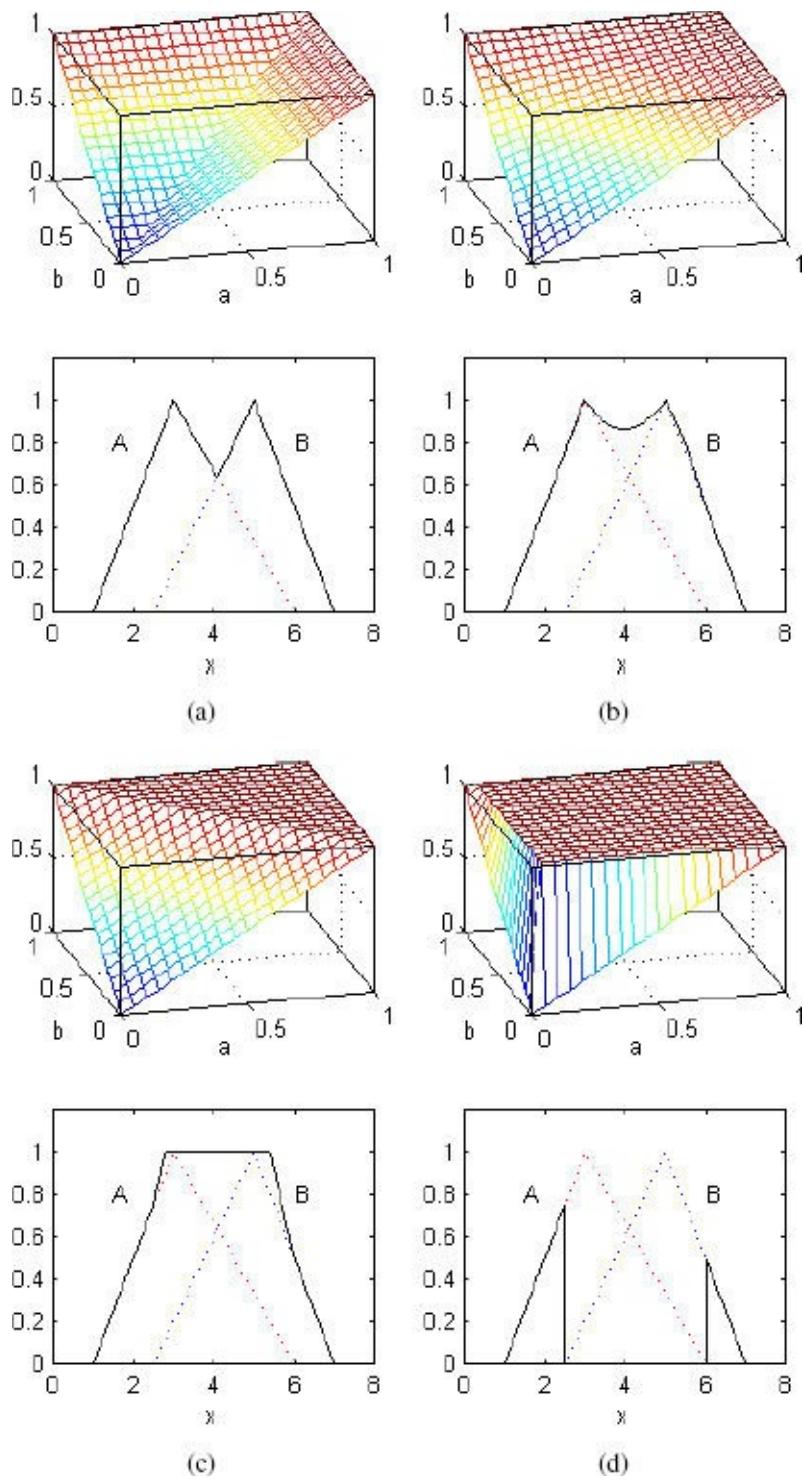


Figure 1.22: The (a) maximum, (b) algebraic sum, (c) Lukasiewicz, (d) drastic sum s-norms and the union of fuzzy sets A and B .

1.4.3. Triangular Norms as Categories of Logical Operators

Fuzzy propositions involve combination of linguistic statements (or their symbolic representations) such as in

- (1) temperature is *high* and humidity is *low*,
- (2) velocity is *low* or noise level is *high*.

These sentences use logical operations \wedge (*and*), \vee (*or*) to combine linguistic statements into propositions. For instance, in the first example we have a conjunction (*and*, \wedge) of

linguistic statements while in the second there is a disjunction (*or*, \vee) of the statements. Given the truth values of each statement, the question is how to determine the truth value of the composite statement or, equivalently, the truth value of the proposition.

Let truth (P) = $p \in [0, 1]$, the truth value of proposition P . Thus, $p = 0$ means that the proposition is false, while $p = 1$ means that P is true. Intermediate values $p \in (0, 1)$ indicate partial truth of the proposition. To compute the truth value of composite propositions coming in the form of $P \wedge Q$, $P \vee Q$ given the truth values p and q of its components, we have to come up with operations that transforms truth values p and q into the corresponding truth values $p \wedge q$ and $p \vee q$. To make these operations meaningful, we require that they satisfy some basic requirements. For instance, it is desirable that $p \wedge q$ and $q \wedge p$ (similarly, $p \vee q$ and $q \vee p$) produce the same truth values. Likewise, we require that the truth value of $(p \wedge q) \wedge r$ is the same as the following combination $p \wedge (q \wedge r)$. In other words, the conjunction and disjunction operations are commutative and associative. Also, when the truth value of an individual statement increases, the truth values of their combinations also increase. Moreover, if P is absolutely false, $p = 0$, then $P \wedge Q$ should also be false no matter what the truth value of Q is. Furthermore, the truth value of $P \vee Q$ should coincide with the truth value of Q . On the other hand, if P is absolutely true, $p = 1$, then the truth value of $P \wedge Q$ should coincide with the truth value of Q , while $P \vee Q$ should also be true. Triangular norms and conorms are the general families of logic connectives that comply with these requirements. Triangular norms provide a general category of logical connectives in the sense that t-norms are used to model conjunction operators while t-conorms serve as models of disjunctions.

Let $L = \{P, Q, \dots\}$ be a set of single (atomic) statements P, Q, \dots , and truth: $L \rightarrow [0, 1]$ a function which assigns truth values $p, q, \dots \in [0, 1]$ to each element of L . Thus, we have:
 $\text{truth}(P \text{ and } Q) \equiv \text{truth}(P \wedge Q) \rightarrow p \wedge q = p \text{ t } q$,
 $\text{truth}(P \text{ or } Q) \equiv \text{truth}(P \vee Q) \rightarrow p \vee q = p \text{ s } q$.

[Table 1.3](#) shows examples of truth values for $P, Q, P \wedge Q$ and $P \vee Q$, when we selected the minimum and product t-norms, and the maximum and algebraic sum t-conorms, respectively. For $p, q \in \{0, 1\}$, the results coincide with the classic interpretation of conjunction and disjunction for any choice of the triangular norm and conorm. The differences are present when $p, q \in (0, 1)$.

Table 1.3: Triangular norms as generalized logical connectives.

p	q	$\text{Min}(p, q)$	$\text{Max}(p, q)$	$p \text{ t } q$	$p + q - p \text{ t } q$
1	1	1	1	1	1
1	0	0	1	0	1
0	1	0	1	0	1
0	0	0	0	0	0
0.2	0.5	0.2	0.5	0.1	0.6
0.5	0.8	0.5	0.8	0.4	0.9
0.8	0.7	0.7	0.8	0.56	0.94

Table 1.4: φ operator for binary values of its arguments.

a	b	$a \Rightarrow b$	$a \varphi b$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	1	1

A point worth noting here concerns the interpretation of set operations in terms of logical connectives. By being supported by the isomorphism between set theory and propositional two-valued logic, the intersection and union can be identified with conjunction and disjunction, respectively. This can also be realized with triangular norms viewed as general conjunctive and disjunctive connectives within the framework of multivalued logic (Klir and Yuan, 1995). Triangular norms also play a key role in different types of fuzzy logic (Klement and Navarra, 1999).

Given a continuous t-norm t , let us define the following φ operator

$$a \varphi b = \sup\{c \in [0, 1] | a \text{ t } c \leq b\} \quad \text{for all } a, b \in [0, 1].$$

This operation can be interpreted as an implication induced by some t-norm,

$$a \varphi b = a \Rightarrow b,$$

and therefore it is, like implication, an inclusion. The operator φ generalizes the classic implication. As Table 1.4 suggests, the two-valued implication arises as a special case of the φ operator in case when $a, b \in \{0, 1\}$.

Note that $a \varphi b (a \Rightarrow b)$, returns 1, whenever $a \leq b$. If we interpret these two truth values as membership degrees, we conclude that $a \varphi b$ models a multivalued inclusion relationship.

1.4.4. Aggregation of Fuzzy Sets

Several fuzzy sets can be combined (aggregated) to provide a single fuzzy set forming the result of such an aggregation operation. For instance, when we compute intersection and union of fuzzy sets, the result is a fuzzy set whose membership function captures the information carried by the original fuzzy sets. This fact suggests a general view of aggregation of fuzzy sets as a certain transformations performed on their membership functions. In general, we encounter a wealth of aggregation operations (Dubois and Prade, 1985; Bouchon-Meunier, 1998; Calvo *et al.*, 2002; Dubois and Prade, 2004; Beliakov *et al.*, 2007).

Aggregation operations are n -ary functions $g: [0, 1]^n \rightarrow [0, 1]$ with the following properties:

1. Monotonicity $g(x_1, x_2, \dots, x_n) \geq g(y_1, y_2, \dots, y_n)$ if $x_i > y_i$
2. Boundary conditions $g(0, 0, \dots, 0) = 0$

$$g(1, 1, \dots, 1) = 1.$$

Since triangular norms and conorms are monotonic, associative, satisfy the boundary conditions, and then they provide a class of associative aggregation operations whose neutral elements are equal to 1 and 0, respectively. We are, however, not restricted to those as the only available alternatives. The following operators constitute important examples.

1.4.4.1. Averaging operations

In addition to monotonicity and the satisfaction of the boundary conditions, averaging operations are idempotent and commutative. They can be described in terms of the generalized mean (Dyckhoff and Pedrycz, 1984)

$$g(x_1, x_2, \dots, x_n) = \sqrt[p]{\frac{1}{n} \sum_{i=1}^n (x_i)^p} \quad p \in \mathbb{R}, \quad p \neq 0.$$

Generalized mean subsumes well-known averages and operators such as

$$p = 1 \quad g(x_1, x_2, \dots, x_n) = \frac{1}{n} \sum_{i=1}^n x_i \quad \text{arithmetic mean,}$$

$$p \rightarrow 0 \quad g(x_1, x_2, \dots, x_n) = \sqrt[n]{\prod_{i=1}^n x_i} \quad \text{geometric mean,}$$

$$p = -1 \quad g(x_1, x_2, \dots, x_n) = \frac{n}{\sum_{i=1}^n 1/x_i} \quad \text{harmonic mean,}$$

$$p \rightarrow -\infty \quad g(x_1, x_2, \dots, x_n) = \min(x_1, x_2, \dots, x_n) \quad \text{minimum,}$$

$$p \rightarrow \infty \quad g(x_1, x_2, \dots, x_n) = \max(x_1, x_2, \dots, x_n) \quad \text{maximum.}$$

The following relationship holds

$$\min(x_1, x_2, \dots, x_n) \leq g(x_1, x_2, \dots, x_n) \leq \max(x_1, x_2, \dots, x_n).$$

Therefore, generalized means range over the values not being covered by triangular norms and conorms.

1.4.4.2. Ordered weighted averaging (OWA) operations

OWA is a weighted sum whose arguments are ordered (Yager, 1988). Let $w = [w_1, w_2, \dots, w_n]^T$, $w_i \in [0, 1]$, be weights such that

$$\sum_{i=1}^n w_i = 1.$$

Let a sequence of membership values $\{A(x_i)\}$ be ordered as follows $A(x_1) \leq A(x_2) \leq \dots \leq A(x_n)$. The family of OWA (A, w) is defined as the

$$\text{OWA}(A, w) = \sum_{i=1}^n w_i A(x_i).$$

By choosing certain forms of w , we can show that OWA includes several special cases of aggregation operators mentioned before. For instance

1. If $w = [1, 0, \dots, 0]^T$ then $\text{OWA}(A, w) = \min(A(x_1), A(x_2), \dots, A(x_n))$,
2. If $w = [0, 0, \dots, 1]^T$ then $\text{OWA}(A, w) = \max(A(x_1), A(x_2), \dots, A(x_n))$,
3. If $w = [1/n, \dots, 1/n]^T$ then $\text{OWA}(A, w) = \frac{1}{n} \sum_{i=1}^n A(x_i) = \text{arithmetic mean}$.

Varying the values of the weights w_i results in aggregation values located in between minimum and maximum,

$$\min(A(x_1), A(x_2), \dots, A(x_n)) \leq \text{OWA}(A, w) \leq \max(A(x_1), A(x_2), \dots, A(x_n)),$$

and OWA behaves as a compensatory operator, similar to the generalized mean.

1.4.4.3. Uninorms and nullnorms

Triangular norms provide one of possible ways to aggregate membership grades. By definition, the identity elements are 1 (t-norms) and 0 (t-conorms). When used in the aggregation operations, these elements do not affect the result of aggregation (that is, $a \wedge 1 = a$ and $a \vee 0 = a$).

Uninorms generalize and unify triangular norms by allowing the identity element to be any value in the unit interval that is $e \in (0, 1)$. Uninorms become t-norms when $e = 1$ and t-conorms when $e = 0$. They exhibit some intermediate characteristics for all remaining values of e . Therefore, uninorms share the same properties as triangular norms with the exception of the identity (Yager and Rybalov, 1996).

A uninorm is a two place operation $u: [0, 1] \times [0, 1] \rightarrow [0, 1]$ that satisfies the following

1. Commutativity: $a \cup b = b \cup a$,
2. Associativity: $a \cup (b \cup c) = (a \cup b) \cup c$,
3. Monotonicity: if $b \leq c$ then $a \cup b \leq a \cup c$,
4. Identity: $a \cup e = a, \forall a \in [0, 1]$,

where $a, b, c \in [0, 1]$.

Examples of uninorm include conjunctive u_c and disjunctive u_d forms of uninorms. They can be obtained in terms of a t-norm t and a conorm s as follows:

(a) If $(0 \cup 1) = 0$, then

$$a \cup_c b = \begin{cases} e \left(\frac{a}{e} \right) t \left(\frac{b}{e} \right) & \text{if } 0 \leq a, b \leq e \\ e + (1-e) \left(\frac{a-e}{1-e} \right) s \left(\frac{b-e}{1-e} \right) & \text{if } e \leq a, b \leq 1 \\ \min(a, b) & \text{otherwise} \end{cases}$$

(b) If $(0 \cup 1) = 1$, then

$$a \cup_d b = \begin{cases} e \left(\frac{a}{e} \right) t \left(\frac{b}{e} \right) & \text{if } 0 \leq a, b \leq e \\ e + (1-e) \left(\frac{a-e}{1-e} \right) s \left(\frac{b-e}{1-e} \right) & \text{if } e \leq a, b \leq 1 \\ \max(a, b) & \text{otherwise} \end{cases}$$

1.5. Fuzzy Relations

Relations represent and quantify associations between objects. They provide a mechanism to model interactions and dependencies between variables, components, modules, etc. Fuzzy relations generalize the concept of relation in the same manner as fuzzy set generalizes the fundamental idea of set. Fuzzy relations have applications especially in information retrieval, pattern classification, modeling and control, diagnostics, and decision-making.

1.5.1. Relations and Fuzzy Relations

Fuzzy relation is a generalization of the concept of relations by admitting the notion of partial association between elements of domains. Intuitively, a fuzzy relation can be seen as a multidimensional fuzzy set. For instance, if X and Y are two domains of interest, a fuzzy relation R is any fuzzy subset of the Cartesian product of X and Y (Zadeh, 1971). Equivalently, a fuzzy relation on $X \times Y$ is a mapping

$$R : X \times Y \rightarrow [0, 1].$$

The membership function of R for some pair (x, y) , $R(x, y) = 1$, denotes that the two objects x and y are fully related. On the other hand, $R(x, y) = 0$ means that these elements are unrelated while the values in between, $0 < R(x, y) < 1$, underline a partial association. For instance, if d_{fs} , d_{nf} , d_{ns} , d_{gf} are documents whose subjects concern mainly fuzzy systems, neural fuzzy systems, neural systems and genetic fuzzy systems, with keywords w_f , w_n and w_g , respectively, then a relation R on $D \times W$, $D = \{d_{fs}, d_{nf}, d_{ns}, d_{gf}\}$ and $W = \{w_f, w_n, w_g\}$ can assume the matrix form with the following entries

$$R = \begin{bmatrix} 1 & 0 & 0.6 \\ 0.8 & 1 & 0 \\ 0 & 0 & 0 \\ 0.8 & 0 & 1 \end{bmatrix}.$$

Since the universes are discrete, R can be represented as a 4×3 matrix (four documents and three keywords) and entries are degrees of memberships. For instance $R(d_{fs}, w_f) = 1$ means that the document content d_{fs} is fully compatible with the keyword w_f whereas $R(d_{fs}, w_n) = 0$ and $R(d_{fs}, w_g) = 0.6$ indicate that d_{fs} does not mention neural systems, but does have genetic systems as part of its content. As with relations, when X and Y are finite with $\text{Card}(X) = n$ and $\text{Card}(Y) = m$, then R can be arranged into a certain $n \times m$ matrix $R = [r_{ij}]$, with $r_{ij} \in [0, 1]$ being the corresponding degrees of association between x_i and y_j .

The basic operations on fuzzy relations, union, intersection, and complement, are analogous to the corresponding operations on fuzzy sets once fuzzy relations are fuzzy sets formed on multidimensional spaces. Their characterization and representation also mimics fuzzy sets.

1.5.1.1. Cartesian product

A procedure to construct fuzzy relations is through the use of Cartesian product extended to fuzzy sets. The concept closely follows the one adopted for sets once they involve pairs of points of the underlying universes, added with a membership degree.

Given fuzzy sets A_1, A_2, \dots, A_n on the respective domains X_1, X_2, \dots, X_n , their Cartesian product $A_1 \times A_2 \times \dots \times A_n$ is a fuzzy relation R on $X_1 \times X_2 \times \dots \times X_n$ with the following membership function

$$R(x_1, x_2, \dots, x_n) = \min\{A_1(x_1), A_2(x_2), \dots, A_n(x_n)\} \quad \forall x_1 \in X_1, \forall x_2 \in X_2, \dots, \forall x_n \in X_n.$$

In general, we can generalize the concept of this Cartesian product using t-norms:

$$R(x_1, x_2, \dots, x_n) = A_1(x_1) \text{t} A_2(x_2) \text{t} \dots \text{t} A_n(x_n) \quad \forall x_1 \in X_1, \forall x_2 \in X_2, \dots, \forall x_n \in X_n.$$

1.5.1.2. Projection of fuzzy relations

Contrasting with the concept of the Cartesian product, the idea of projection is to construct fuzzy relations on some subspaces of the original relation.

If R is a fuzzy relation on $X_1 \times X_2 \times \dots \times X_n$, its projection on $X = X_i \times X_j \times \dots \times X_k$, is a fuzzy relation R_X whose membership function is (Zadeh, 1975a, 1975b)

$$R_X(x_i, x_j, \dots, x_k) = \text{Pr}oj_X R(x_1, x_2, \dots, x_n) = \sup_{x_t, x_u, \dots, x_v} R(x_1, x_2, \dots, x_n),$$

where $I = \{i, j, \dots, k\}$ is a subsequence of the set of indexes $N = \{1, 2, \dots, n\}$, and $J = \{t, u, \dots, v\}$ is a subsequence of N such that $I \cup J = N$ and $I \cap J = \emptyset$. Thus, J is the complement of I with respect to N . Notice that the above expression is computed for all values of $(x_1, x_2, \dots, x_n) \in X_1 \times X_2 \times \dots \times X_n$. [Figure 1.23](#) illustrates projection in the two-dimensional $X \times Y$ case.

1.5.1.3. Cylindrical extension

The notion of cylindrical extension aims to expand a fuzzy set to a multidimensional relation. In this sense, cylindrical extension can be regarded as an operation complementary to the projection operation (Zadeh, 1975a, 1975b).

The cylindrical extension on $X \times Y$ of a fuzzy set of X is a fuzzy relation $cylA$ whose membership function is equal to

$$cylA(x, y) = A(x) \quad \forall x \in X, y \in Y.$$

[Figure 1.24](#) shows the cylindrical extension of a triangular fuzzy set A .

Relation R and its projections on X and Y

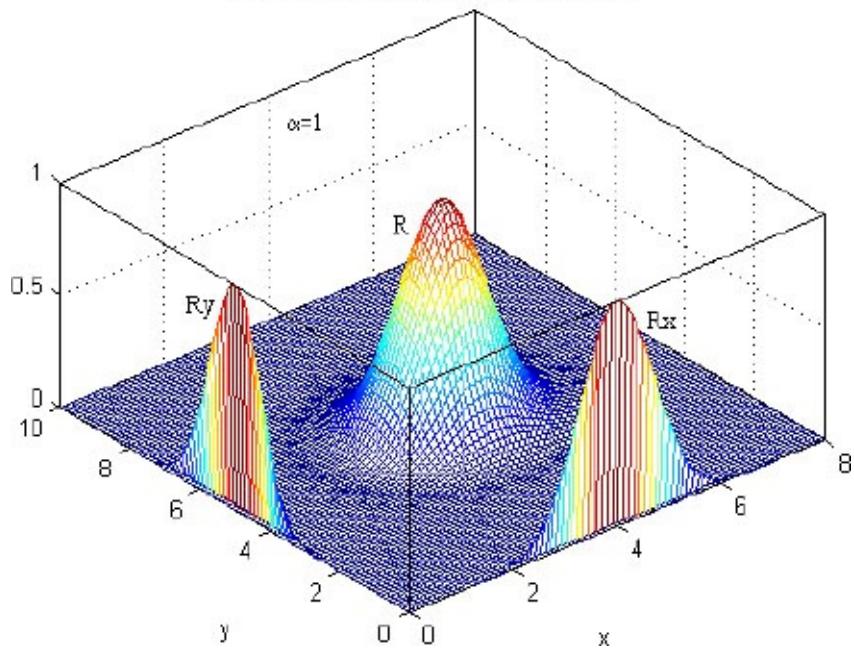


Figure 1.23: Fuzzy relation R and its projections on X and Y .

Cylindrical Extension of A on X and Y

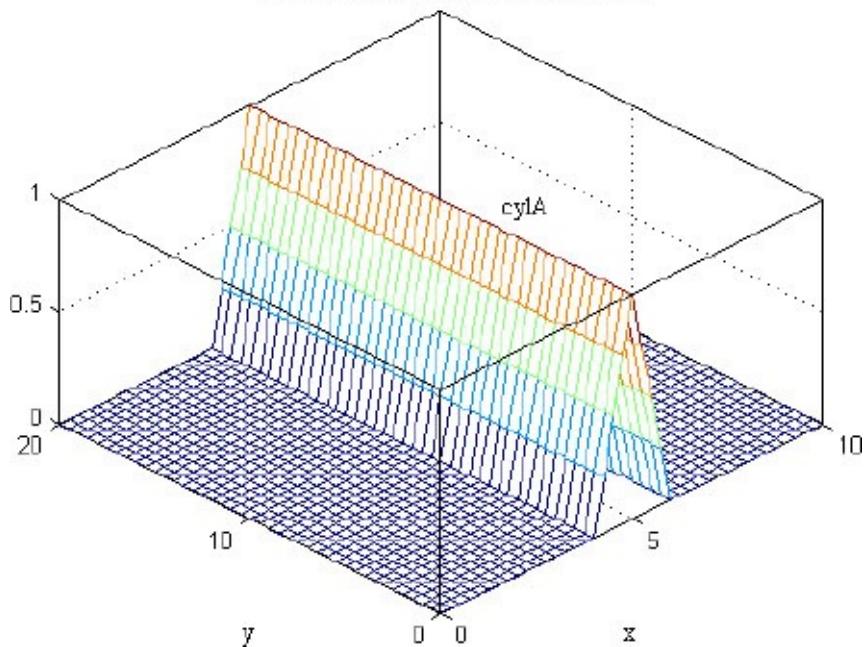


Figure 1.24: Cylindrical extension of a fuzzy set.

1.6. Linguistic Variables

One frequently deals with variables describing phenomena of physical or human systems assuming a finite, quite small number of descriptors.

In contrast to the idea of numeric variables as commonly used, the notion of linguistic variable can be understood as a variable whose values are fuzzy sets. In general, linguistic variables may assume values consisting of words or sentences expressed in a certain language (Zadeh, 1999a). Formally, a linguistic variable is characterized by a quintuple $\langle X, T(X), X, G, M \rangle$ where X is the name of the variable, $T(X)$ is a term set of X whose elements are labels L of linguistic values of X , G is a grammar that generates the names of X , and M is a semantic rule that assigns to each label $L \in T(X)$ a meaning whose realization is a fuzzy set on the universe X with base variable x . [Figure 1.25](#) gives an example.

1.7. Granulation of Data

The notion of granulation emerges as a need to abstract and summarize data to support the processes of comprehension and decision-making. For instance, we often sample an environment for values of attributes of state variables, but we rarely process all details because of our physical and cognitive limitations. Quite often, just a reduced number of variables, attributes, and values are considered because those are the only features of interest given the task under consideration. To avoid all necessary and highly distractive details, we require an effective abstraction procedure. Detailed numeric information is aggregated into a format of information granules where the granules themselves are regarded as collections of elements that are perceived as being indistinguishable, similar, close, or functionally equivalent.

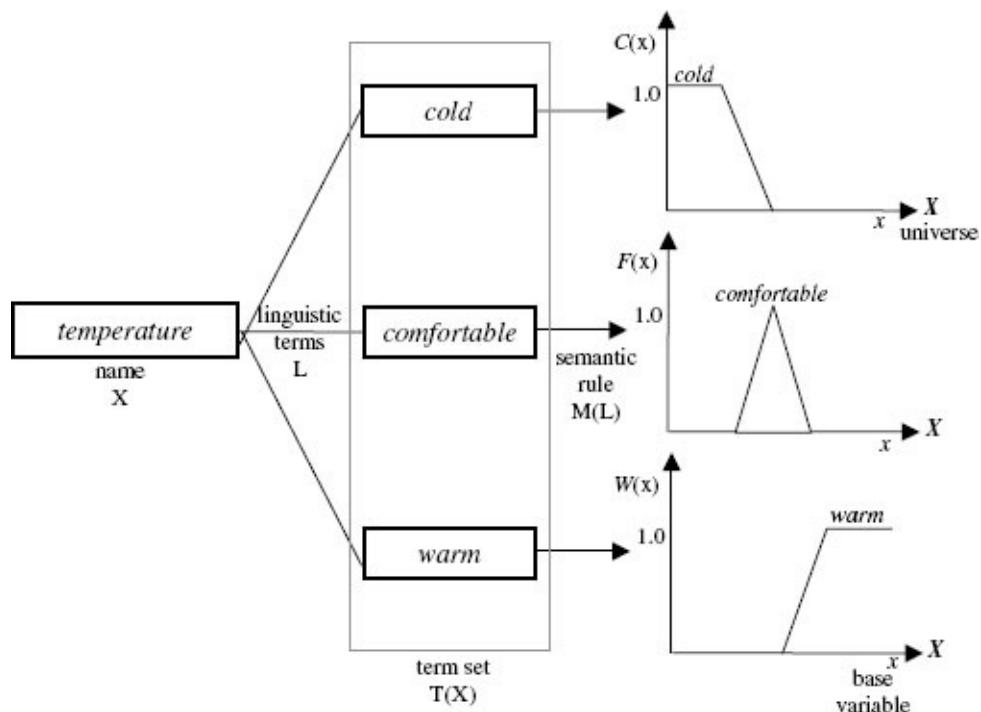


Figure 1.25: An example of the linguistic variable temperature.

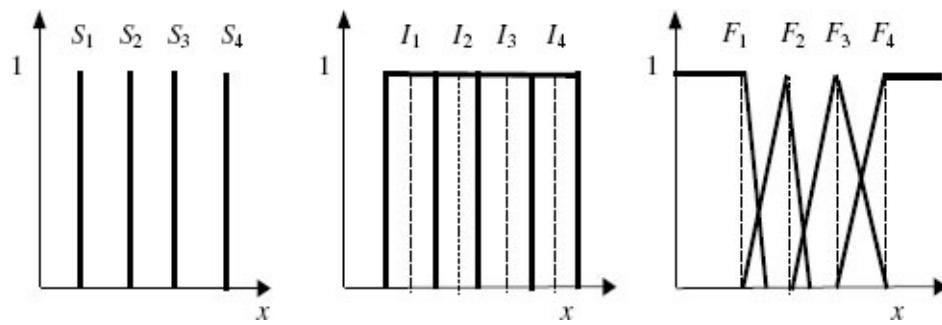


Figure 1.26: Discretization, quantization, and fuzzy granulation.

There are different formalisms and concepts of information granules. For instance, granules can be realized as sets (intervals), rough sets, probability densities (Lin, 2004). Typical examples of the granular data are singletons and intervals. In these two special cases we typically refer to discretization and quantization, as in [Figure 1.26](#). As the specificity of granules increases, intervals become singletons and in this case limit the

quantization results in a discretization process.

Fuzzy sets are examples of information granules. When talking about a family of fuzzy sets, we are typically concerned with fuzzy partitions of X . Given the nature of fuzzy sets, fuzzy granulation generalizes the notion of quantization as in [Figure 1.26](#) and emphasizes a gradual nature of transitions between neighboring information granules (Zadeh, 1999b). When dealing with information granulation we often develop a family of fuzzy sets and move on with the processing that inherently uses all the elements of this families. The existing terminology refers to such collections of data granules as frames of cognition (Pedrycz and Gomide, 2007). In what follows, we briefly review the concept and its main properties.

1.7.1. Frame of Cognition

A frame of cognition results from information granulation when we encounter a finite collection of fuzzy sets—information granules that represent the entire universe of discourse and satisfy a system of semantic constraints. The frame of cognition is a notion of particular interest in fuzzy modeling, fuzzy control, classification, and data analysis.

A frame of cognition consists of several labeled, normal fuzzy sets. Each of these fuzzy sets is treated as a reference for further processing. A frame of cognition can be viewed as a codebook of conceptual entities. We may view them as a family of linguistic landmarks, say *small*, *medium*, *high*, etc. More formally, a frame of cognition Φ

$$\Phi = \{A_1, A_2, \dots, A_m\} \quad (18)$$

is a collection of fuzzy sets defined in the same universe X that satisfies at least two requirements of coverage and semantic soundness.

1.7.2. Coverage

We say that Φ covers X if any element $x \in X$ is compatible with at least one fuzzy sets A_i in Φ , $i \in I = \{1, 2, \dots, m\}$ meaning that it is compatible (coincides) with A_i to some non-zero degree, that is

$$\forall_{x \in X} \exists_{i \in I} A_i(x) > 0. \quad (19)$$

Being stricter, we may require a satisfaction of the so-called δ -level coverage which means that for any element of X , fuzzy sets are activated to a degree not lower than δ

$$\forall_{x \in X} \exists_{i \in I} A_i(x) > \delta, \quad (20)$$

where $\delta \in [0, 1]$. From application perspective, the coverage assures that each element of X is represented by at least one of the elements of Φ , and guarantees any absence of gaps,

that is, elements of X for which there is no fuzzy set being compatible with it.

1.7.3. Semantic Soundness

The notion of semantic soundness is more complicated and difficult to quantify. In principle, we are interested in information granules of Φ that are meaningful. While there is far more flexibility in a way in which a number of detailed requirements could be structured, we may agree upon a collection of several fundamental properties.

Each $A_i, i \in I$, is a unimodal and normal fuzzy set.

Fuzzy sets $A_i, i \in I$, are disjoint enough to assure that they are sufficiently distinct to become linguistically meaningful. This imposes a maximum degree λ of overlap between any two elements of Φ . In other words, given any $x \in X$, there is no more than one fuzzy set A_i such that $A_i(x) \geq \lambda, \lambda \in [0, 1]$.

The number of elements of Φ is low; following the psychological findings reported by Miller and others we consider the number of fuzzy sets forming the frame of cognition to be maintained in the range of 7 ± 2 items.

Coverage and semantic soundness (Oliveira, 1993) are the two essential conditions that should be fulfilled by the membership functions of A_i to achieve interpretability. In particular, δ -coverage and λ -overlapping induce a minimal (δ) and maximal (λ) level of overlap between fuzzy sets, as in [Figure 1.27](#).

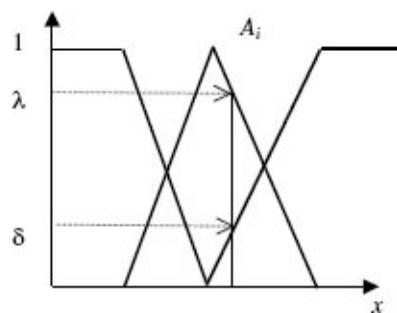


Figure 1.27: Coverage and semantic soundness of a cognitive frame.

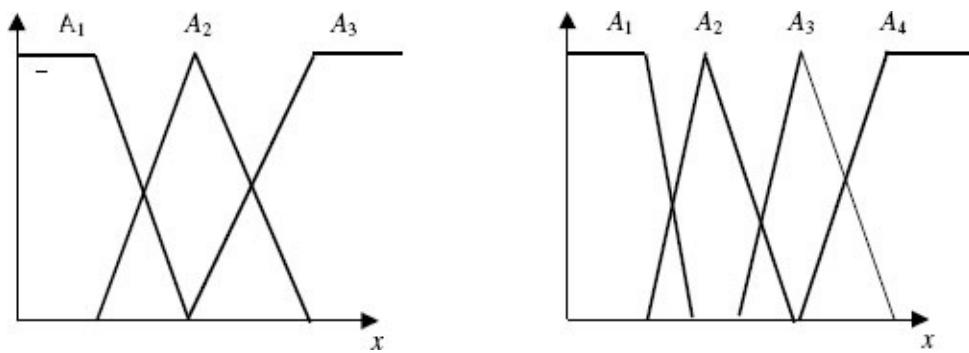


Figure 1.28: Two frames of cognition; Φ_1 is coarser (more general) than Φ_2 .

Considering the families of linguistic labels and associated fuzzy sets embraced in a frame of cognition, several characteristics are worth emphasizing.

1.7.3.1. Specificity

We say that the frame of cognition Φ_1 is more specific than Φ_2 if all the elements of Φ_1 are more specific than the elements of Φ_2 , as in [Figure 1.28](#). Here the specificity $Spec(A_i)$ of the fuzzy sets that compose the cognition frames can be evaluated as suggested in [Section 1.3](#). The less specific cognition frames promotes granulation realized at the higher level of abstraction (generalization). Subsequently, we are provided with the description that captures less details.

1.7.3.2. Granularity

Granularity of a frame of cognition relates to the granularity of fuzzy sets used there. The higher the number of fuzzy sets in the frame is, the finer the resulting granulation. Therefore, the frame of cognition Φ_1 is finer than Φ_2 if $|\Phi_1| > |\Phi_2|$. If the converse holds, Φ_1 is coarser than Φ_2 , as in [Figure 1.28](#).

1.7.3.3. Focus of attention

A focus of attention induced by a certain fuzzy set $A = A_i$ in Φ is defined as a certain **α -cut** of this fuzzy set. By moving A along X while keeping its membership function unchanged, we can focus attention on a certain selected region of X , as shown in [Figure 1.29](#).

1.7.3.4. Information hiding

Information hiding is closely related to the notion of focus of attention and manifests through a collection of elements that are hidden when viewed from the standpoint of membership functions. By modifying the membership function of $A = A_i$ in Φ we can produce an equivalence of the elements positioned within some region of X . For instance, consider a trapezoidal fuzzy set A on \mathbf{R} and its 1-cut (core), the closed interval $[a_2, a_3]$, as depicted in [Figure 1.30](#).

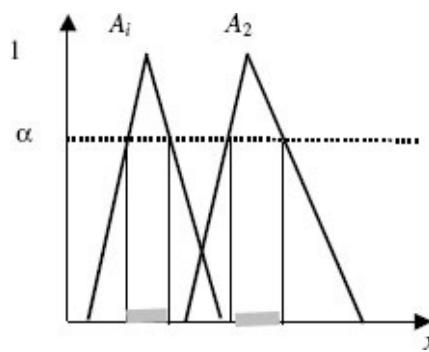


Figure 1.29: Focus of attention; two regions of focus of attention implied by the corresponding fuzzy sets.

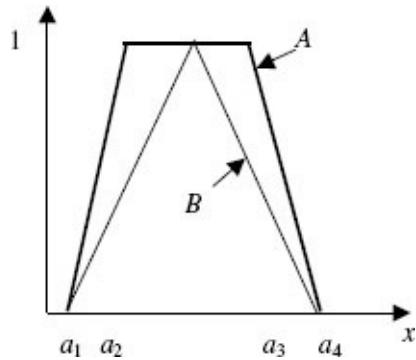


Figure 1.30: A concept of information hiding realized by the use of trapezoidal fuzzy set A . Points in $[a_2, a_3]$ are made indistinguishable. The effect of information hiding is not present in case of triangular fuzzy set B .

All points within the interval $[a_2, a_3]$ are made *indistinguishable* and through the use of this specific fuzzy set they are made equivalent. Hence, more detailed information, a position of a certain point falling within this interval, is *hidden*. In general, by increasing or decreasing the level of the α -cut, we can accomplish a so-called α -information hiding through normalization.

1.8. Conclusion

The chapter has summarized the fundamental notions and concepts of fuzzy set theory. The goal was to offer basic and key contents of interest for computational intelligence and intelligent systems theory and applications. Currently, a number of outstanding books and journals are available to help researchers, scientists and engineers to master the fuzzy set theory and the contributions it brings to develop new approaches through hybridizations and new applications. The references section includes some of them. The remaining chapters of this book provide the readers with a clear picture of the current state of the art in the area.

References

- Bouchon-Meunier, B. (1998). *Aggregation and Fusion of Imperfect Information*. Heidelberg, Germany: Physica-Verlag.
- Beliakov, G., Pradera, A. and Calvo, T. (2007). *Aggregation Functions: A Guide for Practitioners*. Heidelberg, Germany: Springer.
- Calvo, T., Kolesárová, A., Komorníková, M. and Mesiar, R. (2002). *Aggregation Operators: Properties, Classes and Construction Methods*, in *Aggregation Operators: New Trends and Applications*. Heidelberg, Germany: Physica-Verlag.
- Cantor, G. (1883). *Grundlagen Einer Allgemeinen Mannigfaltigekeitslehre*. Leipzig: Teubner.
- Cantor, G. (1895). Breitraage zur begraundung der transfniten mengernlehre. *Math. Ann.*, 46, pp. 207–246.
- Dubois, D. and Prade, H. (1985). A review of fuzzy aggregation connectives. *Info. Sciences*, 36, pp. 85–121.
- Dubois, D. and Prade, H. (1997). The three semantics of fuzzy sets. *Fuzzy Sets Syst.*, 2, pp. 141–150.
- Dubois, D. and Prade, H. (1998). An introduction to fuzzy sets. *Clin. Chim. Acta*, 70, pp. 3–29.
- Dubois, D. and Prade, H. (2004). On the use of aggregation operations in information fusion. *Fuzzy Sets Syst.*, 142, pp. 143–161.
- Dyckhoff, H. and Pedrycz, W. (1984). Generalized means as a models of compensative connectives. *Fuzzy Sets Syst.*, 142, pp. 143–154.
- Klement, E. and Navarra, M. (1999). Propositional fuzzy logics based on Frank t-norms: A comparison. In D. Dubois, H. Parde and W. Klement (eds.), *Fuzzy Sets, Logics and Reasoning about Knowledge*. Dordrecht, the Netherlands: Kluwer Academic Publishers, pp. 25–47.
- Klement, P., Mesiar, R. and Pap, E. (2000). *Triangular Norms*. Dordrecht, Nederland: Kluwer Academic Publishers.
- Klir, G. and Yuan, B. (1995). *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Upper Saddle River, New Jersey, USA: Prentice Hall.
- Lin, T. (2004). Granular computing: Rough sets perspectives, *IEEE Connections*, 2, pp. 10–13.
- Oliveira, J. (1993). On optimal fuzzy systems with I/O interfaces. *Proc. Second IEEE Int. Conf. on Fuzzy Systems*. San Francisco, California, USA, pp. 34–40.
- Pedrycz, W. and Gomide, F. (2007). *Fuzzy Systems Engineering: Toward Human-Centric Computing*. Hoboken, New Jersey, USA: Wiley Intercience.
- Yager, R. (1983). Entropy and specificity in a mathematical theory of evidence. *Int. J. Gen. Syst.*, 9, pp. 249–260.
- Yager, R. (1988). On ordered weighted averaging aggregation operations in multicriteria decision making. *IEEE Trans. Syst. Man Cyber.*, 18, pp. 183–190.
- Yager, R. and Rybalov, A. (1996). Uninorm aggregation operators. *Fuzzy Sets Syst.*, 80, pp. 111–120.
- Zadeh, L. (1965). Fuzzy sets. *Inf. Control*, 8, pp. 338–353.
- Zadeh, L. (1971). Similarity relations and fuzzy orderings. *Inf. Sci.*, 3, pp. 177–200.
- Zadeh, L. (1975a, 1975b). The concept of linguistic variables and its application to approximate reasoning I, II, III. *Info. Sciences*, 8(9), pp. 43–80, 199–251, 301–357.
- Zadeh, L. (1978). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets Syst.*, 3, pp. 3–28.
- Zadeh, L. (1999a). From computing with numbers to computing with words: from manipulation of measurements to manipulation of perceptions. *IEEE Trans. Circ. Syst.*, 45, pp. 105–119.
- Zadeh, L. (1999b). Fuzzy logic = computing with words. In Zadeh, L. and Kacprzyk, J. (eds.), *Computing with Words in Information and Intelligent Systems*. Heidelberg, Germany: Physica-Verlag, pp. 3–23.

Chapter 2

Granular Computing

Andrzej Bargiela and Witold Pedrycz

Research into human-centered information processing, as evidenced through the development of fuzzy sets and fuzzy logic, has brought an additional insight into the transformative nature of the aggregation of inaccurate and/or fuzzy information in terms of the semantic content of data aggregates. This insight has led to the suggestion of the development of the granular computing (GrC) paradigm some 15 years ago and takes as a point of departure, an empirical validation of the aggregated data in order to achieve the closest possible correspondence of the semantics of data aggregates and the entities in the problem domain. Indeed, it can be observed that information abstraction combined with empirical validation of abstractions has been deployed as a methodological tool in various scientific disciplines. This chapter is focused on exploring the foundations of GrC and casting it as a structured combination of algorithmic and non-algorithmic information processing that mimics human, intelligent synthesis of knowledge from imprecise and/or fuzzy information.

2.1. Introduction

Granular Computing (GrC) is frequently defined in an informal way as a general computation theory for effectively using granules such as classes, clusters, subsets, groups, and intervals to build an efficient computational model for complex applications which process large volumes of information presented as either raw data or aggregated problem domain knowledge. Though the GrC term is relatively recent, the basic notions and principles of granular computing have appeared under different names in many related fields, such as information hiding in programming, granularity in artificial intelligence, divide and conquer in theoretical computer science, interval computing, cluster analysis, fuzzy and rough set theories, neutrosophic computing, quotient space theory, belief functions, machine learning, databases, and many others. In the past few years, we have witnessed a renewed and fast growing interest in GrC. GrC has begun to play important roles in bioinformatics, e-Business, security, machine learning, data mining, high-performance computing, and wireless mobile computing in terms of efficiency, effectiveness, robustness and structured representation of uncertainty.

With the vigorous research interest in the GrC paradigm (Bargiela and Pedrycz, 2002a, 2002b, 2003, 2005a, 2005b; Bargiela, 2004; Bargiela *et al.*, 2004a, 2004b; Inuiguchi *et al.*, 2003; Lin, 1998; Lin *et al.*, 2002; Pedrycz, 1989; Pedrycz and Gomide, 1998; Pedrycz *et al.*, 2000; Pedrycz and Bargiela, 2002; Skowron and Stepaniuk, 2001; Yao and Yao, 2002; Yao, 2004a, 2004b, 2005; Zadeh, 1965, 1979, 1997, 2002; Dubois *et al.*, 1997), it is natural to see that there are voices calling for clarification of the distinctiveness of GrC from the underpinning constituent disciplines and from other computational paradigms proposed for large-scale/complex information possessing Pawlak (1991). Recent contributions by Yao and Yao (2002), Yao (2004a, 2004b, 2005) attempt to bring together various insights into GrC from a broad spectrum of disciplines and cast the GrC framework as a structured thinking at the philosophical level and structured problem solving at the practical level.

In this chapter, we elaborate on our earlier proposal (Bargiela and Pedrycz, 2006) and look at the roots of GrC in the light of the original insight of Zadeh (1997) stating, "...fuzzy information granulation in an intuitive form underlies human problem solving ...". We suggest that the *human problem solving* has strong foundations in axiomatic set theory and theory of computability and it underlies some recent research results linking intelligence to physical computation (Bains, 2003, 2005). In fact, re-examining human information processing in this light brings GrC from a domain of computation and philosophy to one of physics and set theory.

The set theoretic perspective on GrC adopted in this chapter offers also a good basis for the evaluation of other human-centered computing paradigms such as the generalized constraint-based computation recently communicated by Zadeh.

2.2. Set Theoretical Interpretation of Granulation

The commonly accepted definition of granulation introduced in Inuiguchi *et al.* (2003), Lin *et al.* (2002), and Yao (2004a) is:

Definition 1: Information granulation is a grouping of elements based on their indistinguishability, similarity, proximity or functionality.

This definition serves well the purpose of constructive generation of granules but it does little to differentiate granulation from clustering. More importantly however, *Definition 1* implies that the nature of information granules is fully captured by their interpretation as subsets of the original dataset within the Intuitive Set Theory of Cantor (1879). Unfortunately, an inevitable consequence of that is that the inconsistencies (paradoxes) associated with intuitive set theory, such as “cardinality of set of all sets” (Cantor, 1879) or “definition of a set that is not a member of itself” (Russel, 1937) are imported into the domain of information granulation.

In order to provide a more robust definition of information granulation we follow the approach adopted in the development of axiomatic set theory. The key realization there was that the commonly accepted intuition, that one can form any set one wants, should be questioned. Accepting the departure point of intuitive set theory we can say that, normally, sets are not members of themselves, i.e., normally, $\sim(y \text{ in } y)$. But, the axioms of intuitive set theory do not exclude the existence of “abnormal” sets, which are members of themselves. So, if we consider a set of all “normal” sets: $x = \{y \mid \sim(y \text{ in } y)\}$ we can axiomatically guarantee the existence of set x :

$$\exists x \forall y (y \in x \Leftrightarrow \sim(y \in y)).$$

If we then substitute x for y , we arrive at a contradiction:

$$\exists x \forall x (x \in x \Leftrightarrow \sim(x \in x)).$$

So, the unrestricted comprehension axiom of the intuitive set theory leads to contradictions and cannot therefore serve as a foundation of set theory.

2.2.1. Zermelo–Fraenkel (ZF) Axiomatization

An early attempt at overcoming the above contradiction was an axiomatic scheme developed by Ernst Zermelo and Abraham Fraenkel (Zermelo, 1908). Their idea was to restrict the comprehension axiom schema by adopting only those instances of it, which are necessary for reconstruction of common mathematics. In other words, the standard approach, of using a formula $F(y)$ to collect the set y having the property F , leads to generation of an object that is not a set (otherwise we arrive at a contradiction). So, looking at the problem the other way, they have concluded that the contradiction constitutes a *de-facto* proof that there are other semantical entities in addition to sets.

The important observation that we can make here is that the semantical transformation of sets through the process of applying some set-forming formula applies also to the process of information granulation and consequently, information granules should be considered as being semantically distinct from the granulated entities. We therefore arrive at a modified definition of information granulation as follows:

Definition 2: *Information granulation is a semantically meaningful grouping of elements based on their indistinguishability, similarity, proximity or functionality.*

Continuing with the ZF approach we must legalize some collections of sets that are not sets. Let $F(y, z_1, z_2, \dots, z_n)$ be a formula in the language of set theory (where z_1, z_2, \dots, z_n are optional parameters). We can say that for any values of parameters z_1, z_2, \dots, z_n the formula F defines a “class” A

$$A = \{y \mid F(y, z_1, z_2, \dots, z_n)\},$$

which consists of all y 's possessing the property F . Different values of z_1, z_2, \dots, z_n give rise to different classes. Consequently, the axiomatization of set theory involves formulation of axiom schemas that represent possible instances of axioms for different classes.

The following is a full set of axioms of the ZF set theory:

Z1, Extensionality:

$$\forall x \forall y [\forall z (z \in x \equiv z \in y) \Rightarrow x = y].$$

Asserts that if sets x and y have the same members, the sets are identical.

Z2, Null Set:

$$\exists x \sim \exists y (y \in x).$$

Asserts that there is a unique empty set.

Z3, Pair Set:

$$\forall x \forall y \exists z \forall w (w \in z \equiv w = x \vee w = y).$$

Asserts that for any set x and y , there exists a pair set of x and y , i.e., a set that has only x and y as members.

Z4, Unions:

$$\forall x \exists y \forall z (z \in y \equiv \exists w (w \in x \wedge z \in w)).$$

Asserts that for any set x there is a set y containing every set that is a member of some member of x .

Z5, Power Set:

$$\forall x \exists y \forall z [z \in y \equiv \forall w (w \in z \Rightarrow w \in x)].$$

Asserts that for any set x , there is a set y which contains as members all those sets whose members are also elements of x , i.e., y contains all of the subsets of x .

Z6, Infinite Set:

$$\exists x [\emptyset \in x \wedge \forall y (y \in x \Rightarrow \cup\{y, \{y\}\} \in x)].$$

Asserts that there is a set x which contains \emptyset as a member and which is such that, whenever y is a member of x , then $y \cup \{y\}$ is a member of x .

Z7, Regularity:

$$\forall x [x \neq \emptyset \Rightarrow \exists y (y \in x \wedge \forall z (z \in x \Rightarrow \sim (z \in y)))].$$

Asserts that every set is “well-founded”, i.e., it rules out the existence of circular chains of sets as well as infinitely descending chains of sets. A member y of a set x with this property is called a “minimal” element.

Z8, Replacement Schema:

$$\forall x \exists y F(x, y) \Rightarrow \forall u \exists v \forall r (r \in v \equiv \exists s (s \in u \wedge F_{x,y}[s, r])).$$

Asserts that given a formula $F(x, y)$ and $F_{x,y}[s, r]$ as a result of substituting s and r for x and y , every instance of the above axiom schema is an axiom. In other words, given a functional formula F and a set u we can form a new set v by collecting all of the sets to which the members of u are uniquely related by F . It is important to note that elements of v need not be elements of u .

Z9, Separation Schema:

$$\forall u \exists v \forall r (r \in v \equiv r \in u \wedge F_x[r]).$$

Asserts that there exists a set v which has as members precisely the members of u which satisfy the formula F . Again, every instance of the above axiom schema is an axiom.

Unfortunately, the presence of the two axiom schemas, Z6 and Z7, implies infinite axiomatization of the ZF set theory. While it is fully acknowledged that the ZF set theory, and its many variants, has advanced our understanding of cardinal and ordinal numbers and has led to the proof of the property of “well-ordering” of sets (with the help of an additional “Axiom of Choice”) (ZFC), the theory seems unduly complex for the purpose of set-theoretical interpretation of information granules.

2.2.2. von Neumann–Bernays–Goedel (NBG) Axiomatization

A different approach to the axiomatization of set theory designed to yield the same results as ZF but with a finite number of axioms (i.e., without the reliance on axiom schemas) has

been proposed by von Neumann in 1920 and subsequently has been refined by Bernays in 1937 and Goedel in 1940 (Goedel, 1940). The defining aspect of NBG set theory is the introduction of the concept of “proper class” among its objects. NBG and ZFC are very closely related and, in fact, NBG is a conservative extension of ZFC.

In NBG, the proper classes are differentiated from sets by the fact that they do not belong to other classes. Thus, in NBG we have

$$x \Leftrightarrow \exists y(x \in y),$$

which can be phrased as x is a set if it belongs to either a set or a class.

The basic observation that can be made about NBG is that it is essentially a two-sorted theory; it involves sets (denoted here by lower-case letters) and classes (denoted by upper-case letters). Consequently, the above statement about membership assumes one of the forms

$$x \in y \text{ or } x \in Y,$$

and statements about equality are in the form

$$x = y \text{ or } X = Y.$$

Using this notation, the axioms of NBG are as follows

N1, Class Extensionality:

$$\forall x[x \in A \Leftrightarrow x \in B] \Rightarrow A = B].$$

Asserts that classes with the same elements are the same.

N2, Set Extensionality:

$$\forall x[x \in a \Leftrightarrow x \in b] \Rightarrow a = b].$$

Asserts that sets with the same elements are the same.

N3, Pairing:

$$\forall x \forall y \exists z \forall w(w \in z \equiv w = x \vee w = y).$$

Asserts that for any set x and y , there exists a set $\{x, y\}$ that has exactly two elements x and y . It is worth noting that this axiom allows definition of ordered pairs and taken together with the Class Comprehension axiom, it allows implementation of relations on sets as classes.

N4, Union:

$$\forall x \exists y \forall z(z \in y \equiv \exists w(w \in x \wedge z \in w)).$$

Asserts that for any set x , there exists a set which contains exactly the elements of x .

N5, Power Set:

$$\forall x \exists y \forall z [z \in y \equiv \exists w (w \in z \Rightarrow w \in x)].$$

Asserts that for any set x , there is a set which contains exactly the subsets of x .

N6, Infinite Set:

$$\exists x [\emptyset \in x \wedge \forall y (y \in x \Rightarrow \cup\{y, \{y\}\} \in x)].$$

Asserts there is a set x , which contains an empty set as an element and contains $y \cup \{y\}$ for each of its elements y .

N7, Regularity:

$$\forall x [x \neq \emptyset \Rightarrow \exists y (y \in x \wedge \forall z (z \in x \Rightarrow \sim (z \in y)))].$$

Asserts that each non-empty set is disjointed from one of its elements.

N8, Limitation of size:

$$\sim x \Leftrightarrow |x| = |V|.$$

Asserts that if the cardinality of x equals to the cardinality of the set theoretic universe V , x is not a set but a proper class. This axiom can be shown to be equivalent to the axioms of Regularity, Replacement and Separation in NBG. Thus the classes that are proper in NBG are in a very clear sense big, while sets are small.

It should be appreciated that the latter has a very profound implication on computation, which processes proper classes. This is because the classes built over countable sets can be uncountable and, as such, do not satisfy the constraints of the formalism of the Universal Turing Machine.

N9, Class Comprehension schema:

Unlike in the ZF axiomatization, this schema consists of a finite set of axioms (thus giving finite axiomatization of NBG).

Axiom of Sets: For any set x , there is a class X such that $x = X$.

Axiom of Complement: For any class X , the complement $V - X = \{x \mid x \notin X\}$

Axiom of Intersection: For any class X and Y the intersection $X \cap Y = \{x \mid x \in X \wedge x \in Y\}$ is a class.

Axiom of Products: For any classes X and Y , the class $X \times Y = \{(x, y) \mid x \in X \wedge y \in Y\}$ is a class. This axiom provides actually for more than what is needed for representing relations on classes. What is actually needed is just that $V \times Y$ is a class.

Axiom of Converses: For any class X , the classes $\text{Conv}_1(X) = \{(y, x) \mid (x, y) \in X\}$

and $\text{Conv}2(X) = \{(y, (x, z)) \mid (x, (y, z)) \in X\}$ exist.

Axiom of Association: For any class X , the classes $\text{Assoc}1(X) = \{((x, y), z) \mid (x, (y, z)) \in X\}$ and $\text{Assoc}2(X) = \{(w, (x, (y, z))) \mid (w, ((x, y), z)) \in X\}$ exist.

Axiom of Ranges: For any class X , the class $\text{Rng}(X) = \{y \mid (\exists x(x, y) \in X)\}$ exists.

Axiom of Membership: The class $[\in] = \{(x, y) \mid x \in y\}$ exists.

Axiom of Diagonal: The class $[=] = \{(x, y) \mid x = y\}$ exists. This axiom can be used to build a relation asserting the equality of any two of its arguments and consequently used to handle repeated variables.

With the above finite axiomatization, the NBG theory can be adopted as a set theoretical basis for GrC. Such a formal framework prompts a powerful insight into the essence of granulation namely that **the granulation process transforms the semantics of the granulated entities**, mirroring the semantical distinction between sets and classes.

The semantics of granules is derived from the domain that has, in general, higher cardinality than the cardinality of the granulated sets. Although, at first, it might be a bit surprising to see that such a semantical transformation is an essential part of information granulation, in fact, we can point to a common framework of many scientific disciplines which have evolved by abstracting from details inherent to the underpinning scientific discipline and developing a vocabulary of terms (proper classes) that have been verified by the reference to real-life (ultimately to the laws of physics). An example of granulation of detailed information into semantically meaningful granules might be the consideration of cells and organisms in Biology rather than consideration of molecules, atoms or subatomic particles when studying the physiology of living organisms.

The operation on classes in NBG is entirely consistent with the operation on sets in the intuitive set theory. The principle of abstraction implies that classes can be formed out of any statement of the predicate calculus, with the membership relation. Notions of equality, pairing and such, are thus matters of definitions (a specific abstraction of a formula) and not of axioms. In NBG, a set represents a class if every element of the set is an element of the class. Consequently, there are classes that do not have representations.

We suggest therefore that the advantage of adopting NBG as a set theoretical basis for GrC is that it provides a framework within which one can discuss a hierarchy of different granulations without running the risk of inconsistency. For instance, one can denote a “large category” as a category of granules whose collection and collection of morphisms can be represented by a class. A “small category” can be denoted as a category of granules contained in sets. Thus, we can speak of “category of all small categories” (which is a “large category”) without the risk of inconsistency.

A similar framework for a set-theoretical representation of granulation is offered by the theory of types published by Russell in 1937 (Russell, 1937). The theory assumes a

linear hierarchy of types: with type 0 consisting of objects of undecided type and, for each natural number n , type $n+1$ objects are sets of type n objects. The conclusions that can be drawn from this framework with respect to the nature of granulation are exactly the same as that drawn from the NBG.

2.2.3. Mereology

An alternative framework for the formalization of GrC, that of mereology, has been proposed by other researchers. The roots of mereology can be traced to the work of Edmund Husserl (Husserl, 1901) and to the subsequent work of Polish mathematician, Stanislaw Lesniewski, in the late 1920s (Lesniewski, 1929a, 1929b). Much of this work was motivated by the same concerns about the intuitive set theory that have spurred the development of axiomatic set theories (ZF, NBG and others) (Goedel, 1940; Zermelo, 1908).

Mereology replaces talk about “sets” with talk about “sums” of objects, objects being no more than the various things that make up wholes. However, such a simple replacement results in an “intuitive mereology” that is analogous to “intuitive set theory”. Such “intuitive mereology” suffers from paradoxes analogous to Russell’s paradox (we can ask: If there is an object whose parts are all the objects that are not parts of themselves; is it a part of itself?). So, one has to conclude that the mere introduction of the mereological concept of “partness” and “wholeness” is not sufficient and that mereology requires axiomatic formulation.

Axiomatic formulation of mereology has been proposed as a first-order theory whose universe of discourse consists of *wholes* and their respective *parts*, collectively called objects (Simons, 1987; Tarski, 1983). A mereological system requires at least one primitive relation, e.g., dyadic Parthood, x is a *part of* y , written as Pxy . Parthood is nearly always assumed to partially order the universe. An immediate defined predicate is x is a *proper part of* y , written $PPxy$, which holds if Pxy is true and Pyx is false. An object lacking proper parts is an *atom*. The mereological universe consists of all objects we wish to consider and all of their proper parts. Two other predicates commonly defined in mereology are Overlap and Underlap. These are defined as follows:

- Oxy is an overlap of x and y if there exists an object z such that Pzx and Pzy both hold.
- Uxy is an underlap of x and y if there exists an object z such that x and y are both parts of z (Pxz and Pyz hold).

With the above predicates, axiomatic mereology defines the following axioms:

M1, Parthood is Reflexive: Asserts that object is part of itself.

M2, Parthood is Antisymmetric: Asserts that if Pxy and Pyx both hold, then x and y are the same object.

M3, Parthood is Transitive: Asserts that if Pxy and Pyz hold then Pxz hold.

M4, Weak Supplementation: Asserts that if $PPxy$ holds, there exists z such that Pzy holds but Ozx does not.

M5, Strong Supplementation: Asserts that if Pyx does not holds, there exists z such that Pzy holds but Ozx does not.

M5a, Atomistic Supplementation: Asserts that if Pxy does not hold, then there exists an atom z such that Pzx holds but Ozy does not.

Top: Asserts that there exists a “universal object”, designated W , such that PxW holds for any x .

Bottom: Asserts that there exists an atomic “null object”, designated N , such that PNx hold for any x .

M6, Sum: Asserts that if Uxy holds, there exists z , called the “sum of x and y ”, such that the parts of z are just those objects which are parts of either x or y .

M7, Product: Asserts that if Oxy holds, there exists z , called the “Product of x and y ”, such that the parts of z are just those objects which are parts of both x and y .

M8, Unrestricted Fusion: Let f be a first order formula having one free variable. Then the fusion of all objects satisfying f exists.

M9, Atomicity: Asserts that all objects are either atoms or fusions of atoms.

It is clear that if “parthood” in mereology is taken as corresponding to “subset” in set theory, there is some analogy between the above axioms of *classical extensional mereology* and those of standard ZF set theory. However, there are some philosophical and common sense objections to some of the above axioms; e.g., transitivity of Parthood (*M3*). Also, the set of above axioms is not minimal since it is possible to derive Weak Supplementation axiom (*M4*) from Strong Supplementation axiom (*M5*).

Axiom *M6* implies that if the universe is finite or if *Top* is assumed, then the universe is closed under sum. Universal closure of product and of supplementation relative to W requires *Bottom*. W and N are evidently the mereological equivalents of the universal and the null sets. Because sum and product are binary operations, *M6* and *M7* admit the sum and product of only a finite number of objects. The *fusion* axiom, *M8*, enables taking the sum of infinitely many objects. The same holds for product. If *M8* holds, then W exists for infinite universes. Hence, *Top* needs to be assumed only if the universe is infinite and *M8* does not hold. It is somewhat strange that while the *Top* axiom (postulating W) is not controversial, the *Bottom* axiom (postulating N) is. Lesniewski rejected *Bottom* axiom and most mereological systems follow his example. Hence, while the universe is closed under sum, the product of objects that do not overlap is typically undefined. Such defined mereology is equivalent to Boolean algebra lacking a 0. Postulating N generates

mereology in which all possible products are definable but it also transforms extensional mereology into a Boolean algebra without a null-element (Tarski, 1983).

The full mathematical analysis of the theories of parthood is beyond the intended scope of this chapter and the reader is referred to the recent publication by Pontow and Schubert (2006) in which the authors prove, by set theoretical means, that there exists a model of general extensional mereology where arbitrary summation of attributes is not possible. However, it is clear from the axiomatization above that the question about the existence of a universal entity containing all other entities and the question about the existence of an empty entity as part of all existing entities are answered very differently by set theory and mereology. In set theory, the existence of a universal entity is contradictory and the existence of an empty set is mandatory, while in mereology the existence of a universal set is stipulated by the respective fusion axioms and the existence of an empty entity is denied. Also, it is worth noting that in mereology there is no straightforward analog to the set theoretical is-element-of relation (Pontow and Schubert, 2006).

So taking into account the above, we suggest the following answer to the underlying questions of this section: Why granulation is necessary? Why the set-theoretical representation of granulation is appropriate?

- The **concept of granulation is necessary** to denote the semantical transformation of granulated entities in a way that is analogous to semantical transformation of sets into classes in axiomatic set theory;
- Granulation interpreted in the context of axiomatic set theory is **very different from clustering**, since it deals with semantical transformation of data and not limits itself to a mere grouping of similar entities; and
- The set-theoretical interpretation of granulation enables **consistent** representation of a hierarchy of information granules.

2.3. Abstraction and Computation

Having established an argument for semantical dimension to granulation, one may ask; how is the meaning (semantics) instilled into real-life information granules? Is the meaning instilled through an algorithmic processing of constituent entities or is it a feature that is independent of algorithmic processing?

The answers to these questions are hinted by von Neumann's *limitation of size principle*, mentioned in the previous section, and are more fully informed by Turing's theoretical model of computation. In his original paper, Turing (1936), he has defined computation as an automatic version of doing what people do when they manipulate numbers and symbols on paper. He proposed a conceptual model which included: (a) an arbitrarily long tape from which one could read as many symbols as needed (from a countable set); (b) means to read and write those symbols; (c) a countable set of states storing information about the completed processing of symbols; and (d) a countable set of rules that governed what should be done for various combinations of input and system state. A physical instantiation of computation, envisaged by Turing, was a human operator (called computer) who was compelled to obey the rules (a)–(d) above. There are several important implications of Turing's definition of computation. First, the model implies that computation explores only a subset of capabilities of human information processing. Second, the constraint that the input and output is strictly symbolic (with symbols drawn from a countable set) implies that the computer does not interact directly with the environment. These are critical limitations meaning that Turing's computer on its own is unable (by definition) to respond to external, physical stimuli. *Consequently, it is not just wrong but essentially meaningless to speculate on the ability of Turing machines to perform human-like intelligent interactions with the real world.*

To phrase it in mathematical terms, the general form of computation, formalized as a Universal Turing Machine (UTM), is defined as mapping of sets that have at most cardinality N_0 (infinite, countable) onto sets with cardinality N_0 . The practical instances of information processing, such as clustering of data, typically involve a finite number of elements both in the input and output sets and represent therefore a more manageable mapping of a finite set with cardinality \max_1 onto another finite set with cardinality \max_2 . The hierarchy of computable clustering can be therefore represented as in [Figure 2.1](#).

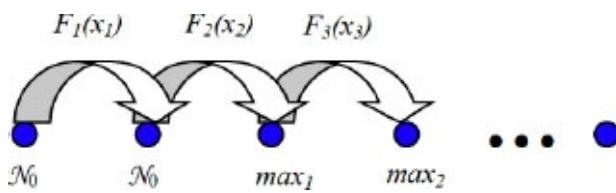


Figure 2.1: Cardinality of sets in a hierarchy of clustering implemented on UTM.

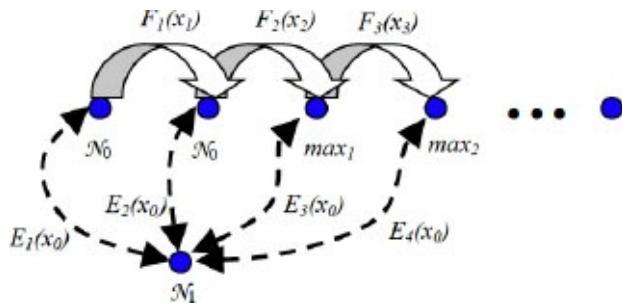
The functions $F_1(x_1) \rightarrow x_2$, $F_2(x_2) \rightarrow x_3$, $F_3(x_3) \rightarrow x_4$ represent mappings of
— infinite (countable) input set onto infinite (countable) output set;

- infinite (countable) input set onto finite output set; and
- finite input set onto finite output set, respectively.

The functional mappings, deployed in the process of clustering, reflect the criteria of similarity, proximity or indistinguishability of elements in the input set and, on this basis, grouping them together into a separate entity to be placed in the output set. In other words, the functional mappings generate data abstractions on the basis of pre-defined criteria and consequently represent UTM computation. However, we need to understand how these criteria are selected and how they are decided to be appropriate in any specific circumstance. Clearly, there are many ways of defining similarity, proximity or indistinguishability. Some of these definitions are likely to have good real-world interpretation, while others may be difficult to interpret or indeed may lead to physically meaningless results.

We suggest that the process of instilling the real-world interpretation into data structures generated by functional mappings $F_1(x_1) \rightarrow x_2$, $F_2(x_2) \rightarrow x_3$, $F_3(x_3) \rightarrow x_4$, involves reference to the real-world, as illustrated in [Figure 2.2](#). This is represented as execution of “experimentation” functions $E_*(x_0)$. These functions map the real-world domain x_0 , which has cardinality N_1 (infinite, continuum), onto sets x_1 , x_2 , x_3 , x_4 , respectively.

At this point, it is important to underline that the experimentation functions $E_1(x_0) \rightarrow x_1$, $E_2(x_0) \rightarrow x_2$, $E_3(x_0) \rightarrow x_3$, $E_4(x_0) \rightarrow x_4$, are not computational, in UTM sense, because their domain have cardinality N_1 . So, the process of defining the criteria for data clustering, and implicitly instilling the meaning into information granules, relies on the laws of physics and not on the mathematical model of computation. Furthermore, the results of experimentation do not depend on whether the experimenter understands or is even aware of the laws of physics. It is precisely because of the fact that we consider the experimentation functions as providing objective evidence.



[Figure 2.2:](#) Mapping of abstractions from the real-world domain (cardinality N_1) onto the sets of clusters.

2.4. Experimentation as a Physical Computation

Recent research (Siegelmann, 1999) has demonstrated that analog computation, in the form of recurrent analog neural networks (RANN) can exceed the abilities of a UTM, if the weights in such neural networks are allowed to take continuous rather than discrete weights. While this result is significant in itself, it relies on the assumptions about the continuity of parameters that are difficult to verify. So, although the brain looks remarkably like a RANN, drawing any conclusions about the hyper-computational abilities of the brain, purely on the grounds of structural similarities, leads to the same questions about the validity of the assumptions about continuity of weights. Of course, this is not to say that these assumptions are not valid, they may well be valid, but we just highlight that this has not been demonstrated yet in a conclusive way.

A pragmatic approach to bridging the gap between the theoretical model of hyper-computation, as offered by RANN, and the human, intelligent information processing (which by definition is hyper-computational) has been proposed by Bains (2003, 2005). Her suggestion was to reverse the original question about hyper-computational ability of systems and to ask: *if the behavior of physical systems cannot be replicated using Turing machines, how can they be replicated?* The answer to this question is surprisingly simple: *we can use inherent computational ability of physical phenomena in conjunction with the numerical information processing ability of UTM*. In other words, the readiness to refine numerical computations in the light of objective evidence coming from a real-life experiment, instills the ability to overcome limitations of the Turing machine. We have advocated this approach in our earlier work (Bargiela, 2004), and have argued that the hyper-computational power of GrC is equivalent to “keeping open mind” in intelligent, human information processing.

In what follows, we describe the model of physical computation, as proposed in Bains (2003), and cast it in the framework of GrC.

2.4.1. A Model of Physical Computation

We define a system under consideration as an identifiable collection of connected elements. A system is said to be *embodied* if it occupies a definable volume and has a collective contiguous boundary. In particular, a UTM with its collection of input/output (I/O) data, states and collection of rules, implementing some information processing algorithm, can be considered a system G whose physical instantiations may refer to specific I/O, processing and storage devices as well as specific energy states. The matter, space and energy outside the boundaries of the embodied system are collectively called the *physical environment* and will be denoted here by P .

A *sensor* is any part of the system that can be changed by physical influences from the environment. Any forces, fields, energy, matter, etc., that may be impinging on the system, are collectively called the *sensor input* ($i \in X$), even where no explicitly-defined sensors

exist.

An *actuator* is any part of the system that can change the environment. Physical changes to the embodied system that manifest themselves externally (e.g., emission of energy, change of position, etc.) are collectively called the *actuator output* ($h \in Y$) of G . A coupled pair of sensor input i_t and actuator output h_t represents an instance of experimentation at time t and is denoted here as E_t .

Since the system G , considered in this study, is a computational system (modeled by UTM) and since the objective of the evolution of this system is to mimic human intelligent information processing we will define G_t as the *computational intelligence function* performed by the embodied system G . Function G_t maps the I/O at specific time instances t , resolved with arbitrarily small accuracy $\delta t > 0$, so as not to preclude the possibility of a continuous physical time. We can thus formally define the computational intelligence function as

$$G_t(i_t) \rightarrow h_{t+\delta t}.$$

In the proposed physical model of computation, we stipulate that G_t causes only an immediate output in response to an immediate input. This stipulation does not prevent one from implementing some plan over time but it implies that a controller that would be necessary to implement such plan is part of the intelligence function. The adaptation of G_t in response to evolving input i_t can be described by the *computational learning function*, $L_G : L_G(G_t, i_t) \rightarrow G_{t+\delta t}$.

Considering now the impact of the system behavior on the environment we can define the *environment reaction function* mapping system output h (environment input) to environment output i (system input) as

$$P_t(h_t) \rightarrow i_{t+\delta t}$$

The adaptation of the environment P over time can be described by the *environment learning function*, $L_P : L_P(P_t, h_t) \rightarrow P_{t+\delta t}$.

2.4.2. Physical Interaction between P and G

The interaction between the system G and its physical environment P may be considered to fall into one of the two classes: *real interaction* and *virtual interaction*. Real interaction is a pure physical process in which the output from the environment P is in its entirety forwarded as an input to the system G and conversely the output from G is fully utilized as input to P .

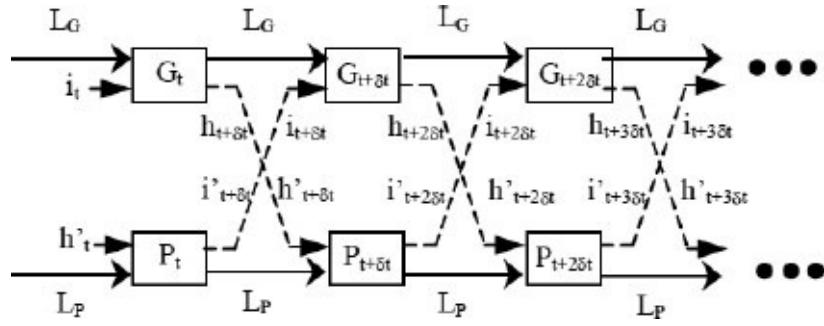


Figure 2.3: Evolution of a system in an experiment with physical interaction.

Referring to the notation in [Figure 2.3](#), real interaction is one in which $h_t = h'_t$ and $i_t = i'_t$ for all time instances t . Unfortunately, this type of interaction does not accept the limitations of the UTM, namely, the processing of only a pre-defined set of symbols rather than a full spectrum of responses from the environment. Consequently, this type of interaction places too high demands on the information processing capabilities of G and, in practical terms, is limited to interaction of physical objects as governed by the laws of physics. In other words the intelligence function and its implementation are one and the same.

2.4.3. Virtual Interaction

An alternative mode of interaction is *virtual interaction*, which is mediated by symbolic representation of information. Here, we use the term *symbol* as it is defined in the context of UTM: a letter or sign taken from a finite alphabet to allow distinguishability.

We define V_t as the *virtual computational intelligence function*, analogous to G_t in terms of information processing, and V'_t as the *complementary computational intelligence function*, analogous to G_t in terms of communication with the physical environment. With the above definitions, we can lift some major constraints of physical interactions, with important consequences. The complementary function V'_t can implement an interface to the environment, filtering real-life information input from the environment and facilitating transfer of actuator output, while the virtual intelligence function V_t can implement UTM processing of the filtered information. This means that i_t does not need to be equal to i'_t and h_t does not need to be equal to h'_t . In other words, I/O may be considered selectively rather than in their totality. The implication being that many physically distinguishable states may have the same symbolic representation at the virtual computational intelligence function level. The relationship between the two components of the computational intelligence is illustrated in [Figure 2.4](#).

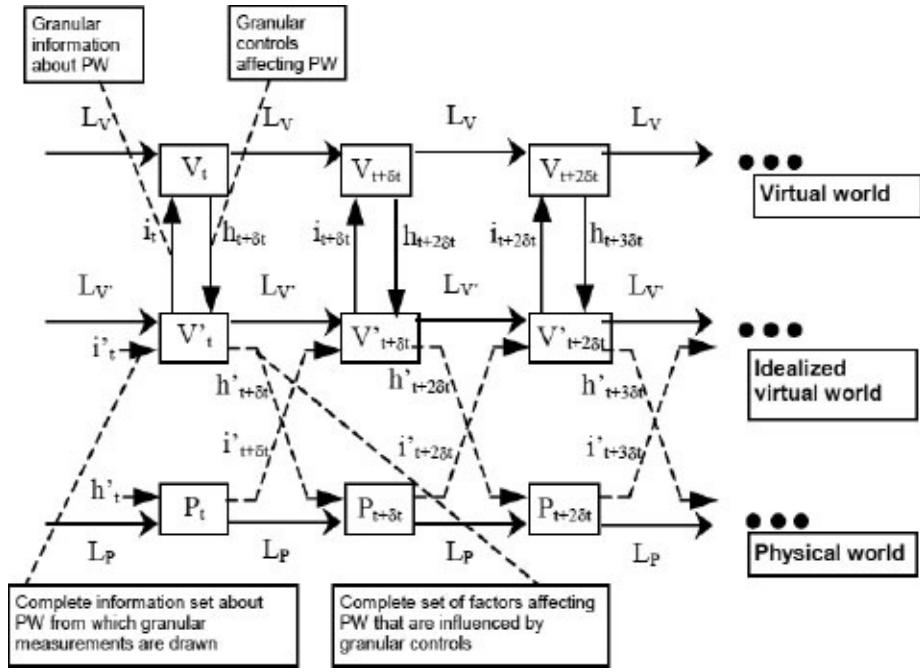


Figure 2.4: Evolution of a system in an experiment with virtual interaction.

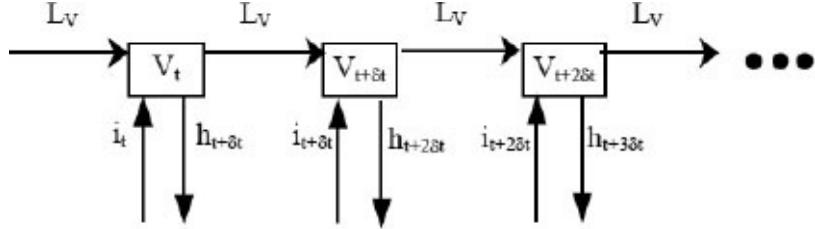


Figure 2.5: The paradigm of computing with perceptions within the framework of virtual interaction.

It should be pointed out that typically we think of the complementary function V'_t as some mechanical or electronic device (utilizing the laws of physics in its interaction with the environment) but a broader interpretation that includes human perception, as discussed by Zadeh (1997), is entirely consistent with the above model. In this broader context, the UTM implementing the virtual computational intelligence function can be referred to as *computing with perceptions* or *computing with words* (see [Figure 2.5](#)).

Another important implication of the virtual interaction model is that V and P need not have any kind of conserved relationship. This is because only range/modality of subsets of i'_t and h'_t attach to V and these subsets are defined by the choice of sensor/actuator modalities. So, we can focus on the choice of modalities, within the complementary computational intelligence function, as a mechanism through which one can exercise the assignment of semantics to both I/O of the virtual intelligence function. To put it informally, the complementary function is a facility for defining a “language” in which we chose to communicate with the real world.

Of course, to make the optimal choice (one that allows undistorted perception and interaction with the physical environment), it would be necessary to have a complete knowledge of the physical environment. So, in its very nature the process of defining the semantics of I/O of the virtual intelligence function is iterative and involves evolution of our understanding of the physical environment.

2.5. Granular Computation

An important conclusion from the discussion above is that the discovery of semantics of information abstraction, referred to sometimes as structured thinking, or a philosophical dimension of GrC, can be reduced to physical experimentation. This is a very welcome development as it gives a good basis for the formalization of the GrC paradigm.

We argue here that GrC should be defined as a **structured combination** of algorithmic abstraction of data and non-algorithmic, empirical verification of the semantics of these abstractions. This definition is general in that it neither prescribes the mechanism of algorithmic abstraction nor it elaborates on the techniques of experimental verification. Instead, it highlights the essence of combining computational and non-computational information processing. Such a definition has several advantages:

- it emphasizes the complementarity of the two constituent functional mappings;
- it justifies the hyper-computational nature of GrC;
- it places physics alongside set theory as the theoretical foundations of GrC;
- it helps to avoid confusion between GrC and purely algorithmic data processing while taking full advantage of the advances in algorithmic data processing.

2.6. An Example of Granular Computation

We illustrate here an application of the granular computation, cast in the formalism of set theory, to a practical problem of analyzing traffic queues. A three-way intersection is represented in [Figure 2.7](#). The three lane-occupancy detectors (inductive loops), labeled here as “east”, “west” and “south” provide counts of vehicles passing over them. The counts are then integrated to yield a measure of traffic queues on the corresponding approaches to the junction. A representative sample of the resulting three-dimensional time series of traffic queues is illustrated in [Figure 2.8](#).

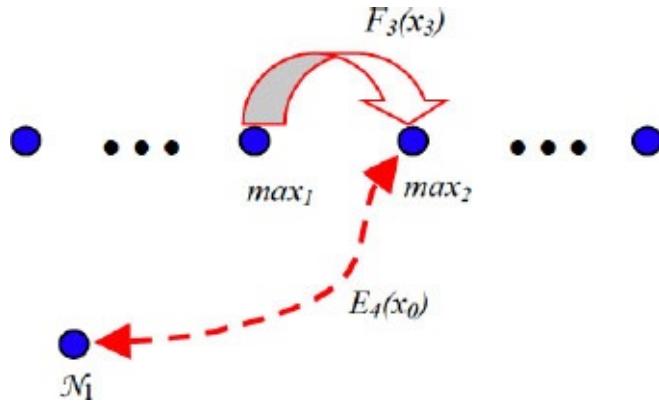


Figure 2.6: An instance of GrC involving two essential components: algorithmic clustering and empirical evaluation of granules.

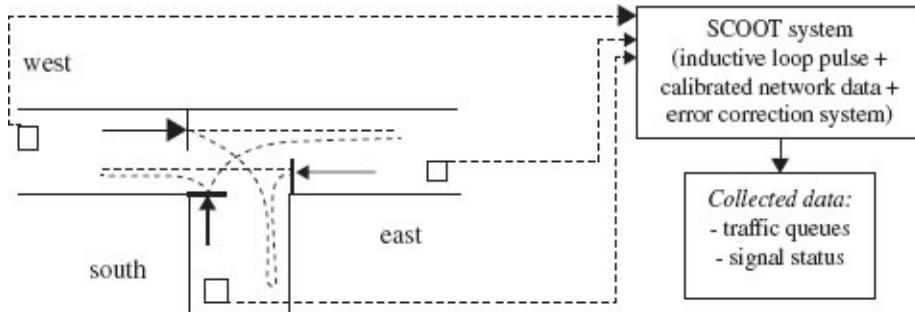


Figure 2.7: A three-way intersection with measured traffic queues.

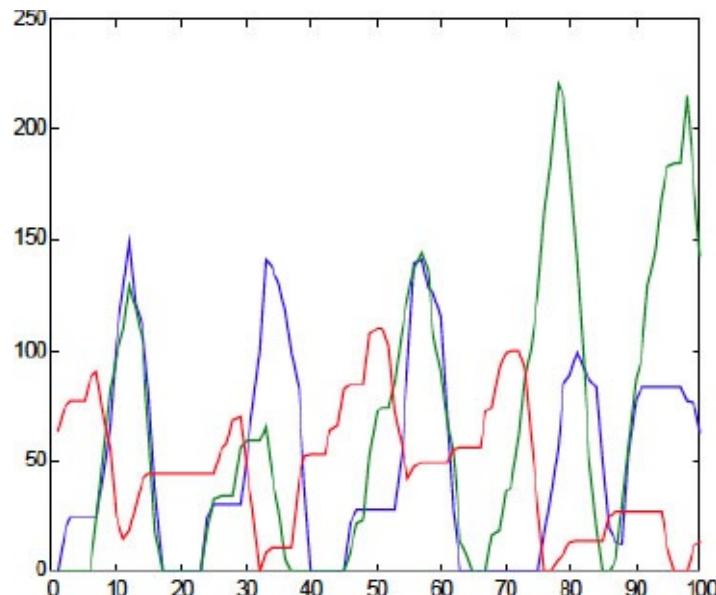


Figure 2.8: A subset of 100 readings from the time series of traffic queues data.

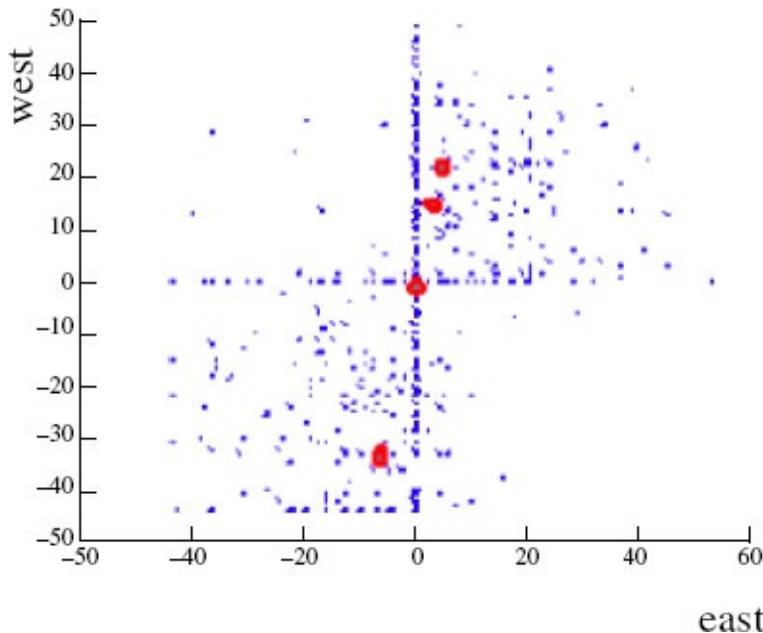


Figure 2.9: FCM prototypes as subset of the original measurements of traffic queues.

It is quite clear that, on its own, data depicted in [Figure 2.8](#) reflects primarily the signaling stages of the junction. This view is reinforced, if we plot the traffic queues on a two-dimensional plane and apply some clustering technique [such as Fuzzy C-Means (FCM)] to identify prototypes that are the best (in terms of the given optimality criterion) representation of data. The prototypes, denoted as small circles in [Figure 2.9](#), indicate that the typical operation of the junction involves simultaneously increasing and decreasing queues on the “east” and “west” junction. This of course corresponds to “red” and “green” signaling stages. It is worth emphasizing here that the above prototypes can be considered as a simple subset of the original numerical data since the nature of the prototypes is entirely consistent with that of the original data.

Unfortunately, within this framework, the interpretation of the prototype indicating “zero” queue in both “east” and “west” direction is not very informative. In order to uncover the meaning of this prototype, we resort to a granulated view of data. [Figure 2.10](#) represents traffic queue data that has been granulated based on maximization of information density measure discussed in [Bargiela et al. \(2006\)](#). The semantics of the original readings is now changed from point representation of queues into interval (hyperbox) representation of queues. In terms of set theory, we are dealing here with a class of hyperboxes, which is semantically distinct from point data.

Applying FCM clustering to granular data results in granular prototypes denoted, in [Figure 2.10](#), as rectangles with bold boundaries overlaid on the granulated data. In order to ascertain that the granulation does not distort the essential features of the data, different granulation parameters have been investigated and a representative sample of two granulations is depicted in [Figure 2.10](#). The three FCM prototypes lying in the areas of simultaneous increase and decrease of traffic queues have identical interpretation as the corresponding prototypes in [Figure 2.9](#). However, the central prototype highlights a physical property of traffic that was not captured by the numerical prototype.

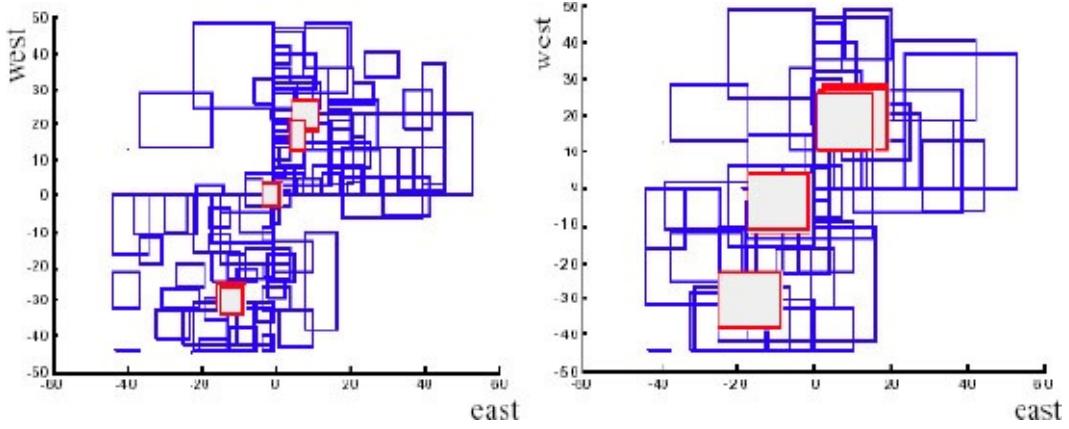


Figure 2.10: Granular FCM prototypes representing a class that is semantically distinct from the original point data.

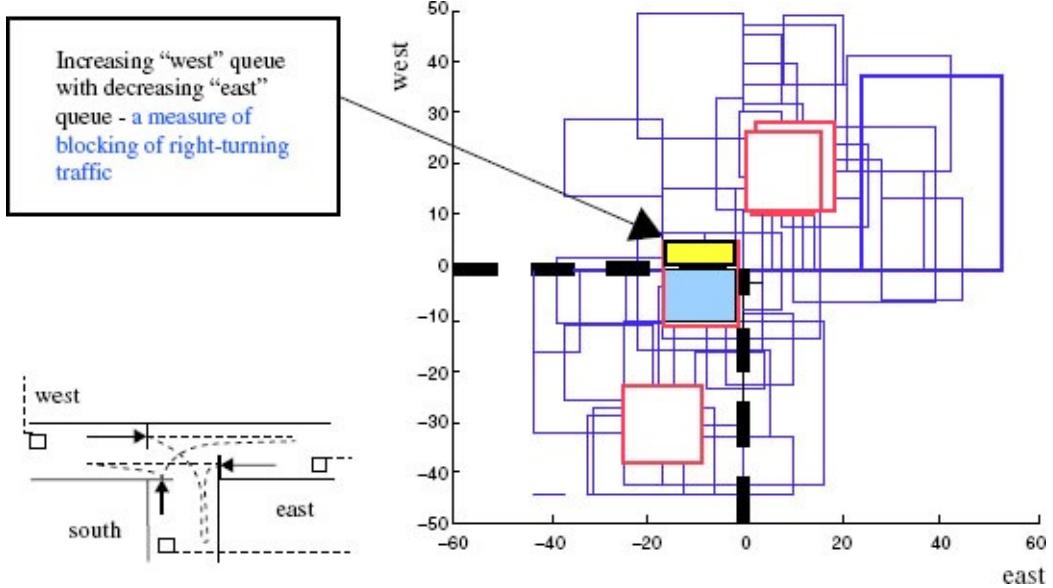


Figure 2.11: Granular prototype capturing traffic delays for “right turning” traffic.

[Figure 2.11](#) illustrates the richer interpretation of the central prototype. It is clear that the traffic queues on the western approach are increasing while the traffic queues on the eastern approach are decreasing. This is caused by the “right turning” traffic being blocked by the oncoming traffic from the eastern junction. It is worth noting that the prototype is unambiguous about the absence of a symmetrical situation where an increase of traffic queues on the eastern junction would occur simultaneously with the decrease of the queues on the western junction. The fact is that this is a three-way junction with no right turn for the traffic from the eastern junction and has been captured purely from the granular interpretation of data. Note that the same cannot be said about the numerical data illustrated in [Figure 2.9](#).

As we have argued in this chapter, the essential component of GrC is the experimental validation of the semantics of the information granules. We have conducted a planned experiment in which we placed counting devices on the entrance to the “south” link. The proportion of the vehicles entering the “south” junction (during the green stage of the “east–west” junction) to the count of vehicles on the stop line on the “west” approach, represents a measure of the right turning traffic. The ratio of these numerical counts was 0.1428. A similar measurement derived from two different granulations depicted in [Figure](#)

[2.10](#) was 0.1437 and 0.1498. We conclude therefore that the granulated data captured the essential characteristics of the right turning traffic and that, in this particular application, the granulation parameters do not affect the result to a significant degree (which is clearly a desirable property).

A more extensive experimentation could involve verification of the granular measurement of right turning traffic for drivers that have different driving styles in terms of acceleration and gap acceptance. Although we do not make any specific claim in this respect, it is possible that the granulation of traffic queue data would need to be parameterized with the dynamic driver behavior data. Such data could be derived by differentiating the traffic queues (measurement of the speed of change of queues) and granulating the resulting six-dimensional data.

References

- Bains, S. (2003). Intelligence as physical computation. *AISBJ*, 1(3), 225–240.
- Bains, S. (2005). Physical computation and embodied artificial intelligence. Ph.D. thesis. The Open University, January, 1–199.
- Bargiela, A. and Pedrycz, W. (2002a). *Granular Computing: An Introduction*. Dordrecht, Netherlands: Kluwer Academic Publishers.
- Bargiela, A. and Pedrycz, W. (2002b). From numbers to information granules: a study of unsupervised learning and feature analysis. In Bunke, H. and Kandel, A. (eds.), *Hybrid Methods in Pattern Recognition*. Singapore: World Scientific, pp. 75–112.
- Bargiela, A. and Pedrycz, W. (2003). Recursive information granulation: aggregation and interpretation issues. *IEEE Trans. Syst. Man Cybern. SMC-B*, 33(1), pp. 96–112.
- Bargiela, A. (2004). Hyper-computational characteristics of granular computing. *First Warsaw Int. Semin. Intell. Syst.-WISIS 2004*, Invited lectures. Warsaw, May, pp. 1–8.
- Bargiela, A., Pedrycz, W. and Tanaka, M. (2004a). An inclusion/exclusion fuzzy hyperbox classifier. *Int. J. Knowl.-Based Intell. Eng. Syst.*, 8(2), pp. 91–98.
- Bargiela, A., Pedrycz, W. and Hirota, K. (2004b). Granular prototyping in fuzzy clustering. *IEEE Trans. Fuzzy Syst.*, 12(5), pp. 697–709.
- Bargiela, A. and Pedrycz, W. (2005a). A model of granular data: a design problem with the Tchebyschev FCM. *Soft Comput.*, 9(3), pp. 155–163.
- Bargiela, A. and Pedrycz, W. (2005b). Granular mappings. *IEEE Trans. Syst. Man Cybern. SMC-A*, 35(2), pp. 288–301.
- Bargiela, A. and Pedrycz, W. (2006). The roots of granular computing. *Proc. IEEE Granular Comput. Conf.*, Atlanta, pp. 741–744. May.
- Bargiela, A., Kosonen, I., Pursula, M. and Peytchev, E. (2006). Granular analysis of traffic data for turning movements estimation. *Int. J. Enterp. Inf. Syst.*, 2(2), pp. 13–27.
- Cantor, G. (1879). Über einen satz aus der theorie der stetigen mannigfaltigkeiten. *Göttinger Nachr.*, pp. 127–135.
- Dubois, D., Prade, H. and Yager, R. (eds.). (1997). *Fuzzy Information Engineering*. New York: Wiley.
- Goedel, K. (1940). *The Consistency of the Axiom of Choice and of the Generalized Continuum Hypothesis with the Axioms of Set Theory*. Princeton, NJ: Princeton University Press.
- Husserl, E. (1901). Logische untersuchungen. *Phanomenologie und theorie der erkenntnis*, 2, pp. 1–759.
- Inuiguchi, M., Hirano, S. and Tsumoto, S. (eds.). (2003). *Rough Set Theory and Granular Computing*. Berlin: Springer.
- Lesniewski, S. (1929a). Über funktionen, deren felder gruppen mit rucksicht auf diese funktionen sind. *Fundamenta Mathematicae*, 13, pp. 319–332.
- Lesniewski, S. (1929b). Grundzuge eines neuen systems der grundlagen der mathematic. *Fundamenta Mathematicae*, 14, pp. 1–81.
- Lin, T. Y. (1998). Granular computing on binary relations. In Polkowski, L. and Skowron, A. (eds.). *Rough Sets in Knowledge Discovery: Methodology and Applications*. Heidelberg, Germany: Physica-Verlag, pp. 286–318.
- Lin, T. Y., Yao, Y. Y. and Zadeh, L. A. (eds.). (2002). *Data Mining, Rough Sets and Granular Computing*, Heidelberg, Germany: Physica-Verlag.
- Pawlak, Z. (1991). *Rough Sets: Theoretical Aspects of Reasoning about Data*. Dordrecht, Netherlands: Kluwer Academic Publishers.
- Pawlak, Z. (1999). Granularity of knowledge, indiscernibility and rough sets. *Proc. IEEE Conf. Evolutionary Comput.*, Anchorage, Alaska, pp. 106–110.
- Pedrycz, W. (1989). *Fuzzy Control and Fuzzy Systems*. New York: Wiley.
- Pedrycz, W. and Gomide, F. (1998). *An Introduction to Fuzzy Sets*. Cambridge, MA: MIT Press.
- Pedrycz, W., Smith, M. H. and Bargiela, A. (2000). Granular clustering: a granular signature of data. *Proc. 19th Int.*

(IEEE) Conf. NAFIPS'2000. Atlanta, pp. 69–73.

- Pedrycz, W. and Bargiela, A. (2002). Granular clustering: a granular signature of data. *IEEE Trans. Syst. Man Cybern.*, 32(2), pp. 212–224.
- Pontow, C. and Schubert, R. (2006). A mathematical analysis of parthood. *Data Knowl. Eng.*, 59(1), pp. 107–138.
- Russell, B. (1937). New foundations for mathematical logic. *Am. Math. Mon.*, 44(2), pp. 70–80.
- Siegelmann, H. (1999). *Neural Network and Analogue Computation: Beyond the Turing limit*. Boston, MA: Birkhauser.
- Simons, P. (1987). *Parts: A Study in Ontology*. Oxford, UK: Oxford University Press.
- Skowron, A. and Stepaniuk, J. (2001). Information granules: towards foundations of granular computing. *Int. J. Intell. Syst.*, 16, pp. 57–85.
- Tarski, A. (1983). Foundations of the geometry of solids. In *Logic, Semantics, Metamathematics*: Indianapolis: Hackett.
- Turing, A. (1936). On computable numbers, with an application to the entscheidungs problem. *Proc. London Math. Soc.*, 42, pp. 230–265.
- Yao, Y. Y. and Yao, J. T. (2002). Granular computing as a basis for consistent classification problems. *Proc. PAKDD'02 Workshop on Found. Data Min.*, pp. 101–106.
- Yao, Y. Y. (2004a). Granular computing. *Proc. Fourth Chin. National Conf. Rough Sets Soft Comput. Sci.*, 31, pp. 1–5.
- Yao, Y. Y. (2004b). A partition model of granular computing. *LNCS Trans. Rough Sets*, 1, pp. 232–253.
- Yao, Y. Y. (2005). Perspectives on granular computing. *Proc. IEEE Conf. Granular Comput.*, 1, pp. 85–90.
- Zadeh, L. A. (1965). Fuzzy sets. *Inf. Control*, 8, pp. 338–353.
- Zadeh, L. A. (1979). Fuzzy sets and information granularity. In Gupta, N., Ragade, R. and Yager, R. (eds.), *Advances in Fuzzy Set Theory and Applications*. Amsterdam: North-Holland Publishing Company.
- Zadeh, L. A. (1997). Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. *Fuzzy Sets Syst.*, 90, pp. 111–127.
- Zadeh, L. A. (2002). From computing with numbers to computing with words—from manipulation of measurements to manipulation of perceptions. *Int. J. Appl. Math. Comput. Sci.*, 12(3), pp. 307–324.
- Zermelo, E. (1908). Untersuchungen ueber die grundlagen der mengenlehre. *Math. Annalen*, 65, pp. 261–281.

Chapter 3

Evolving Fuzzy Systems—Fundamentals, Reliability, Interpretability, Useability, Applications

Edwin Lughofer

This chapter provides a round picture of the development and advances in the field of *evolving fuzzy systems* (EFS) made during the last decade since their first appearance in 2002. Their basic difference to conventional fuzzy systems (discussed in other chapters in this book) is that they can be learned from data on-the-fly (fast) during online processes in an incremental and mostly single-pass manner. Therefore, they stand for a very emerging topic in the field of soft computing for addressing modeling problems in the quickly increasing complexity of real-world applications, more and more implying a shift from batch offline model design phases (as conducted since the 80s) to permanent online (active) model teaching and adaptation. The focus will be placed on the definition of various model architectures used in the context of EFS, on providing an overview about the basic learning concepts, on listing the most prominent EFS approaches (fundamentals), and on discussing advanced aspects toward an improved stability, reliability and useability (usually must-to-haves to guarantee robustness and user-friendliness) as well as an educated interpretability (usually a nice-to-have to offer insights into the systems' nature). It will be concluded with a list of real-world applications where various EFS approaches have been successfully applied with satisfactory accuracy, robustness and speed.

3.1. Introduction—Motivation

Due to the increasing complexity and permanent growth of data acquisition sites in today's industrial systems, there is an increasing demand of fast modeling algorithms from online data streams (Gama, 2010). Such algorithms are ensuring that models can be quickly adapted to the actual system situation and thus are able to provide reliable outputs at any time during online real-world processes. Their changing operating conditions, environmental influences and new unexplored system states may trigger quite a dynamic behavior, causing previously trained models to become inefficient or even inaccurate (Sayed-Mouchaweh and Lughofe, 2012). In this sense, conventional static models which are trained once in an offline stage and are not able to adapt dynamically to the actual system states are not an adequate alternative for coping with these demands (severe downtrends in accuracy have been examined in previous publications). A list of potential real-world application examples relying on online dynamic aspects and thus demanding flexible modeling techniques can be found in [Table 3.3 \(Section 3.6.5\)](#).

Another challenge which has recently become a very hot topic within the machine learning community and is given specific attention in the new European framework programme Horizon 2020, is the processing and mining of the so-called *Big Data*,¹ usually stemming from very large databases (VLDB).² The occurrence of *Big Data* takes place in many areas such as meteorology, genomics, connectomics, complex physics simulations and biological and environmental research (Reichmann *et al.*, 2011). This data is that big (exabytes) such that it cannot be handled in a one-shot experience, such as exceeding virtual memory of nowadays conventional computers. Thus, standard batch modeling techniques are not applicable.

In order to tackle the aforementioned requirements, the field of “evolving intelligent systems (EISs)”³ or, in a wider machine learning sense, the field of “learning in dynamic environments (LDE)” enjoyed increasing attraction during the last years (Angelov *et al.*, 2010). This even lead to the emergence of their own journal in 2010, termed as “Evolving Systems” at Springer (Heidelberg).⁴ Both fields support learning topologies which operate in single-pass manner and are able to update models and surrogate statistics on-the-fly and on demand. Single-pass nature and incrementality of the updates assure online and in most cases even real-time learning and model training capabilities. While EIS focuses mainly on adaptive evolving models within the field of soft computing, LDE goes a step further and also joins incremental machine learning and data mining techniques, originally stemming from the area of “incremental heuristic search”.⁵ The update in these approaches concerns both parameter adaptation and structural changes depending on the degree of change required. The structural changes are usually enforced by evolution and pruning components, and finally responsible for the terminus *Evolving Systems*. In this context, *Evolving* should be not confused with *Evolutionary* (as sometimes happened in the past, unfortunately). Evolutionary approaches are usually applied in the context of

complex optimization problems to learn parameters and structures based on genetic operators, but they do this by using all the data in an iterative optimization procedure rather than integrating new knowledge permanently on-the-fly.

Apart from the requirements and demands in industrial (production and control) systems, another important aspect about evolving models is that they provide the opportunity for self-learning computer systems and machines. In fact, evolving models are permanently updating their knowledge and understanding about diverse complex relationships and dependencies in real-world application scenarios by integrating new system behaviors and environmental influences (which are manifested in the captured data). Their learning follows a life-long learning context and is never really terminated, but lasts as long as new information arrives. Therefore, they can be seen as a valuable contribution within the field of computational intelligence (Angelov and Kasabov, 2005) or even in artificial intelligence (Lughofer, 2011a).

There are several possibilities for using an adequate model architecture within the context of an evolving system. This strongly depends on the learning problem at hand, in particular, whether it is supervised or unsupervised. In case of the later, techniques from the field of clustering, usually termed as *incremental clustering* (Bouchachia, 2011) are a prominent choice. In case of classification and regression models, the architectures should support decision boundaries respectively approximation surfaces with an arbitrary nonlinearity degree. Also, the choice may depend on past experience with some machine learning and data mining tools: for instance, it is well-known that SVMs are usually among the top-10 performers for many classification tasks (Wu *et al.*, 2006), thus are a reasonable choice to be used in an online classification setting as well [in form of incremental SVMs (Diehl and Cauwenberghs, 2003; Shilton *et al.*, 2005)]; whereas in a regression setting they are usually performing much weaker. Soft computing models such as neural networks (Haykin, 1999), fuzzy systems (Pedrycz and Gomide, 2007) or genetic programming (Affenzeller *et al.*, 2009) and any hybrid concepts of these [e.g., neuro-fuzzy systems (Jang, 1993)] are all known to be universal approximators (Balas *et al.*, 2009) and thus able to resolve nonlinearities implicitly contained in the systems' behavior (and thus reflected in the data streams). Neural networks suffer from their black box nature, i.e., not allowing operators and users any insight into the models extracted from the streams. This may be essential in many contexts for the interpretation of model outputs to realize why certain decisions have been made etc. Genetic programming is a more promising choice in this direction, however, it is often expanding unnecessarily complex formulas with many nested functional terms [suffering from the so-called bloating effect (Zavoianu, 2010)], which are again hard to interpret.

Fuzzy systems are specific mathematical models which build upon the concept of fuzzy logic, firstly introduced in 1965 by Lotfi A. Zadeh (Zadeh, 1965), are a very useful alternative, as they contain rules which are linguistically readable and interpretable. This mimicks the human thinking about relationships and structural dependencies being present

in a system. This will become more clear in the subsequent section when mathematically defining possible architecture variants within the field of fuzzy systems, which have been also used in the context of data stream mining and evolving systems. Furthermore, the reader may refer to [Chapter 1](#) in this book, where the basic concepts of fuzzy sets and systems are introduced and described in detail.

3.2. Architectures for Evolving Fuzzy Systems (EFSs)

The first five subsections are dedicated to architectures for regression problems, for which EFS have been preliminary used. Then, various variants of fuzzy classification model structures are discussed, as have been recently introduced for representing of decision boundaries in various forms in evolving fuzzy classifiers (EFCs).

3.2.1. Mamdani

Mamdani fuzzy systems (Mamdani, 1977) are the most common choice for coding expert knowledge/experience into a rule-based IF-THEN form, examples can be found in Holmblad and Ostergaard (1982); Leondes (1998) or Carr and Tah (2001); Reveiz and Len (2010).

In general, assuming p input variables (features), the definition of the i th rule in a single output Mamdani fuzzy system is as follows:

$$\text{Rule}_i : \text{IF } (x_1 \text{ IS } \mu_{i1}) \text{ AND } \dots \text{ AND } (x_p \text{ IS } \mu_{ip}) \text{ THEN } l_i(\vec{x}) \text{ IS } \Phi_i,$$

with Φ_i the consequent fuzzy set in the fuzzy partition of the output variable used in the consequent $l_i(\vec{x})$ of the i th rule, and $\mu_{i1}, \dots, \mu_{ip}$ are the fuzzy sets appearing in the rule antecedents. The *rule firing degree* (also called *rule activation level*) for a concrete input vector $\vec{x} = (x_1, \dots, x_p)$ is then defined by:

$$\mu_i(\vec{x}) = \mathsf{T}_{j=1}^p \mu_{ij}(x_j), \quad (1)$$

with T a specific conjunction operator, denoted as t -norm Klement *et al.* (2000)—most frequently, minimum or product are used, i.e.,

$$\mu_i(\vec{x}) = \min_{j=1}^p (\mu_{ij}(x_j)) \quad \mu_i(\vec{x}) = \prod_{j=1}^p (\mu_{ij}(x_j)). \quad (2)$$

It may happen that $\Phi_i = \Phi_j$ for some $i \neq j$. Hence, a t -conorm (Klement *et al.*, 2000) is applied which combines the rule firing levels of those rules having the same consequents to one output set. The most common choice for the t -conorm is the maximum operator. In this case, the consequent fuzzy set is cut at the alpha-level:

$$\alpha_i = \max_{j_i=1, \dots, C_i} (\mu_{j_i}(\vec{x})), \quad (3)$$

with C_i the number of rules whose consequent fuzzy set is the same as for Rule i , and j_i the indices of these rules. This is done for the whole fuzzy rule-base and the various α -cut output sets are joined to one fuzzy area employing the *supremum operator*. An example of such a fuzzy output area is shown in [Figure 3.1](#).

In order to obtain a crisp output value, a defuzzification method is applied, most commonly used are the mean of maximum (MOM) over the whole area, the center of gravity (COG) or the bisector (Leekwijck and Kerre, 1999) which is the vertical line that will divide the whole area into two sub-regions of equal areas. For concrete formulas of the defuzzification operators, please refer to Piegat (2001) and Nguyen *et al.* (1995). MOM and COG are exemplarily shown in [Figure 3.1](#).

Due to the defuzzification process, it is quite intuitive that the inference in Mamdani fuzzy systems loses some accuracy, as an output fuzzy number is reduced to a crisp number. Therefore, they have been hardly applied within the context of online modeling and data stream mining, where the main purpose is to obtain an accurate evolving fuzzy model in the context of *precise evolving fuzzy modeling* [an exception can be found in Rubio (2009), termed as the SOFMLS approach]. On the other hand, they are able to provide linguistic interpretability on the output level, thus may be preferable in the context of interpretability demands for knowledge gaining and reasoning (see also [Section 3.5](#)). The approach in Ho *et al.* (2010), tries to benefit from this while keeping the precise modeling spirit by applying a switched architecture, joining Mamdani and Takagi–Sugeno type consequents in form of a convex combination.

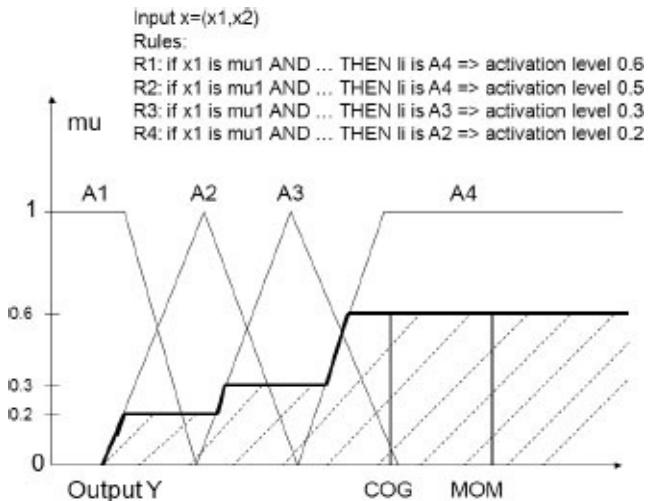


Figure 3.1: MOM and COG defuzzification for a rule consequent partition (fuzzy partition in output variable Y) in a Mamdani fuzzy system, the shaded area indicates the joint consequents (fuzzy sets) in the active rules (applying supremum operator); the cutoff points (alpha cuts) are according to maximal membership degrees obtained from the rule antecedent parts of the active rules.

3.2.2. Takagi–Sugeno

Opposed to Mamdani fuzzy systems, Takagi–Sugeno (TS) fuzzy systems (Takagi and Sugeno, 1985) are the most common architectorial choices in evolving fuzzy systems approaches (Lughofer, 2011b). This has several reasons. First of all, they enjoy a large attraction in many fields of real-world applications and systems engineering (Pedrycz and Gomide, 2007), ranging from process control (Babuska, 1998; Karer and Skrjanc, 2013; Piegat, 2001), system identification (Abonyi, 2003; Nelles, 2001), through condition monitoring (Serdio *et al.*, 2014a, 2014b) and chemometric calibration (Cernuda *et al.*,

2013; Skrjanc, 2009) to machine vision and texture processing approaches (Lughofer, 2011b; Riaz and Ghafoor, 2013). Thus, their robustness and applicability for the standard batch modeling case has been already proven since several decades. Second, they are known to be universal approximators (Castro and Delgado, 1996), i.e., being able to model any implicitly contained nonlinearity with a sufficient degree of accuracy, while their interpretable capabilities are still intact or may offer even advantages: while the antecedent parts remain linguistic, the consequent parts can be interpreted either in a more physical sense (see Bikdash, 1999; Herrera *et al.*, 2005) or as local functional tendencies (Lughofer, 2013) (see also [Section 3.5](#)). Finally, parts of their architecture (the consequents) can be updated exactly by recursive procedures, as will be described in [Section 3.3.1](#). This is a strong point as they are converging to the same solution as when (hypothetically) sending all data samples at once into the optimization process (*true optimal incremental solutions*).

3.2.2.1. Takagi–Sugeno standard

A single rule in a (single output) standard TS fuzzy system is of the form

$$\text{Rule } i : \text{IF } (x_1 \text{ IS } \mu_{i1}) \text{ AND } \dots \text{ AND } (x_p \text{ IS } \mu_{ip}), \quad (4)$$

$$\text{THEN } l_i(\vec{x}) = w_{i0} + w_{i1}x_1 + w_{i2}x_2 + \dots + w_{ip}x_p, \quad (5)$$

where $\vec{x} = (x_1, \dots, x_p)$ is the p -dimensional input vector and μ_{ij} the fuzzy set describing the j -th antecedent of the rule. Typically, these fuzzy sets are associated with a linguistic label. As in case of Mamdani fuzzy systems, the AND connective is modeled in terms of a t-norm, i.e., a generalized logical conjunction (Klement *et al.*, 2000). Again, the output $l_i = l_i(\vec{x})$ is the so-called consequent function of the rule.

The output of a TS system consisting of C rules is a linear combination of the outputs produced by the individual rules (through the l_i 's), where the contribution of each rule is given by its normalized degree of activation Ψ_i , thus:

$$\hat{f}(\vec{x}) = \hat{y} = \sum_{i=1}^C \Psi_i(\vec{x}) \cdot l_i(\vec{x}) \quad \text{with } \Psi_i(\vec{x}) = \frac{\mu_i(\vec{x})}{\sum_{j=1}^C \mu_j(\vec{x})}, \quad (6)$$

with $\mu_i(\vec{x})$ as in [Equation \(1\)](#). From a statistical point of view, a TS fuzzy model can be interpreted as a collection of piecewise local linear predictors by a smooth (normalized) kernel, thus in its local parts (rules) having some synergies with local weighted regression (LWR) (Cleveland and Devlin, 1988). The difference is that in LWR the model is extracted on demand based on the nearest data samples [also termed as the reference base in an instance-based learning context for data streams (Shaker and Hüllermeier, 2012)] while TS fuzzy systems are providing a global model defined over the whole feature space (thus preferable in the context of interpretability issues and online prediction speed).

The most convenient choice for fuzzy sets in EFS and fuzzy systems design in general are Gaussian functions, which lead to the so-called *fuzzy basis function networks* (Wang and Mendel, 1992) and multi-variate kernels following normal distributions are achieved for representing the rules' antecedent parts:

$$\mu_i(\vec{x}) = \prod_{i=1}^p \exp\left(-\frac{1}{2} \frac{(x_i - c_i)^2}{\sigma_i^2}\right). \quad (7)$$

In this sense, the linear hyper-planes l_i are connected with multi-variate Gaussians to form an overall smooth function. Then, the output form in Equation (6) becomes some synergies with Gaussian mixture models (GMMs) (Day, 1969; Sun and Wang, 2011), often used for clustering and pattern recognition tasks (Bishop, 2007; Duda *et al.*, 2000). The difference is that l_i 's are hyper-planes instead of singleton weights and do not reflect the degree of density of the corresponding rules (as *mixing proportion*), but the linear trend of the approximation/regression surface in the corresponding local parts.

3.2.2.2. Takagi–Sugeno generalized

Recently, the generalized form of TS fuzzy systems has been offered to the evolving fuzzy systems community, launching its origin in Lemos *et al.* (2011a) and Leite *et al.* (2012a); latter explored and further developed in Pratama *et al.* (2014a) and Pratama *et al.* (2014b). The basic principle is that it employs multi-dimensional normal (Gaussian) distributions in arbitrary position for representing single rules. Thus, it overcomes the deficiency of not being able to model local correlations between input and output variables appropriately, as is the case with the t-norm operator used in standard rules (Klement *et al.*, 2000)—these may represent inexact approximations of the real local trends and finally causing information loss in rules (Abonyi *et al.*, 2002).

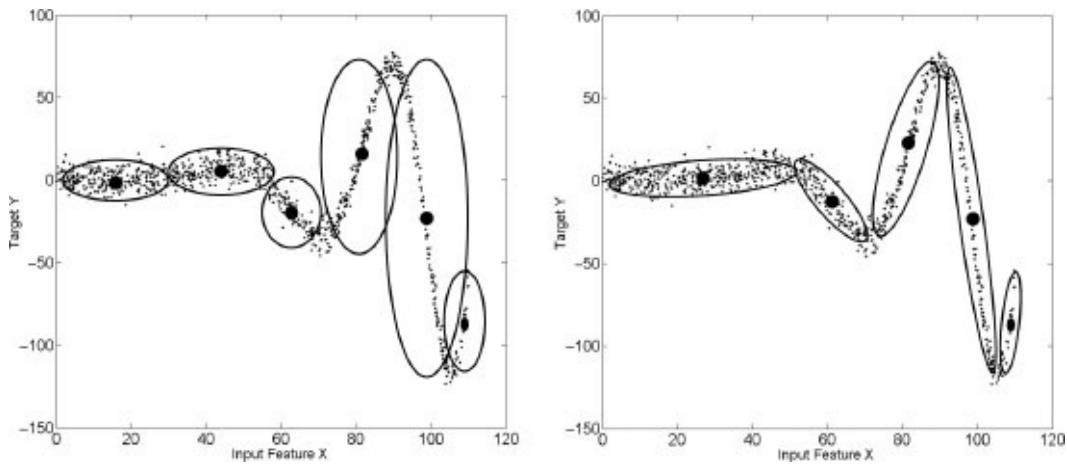


Figure 3.2: Left: Conventional axis parallel rules (represented by ellipsoids) achieve an inaccurate representation of the local trends (correlations) of a nonlinear approximation problem (defined by noisy data samples). Right: Generalized rules (by rotation) achieve a much more accurate representation.

An example for visualizing this problematic nature is provided in Figure 3.2: in the left image, axis-parallel rules (represented by ellipsoids) are used for modeling the partial

tendencies of the regression curves which are not following the input axis direction, but are rotated to some degree; obviously, the volume of the rules are artificially blown-up and the rules do not represent the real characteristics of the local tendencies well → *information loss*. In the right image, non axis-parallel rules using general multivariate Gaussians are applied for a more accurate representation (rotated ellipsoids).

To avoid such information loss, the generalized fuzzy rules have been defined in Lemos *et al.* (2011a) (there used for evolving stream mining), as

$$\text{IF } \vec{x} \text{ IS (about)} \Psi_i \text{ THEN } l_i(\vec{x}) = w_{i0} + w_{i1}x_1 + w_{i2}x_2 + \cdots + w_{ip}x_p, \quad (8)$$

where Ψ denotes a high-dimensional kernel function, which in accordance to the basis function networks spirit are given by the generalized multivariate Gaussian distribution:

$$\Psi_i(\vec{x}) = \exp\left(-\frac{1}{2}(\vec{x} - \vec{c}_i)^T \Sigma_i^{-1}(\vec{x} - \vec{c}_i)\right), \quad (9)$$

with \vec{c}_i the center and Σ_i^{-1} the inverse covariance matrix of the i th rule, allowing any possible rotation and spread of the rule. It is also known in the neural network literature that Gaussian radial basis functions are a nice option to characterize local properties (Lemos *et al.*, 2011a; Lippmann, 1991); especially, someone may inspect the inner core part, i.e., all samples fulfilling $(\vec{x} - \vec{c}_i)^T \Sigma_i^{-1}(\vec{x} - \vec{c}_i) \leq 1$, as the characteristic contour/spread of the rule.

The fuzzy inference then becomes a linear combination of multivariate Gaussian distributions in the form:

$$\hat{y} = \frac{\sum_{i=1}^C l_i(\vec{x}) * \exp\left(-\frac{1}{2}(\vec{x} - \vec{c}_i)^T \Sigma_i^{-1}(\vec{x} - \vec{c}_i)\right)}{\sum_{i=1}^C \exp\left(-\frac{1}{2}(\vec{x} - \vec{c}_i)^T \Sigma_i^{-1}(\vec{x} - \vec{c}_i)\right)} = \sum_{i=1}^C l_i(\vec{x}) \Phi_i(\vec{x}), \quad (10)$$

with C the number of rules, $l_i(\vec{x})$ the consequent hyper-plane of the i th rule and Φ_i the normalized membership degrees, summing up to 1 for each query sample.

In order to maintain (input/output) interpretability of the evolved TS fuzzy models for users/operators (see also [Section 3.5](#)), the authors in Lughofer *et al.* (2013) foresee a projection concept to form fuzzy sets and classical rule antecedents. It relies on the angle between the principal components directions and the feature axes, which has the effect that long spread rules are more effectively projected than when using the inner contour spreads (through axis parallel cutting points). The spread σ_i of the projected fuzzy set is set according to:

$$\sigma_i = \max_{j=1,\dots,p} \left(\frac{r}{\sqrt{\lambda_j}} \cos(\Phi(e_i, a_j)) \right), \quad (11)$$

with r the range of influence of one rule, usually set to 1, representing the (inner)

characteristic contour/spread of the rule (as mentioned above). The center of the fuzzy set in the i th dimension is set equal to the i th coordinate of the rule center. $\Phi(e_i, a_j)$ denotes the angle between principal component direction (eigenvector a_j) and the i th axis e_i , λ_j the eigenvalue of the j th principal component.

3.2.2.3. Takagi–Sugeno extended

An extended version of Takagi–Sugeno fuzzy systems in the context of evolving systems has been applied in Komijani *et al.* (2012). There, instead of a hyper-plane $l_i = w_{i0} + w_{i1}x_1 + w_{i2}x_2 + \dots + w_{ip}x_p$, the consequent function for the i th rule is defined as LS_SVM model according to Smola and Schölkopf (2004):

$$l_i(\vec{x}) = \sum_{k=1}^N \alpha_{ik} K(\vec{x}, \vec{x}_k) + \beta_i, \quad (12)$$

with $K(.,.)$ a kernel function fulfilling the Mercer criterion (Mercer, 1909) for characterizing a symmetric positive semi-definite kernel (Zaanen, 1960), N the number of training samples and α and β the consequent parameters (support vectors and intercept) to learn. The l_i 's can be, in principle, combined within in any inference scheme, either with the standard one in Equation (6) or with the generalized one in Equation (10) [in Komijani *et al.* (2012), they are combined with Equation (6)]. The advantage of these consequents is that they are supposed to provide more accuracy, as a support vector regression modeling (Smola and Schölkopf, 2004) is applied to each local region. Hence, nonlinearities within local regions may be better resolved. On the other hand, the consequents are more difficult to interpret.

3.2.3. Type-2

Type-2 fuzzy systems were invented by Lotfi Zadeh in 1975 (Zadeh, 1975) for the purpose of modeling the uncertainty in the membership functions of usual (type-1) fuzzy sets. The distinguishing feature of a type-2 fuzzy set $\tilde{\mu}_{ij}$ versus its type-1 counterpart μ_{ij} is that the membership function values of $\tilde{\mu}_{ij}$ are blurred, i.e., they are no longer a single number in $[0, 1]$, but are instead a continuous range of values between 0 and 1, say $[a, b] \subseteq [0, 1]$. One can either assign the same weighting or a variable weighting to membership function values in $[a, b]$. When the former is done, the resulting type-2 fuzzy set is called an interval type-2 fuzzy set. When the latter is done, the resulting type-2 fuzzy set is called a general type-2 fuzzy set (Mendel and John, 2002).

The i th rule of an interval-based type-2 fuzzy system is defined in the following way (Liang and Mendel, 2000; Mendel, 2001):

$$\text{Rule}_i : \text{IF } x_1 \text{ IS } \tilde{\mu}_{i1} \text{ AND } \dots \text{ AND } x_p \text{ IS } \tilde{\mu}_{ip} \text{ THEN } l_i(\vec{x}) = \tilde{f}_i,$$

with \tilde{f}_i a general type-2 uncertainty function.

In case of a Takagi–Sugeno-based consequent scheme (as e.g., used in Juang and Tsao (2008), the first approach of an evolving type-2 fuzzy system), the consequent function becomes:

$$l_i(\vec{x}) = \tilde{w}_{i0} + \tilde{w}_{i1}x_1 + \tilde{w}_{i2}x_2 + \cdots + \tilde{w}_{ip}x_p, \quad (13)$$

with \tilde{w}_{ij} an interval set (instead of a crisp continuous value), i.e.,

$$\tilde{w}_{ij} = [c_{ij} - s_{ij}, c_{ij} + s_{ij}]. \quad (14)$$

In case of a Mamdani-based consequent scheme (as e.g., used in Tung *et al.* (2013), a recent evolving approach), the consequent function becomes: $l_i = \tilde{\Phi}_i$, with $\tilde{\Phi}_i$ a type two fuzzy set.

An enhanced approach for eliciting the final output is applied, the so-called Karnik–Mendel iterative procedure (Karnik and Mendel, 2001), where a type reduction is performed before the defuzzification process. In this procedure, the consequent values $\bar{l}_i = c_{i0} - s_{i0} + (c_{i1} - s_{i1})x_1 + (c_{i2} - s_{i2})x_2 + \cdots + (c_{ip} - s_{ip})x_p$ and $\underline{l}_i = c_{i0} + s_{i0} + (c_{i1} + s_{i1})x_1 + (c_{i2} + s_{i2})x_2 + \cdots + (c_{ip} + s_{ip})x_p$ are sorted in ascending order denoted as \bar{y}_i and \underline{y}_i for all $i = 1, \dots, C$. Accordingly, the membership values $\bar{\psi}_i(\vec{x})$ and $\underline{\psi}_i(\vec{x})$ are sorted in ascending order denoted as $\bar{\psi}_i(\vec{x})$ and $\underline{\psi}_i(\vec{x})$. Then, the outputs \bar{y} and \underline{y} are computed by:

$$\bar{y} = \frac{\sum_{i=1}^L \bar{\psi}_i(\vec{x})\bar{y}_i + \sum_{i=L+1}^C \underline{\psi}_i(\vec{x})\bar{y}_i}{\sum_{i=1}^L \bar{\psi}_i(\vec{x}) + \sum_{i=L+1}^C \underline{\psi}_i(\vec{x})} \quad \underline{y} = \frac{\sum_{i=1}^R \underline{\psi}_i(\vec{x})\underline{y}_i + \sum_{i=R+1}^C \bar{\psi}_i(\vec{x})\underline{y}_i}{\sum_{i=1}^R \underline{\psi}_i(\vec{x}) + \sum_{i=R+1}^C \bar{\psi}_i(\vec{x})} \quad (15)$$

with L and R positive numbers, often $L = \frac{C}{2}$ and $R = \frac{C}{2}$. Taking the average of these two yields the final output value y .

3.2.4. Neuro-Fuzzy

Most of the neuro-fuzzy systems (Fuller, 1999) available in literature can be interpreted as a layered structural form of Takagi–Sugeno–Kang fuzzy systems. Typically, the fuzzy model is transformed into a neural network structure (by introducing layers, connections and weights between the layers) and learning methods already established in the neural network context are applied to the neuro-fuzzy system. A well-known example for this is the ANFIS approach (Jang, 1993), where the back-propagation algorithm (Werbos, 1974) is applied and the components of the fuzzy model (fuzzification, calculation of rule fulfillment degrees, normalization, defuzzification), represent different layers in the neural network structure. However, the inference scheme finally leads to the same model outputs as for conventional TS fuzzy systems. A visualization example is presented in [Figure 3.3](#). This layered structure will be used by several EFS approaches as can be seen from [Tables](#)

3.1 and 3.2.

Recently, a new type of neuro-fuzzy architecture has been proposed by Silva *et al.* (2014), termed as neo-fuzzy neuron network, and applied in evolving context. It relies on the idea to use a set of TS fuzzy rules for each input dimension independently and then connect these with a conventional sum for obtaining the final model output. The domain of each input i is granulated into m complementary membership functions.

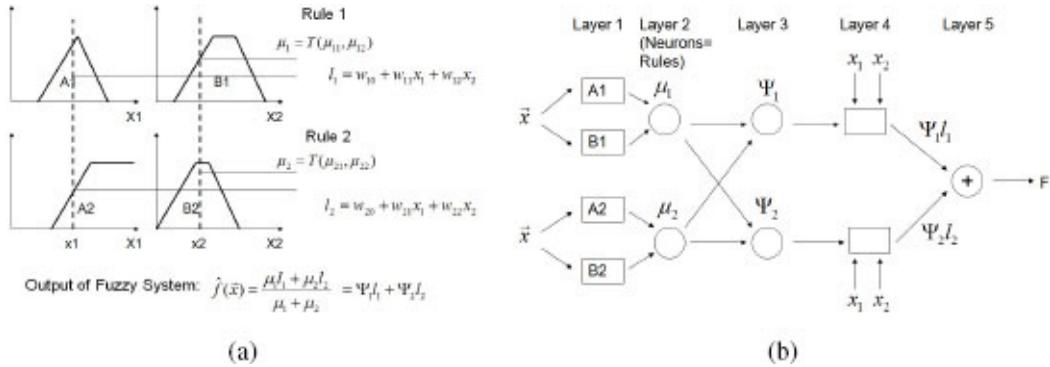


Figure 3.3: (a) Standard Takagi–Sugeno type fuzzy system, (b) Equivalent neural network structure.

3.2.5. Hierarchical Structures

Hierarchical architectures for evolving fuzzy modeling have been recently introduced in Shaker *et al.* (2013) and Lemos *et al.* (2011b). Both architectures have been designed for the purposes to provide a more slim, thus more transparent rule-base by inducing rules with flexible lengths. This is opposed to all the flat model architectures which have been presented above, always using all input features in all rules' antecedent parts.

The first approach is an incremental extension of top-down induction of fuzzy pattern trees (Senge and Hullermeier, 2011) and thus uses a hierarchical tree-like concept to evolve nodes and leaves on demand. Thereby, a collection of fuzzy sets and aggregation operators can be specified by the user as allowed patterns and conjunction operators in the leaf nodes. In particular, a pattern tree has the outlook as shown in the example of [Figure 3.4](#). Thereby, one basic difference to classical fuzzy systems is that the conjunction operator do not necessarily have to be t-norms [can be a more general aggregation operator (Saminger-Platz *et al.*, 2007)] and the type of the fuzzy sets can be different in different tree levels [as indicated in the rectangles in [Figure 3.4](#) (left)], allowing a composition of a mixture of patterns in hierarchical form. Another difference is the possibility to obtain a single compact rule for describing a certain characteristics of the output (a good house price quality with 0.9 in the example in [Figure 3.4](#)).

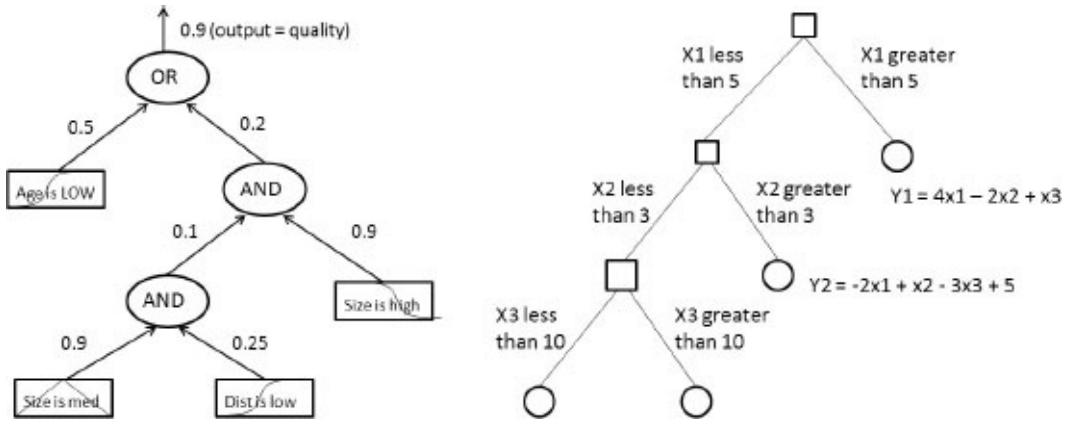


Figure 3.4: Left: Example of a fuzzy pattern tree which can be read as “IF ((Size is med AND Dist is high) AND Size is HIGH) OR Age is LOW THEN Output (Quality) is 0.9”. Right: Example of a fuzzy decision tree with four rules, a rule example is “IF x_1 is LESS THAN 5 AND x_2 is GREATER THAN 3 THEN $y_2 = -2x_1 + x_2 - 3x_3 + 5$ ”.

The second one (Lemos *et al.*, 2011b) has some synergies to classical decision trees for classification tasks [CART (Breiman *et al.*, 1993) and C4.5 (Quinlan, 1994)], where, however, the leafs are not class labels, but linear hyper-planes as used in classical TS fuzzy systems. Thus, as the partitioning may be arbitrarily fine-granulated as is the case for classical TS fuzzy systems, they still enjoy the favorable properties of being universal approximators. A visual example of such a tree is shown in the right image in Figure 3.4. It is notable that the nodes do not contain crisp decision rules, but fuzzy terms “Smaller Than” and “Greater Than”, which are represented by sigmoidal fuzzy sets: e.g., “Less Than 5” is a fuzzy set which cuts the fuzzy set “Greater than 5” at $x = 5$ with a membership degree 0.5. One path from the root node to a terminal node represents a rule, which is then similar to a classical TS fuzzy rule, but allowing an arbitrary length.

3.2.6. Classifiers

Fuzzy classifiers have been enjoyed a wide attraction in various applications since almost two decades (Eitzinger *et al.*, 2010; Kuncheva, 2000; Nakashima *et al.*, 2006). Their particular strength is the ability to model decision boundaries with arbitrary nonlinearity degree while maintaining interpretability in the sense “which rules on the feature set imply which output class labels (classifier decisions)”. In a winner-takes-it-all concept, the decision boundary proceeds between rule pairs having different majority class labels. As rules are usually nonlinear contours in the high-dimensional space, the nonlinearity of the decision boundary is induced — enjoying arbitrary complexity due to a possible arbitrary number of rules. If rules have linear contours, then overall nonlinearity is induced in form of piecewise linear boundaries between rule pairs.

3.2.6.1. Classical and extended single-model

The rule in a classical fuzzy classification model architecture with singleton consequent labels is a widely studied architecture in the fuzzy systems community (Ishibuchi and Nakashima, 2001; Kruse *et al.*, 1994; Kuncheva, 2000; Nauck and Kruse, 1998) and is

defined by:

$$\text{Rule}_i : \text{IF } x_1 \text{ IS } \mu_{i1} \text{ AND } \dots \text{ AND } x_p \text{ IS } \mu_{ip} \text{ THEN } l_i = L_i, \quad (16)$$

where L_i is the crisp output class label from the set $\{1, \dots, K\}$ with K the number of classes for the i th rule. This architecture precludes use of confidence labels in the single classes per rule. In case of clean classification rules, when each single rule contains/covers training samples from a single class, this architecture provides adequate resolution of the class distributions. However, in real-world problems, classes usually overlap significantly and therefore often rules are extracted containing samples from more than one class.

Thus, an extended fuzzy classification model that includes the confidence levels $conf_{i1}, \dots, conf_{iK}$ of the i th rule in the single classes has been applied in evolving, adaptive learning context (see e.g., Bouchachia, 2009; Bouchachia and Mittermeir, 2006):

$$\begin{aligned} \text{Rule}_i : & \text{ IF } x_1 \text{ IS } \mu_{i1} \text{ AND } \dots \text{ AND } x_p \text{ IS } \mu_{ip} \\ & \text{THEN } l_i = [conf_{i1}, conf_{i2}, \dots, conf_{iK}]. \end{aligned} \quad (17)$$

Thus, a local region represented by a rule in the form of Equation (17) can better model class overlaps in the corresponding part of the feature space: for instance, three classes overlap with a support of 200, 100 and 50 samples in one single fuzzy rule; then, the confidence in Class #1 would be intuitively 0.57 according to its relative frequency (200/350), in Class #2 it would be 0.29 (100/350) and in Class #3 it would be 0.14 (50/350). A more enhanced treatment of class confidence levels will be provided in [Section 3.4.5](#) when describing options for representing reliability in class responses.

In a winner-takes-it-all context [the most common choice in fuzzy classifiers (Kuncheva, 2000)], the final classifier output L will be obtained by

$$L = l_{i^*} \text{ with } l_{i^*} = \operatorname{argmax}_{1 \leq k \leq K} conf_{i^*k} \quad i^* = \operatorname{argmax}_{1 \leq i \leq C} \mu_i(\vec{x}) \text{ with} \quad (18)$$

In a more enhanced (weighted) classification scheme as recently used for evolving fuzzy classifiers (EFC) in Lugofer (2012a), the *degree of purity* is respected as well and integrated into the calculation of the final classification response L :

$$L = \operatorname{argmax}_{k=m, m*} \left(conf_k = \frac{\mu_1(\vec{x})h_{1,k} + \mu_2(\vec{x})h_{2,k}}{\mu_1(\vec{x}) + \mu_2(\vec{x})} \right) \quad (19)$$

with

$$h_{1,k} = \frac{h_{1,k}}{h_{1,m} + h_{1,m*}} \quad h_{2,k} = \frac{h_{2,k}}{h_{2,m} + h_{2,m*}} \quad (20)$$

and $h_{i,k}$ the class frequency of class k in rule i , and $\mu_1(\vec{x})$ the membership degree of the nearest rule (with majority class m), and $\mu_2(\vec{x})$ the membership degree of the second

nearest rule with a different majority class $m^* \neq m$. This difference is important as two nearby lying rules with the same majority class label do not induce a decision boundary in-between them. The nearest rule and second nearest rule are obtained by sorting the membership degrees of the current query point to all rules.

[Figure 3.5](#) shows an example for decision boundaries induced by Equation (18) (left image) and by Equation (19) (right image). Obviously, a more purified rule (i.e., having less overlap in classes, right side) is favored among that with significant overlap (left side), as the decision boundary moves away → more samples are classified to the majority class in the purified rule, which is intended to obtain a clearer, less uncertain decision boundary. A generalization of Equation (19) would be that k varies over all classes: then, an overwhelmed but significant class in two nearby lying rules may become also the final output class label L , although it has no majority in both rules. On the other hand, this variant would then be able to output a certainty level for each class, an advantage which could be used when calculating a kind of reliability degree overall classes (see [Section 3.6.4](#)) respectively when intending to normalize and study class certainty distributions. This variant has not been studied under the scope of EFC so far.

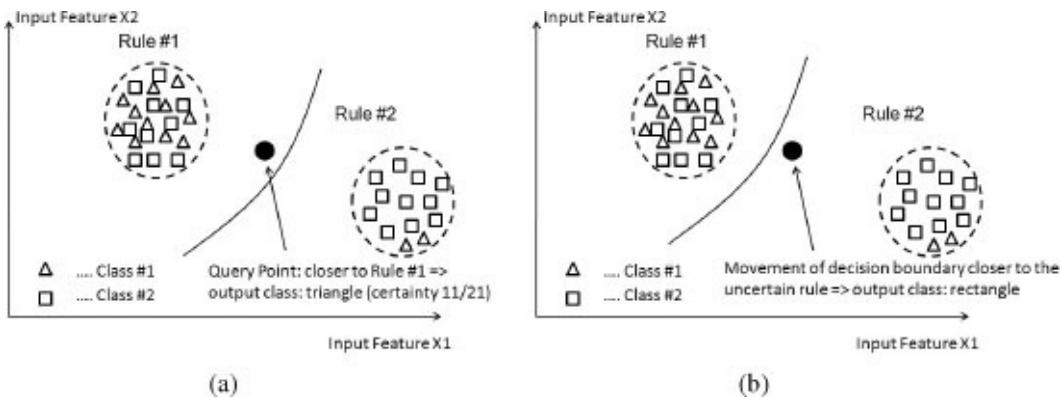


Figure 3.5: (a) Classification according to the winner takes it all concepts using Equation (18); (b) The decision boundary moves towards the more unpurified rule due to the *gravitation concept* applied in Equation (19).

3.2.6.2. Multi-model one-versus-rest

The first variant of multi-model architecture leans on the well-known *one-versus-rest* classification scheme from the field of machine learning (Bishop, 2007) and has been introduced in the fuzzy community and especially evolving fuzzy systems community in Angelov *et al.* (2008). It diminishes the problematic of having complex nonlinear multi-decision boundaries in case of multi-class classification problems, which is the case for single model architecture as all classes are coded into one model. This is achieved by representing K binary classifiers for the K different classes, each one for the purpose to discriminate one single class from the others (→ *one-versus-rest*). Thus, during the training cycle (batch or incremental), for the k th classifier all feature vectors resp. samples belonging to the k th class are assigned a label of 1, and all other samples belonging to other classes are assigned a label of 0.

The nice thing is that a (single model) classification model $D(f) = C$ respectively any regression model $D(f) = R$ (such as Takagi–Sugeno variants discussed in this section) can be applied for one sub-model in the ensemble. Interestingly, in Angelov *et al.* (2008), it has been studied that, when using Takagi–Sugeno architecture for the binary classifiers by regressing on $\{0, 1\}$, the masking problem as occurring in *linear regression by indicator matrix* approach can be avoided (Hastie *et al.*, 2009). This is due to the increased flexibility of TS fuzzy systems, being able to resolve nonlinearities in the class regression surfaces.

At the classification stage for a new query point \vec{x} , the model which is producing the maximal model response is used as basis for the final classification label output L , i.e.,

$$\begin{aligned} L &= \text{argmax}_{m=1, \dots, K} \hat{f}_m(\vec{x}) \text{ in case when } D(f) = R, \\ L &= \text{argmax}_{m=1, \dots, K} \text{conf}_m(\vec{x}) \text{ in case when } D(f) = C. \end{aligned} \quad (21)$$

Recently, a rule-based one-versus-rest classification scheme was proposed within the context of a MIMO (Multiple Input Multiple Output) fuzzy system and applied in an evolving classification context (Pratama *et al.*, 2014c). There, a rule is defined by:

$$\text{IF } \vec{x} \text{ IS (about) } \Psi_i \text{ THEN } l_i = \vec{x} \Omega_i, \quad (22)$$

where

$$\Omega_i = \begin{bmatrix} w_{i0}^1 & w_{i0}^2 & \dots & w_{i0}^K \\ w_{i1}^1 & w_{i1}^2 & \dots & w_{i1}^K \\ \vdots & \vdots & \vdots & \vdots \\ w_{ip}^1 & w_{ip}^2 & \dots & w_{ip}^K \end{bmatrix}.$$

Thus, a complete hyper-plane for each class per rule is defined. This offers the flexibility to regress on different classes within single rules, thus to resolve class overlaps in a single region by multiple regression surfaces (Pratama *et al.*, 2014c).

3.2.6.3. Multi-model all-pairs

The multi-model *all-pairs* (*aka all-versus-all*) classifier architecture, originally introduced in the machine learning community (Allwein *et al.*, 2001; Fürnkranz, 2002) and firstly introduced for (evolving) fuzzy classifiers design in Lughofe and Buchtala (2013), overcomes the often occurring imbalanced learning problems induced by one-versus-rest classification scheme in case of *multi-class (polychotomous)* problems. On the other hand, it is well known that imbalanced problems cause severe down-trends in classification accuracy (He and Garcia, 2009). Thus, it is beneficial to avoid imbalanced problems while still trying to enforce the decision boundaries as easy as possible to learn. This is achieved by the all-pairs architecture, as for each class pair (k, l) an own classifier is trained,

decomposing the whole learning problem into binary less complex sub-problems.

Formally, this can be expressed by a classifier $\mathbb{C}_{k,l}$ which is induced by a training procedure $\mathbb{T}_{k,l}$ when using (only) the class samples belonging to classes k and l :

$$\mathbb{C}_{k,l} \leftarrow \mathbb{T}_{k,l}(X_{k,l}) \quad X_{k,l} = \{\vec{x} | L(\vec{x}) = k \vee L(\vec{x}) = l\}, \quad (23)$$

with $L(\vec{x})$ the class label associated with feature vector \vec{x} . This means that $\mathbb{C}_{k,l}$ is a classifier for separating samples belonging to class k from those belonging to class l . It is notable that any classification architecture as discussed above in [Section 3.2.6.1](#) or any regression-based model as defined in [Section 3.2.2](#) can be used for $\mathbb{C}_{k,l}$.

When classifying a new sample \vec{x} , each classifier outputs a confidence level $\text{conf } f_{k,l}$ which denotes the degree of preference of class k over class l for this sample. This degree lies in $[0, 1]$ where 0 means no preference, i.e., a crisp vote for class l , and 1 means a full preference, i.e., a crisp vote for class k . This is conducted for each pair of classes and stored into a *preference relation matrix* R :

$$R = \begin{bmatrix} 0 & \text{conf}_{1,2} & \text{conf}_{1,3} & \dots & \text{conf}_{1,K} \\ \text{conf}_{2,1} & 0 & \text{conf}_{2,3} & \dots & \text{conf}_{2,K} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \text{conf}_{K,1} & \text{conf}_{K,2} & \text{conf}_{K,3} & \dots & 0 \end{bmatrix}. \quad (24)$$

If we assume reciprocal preferences, i.e., $\text{conf } f_{k,l} = 1 - \text{conf } f_{l,k}$, then the training of half of the classifiers can be omitted, hence finally $\frac{K(K-1)}{2}$ binary classifiers are obtained. The preference relation matrix in Equation (24) opens another interpretation dimension on output level: considerations may go into partial uncertainty reasoning or preference relational structure in a fuzzy sense (Hüllermeier and Brinker, 2008). In the most convenient way, the final class response is often obtained by:

$$L = \text{argmax}_{k=1,\dots,K}(\text{score}_k) = \text{argmax}_{k=1,\dots,K} \left(\sum_{K \geq i \geq 1} \text{conf}_{k,i} \right). \quad (25)$$

i.e., the class with the highest score = highest preference degree summed up over all classes is returned by the classifier.

In Fürnkranz (2002, 2001) it was shown that pairwise classification is not only more accurate than one-versus-rest technique, but also more efficient regarding computation times [see also Lughofer and Buchtala (2013)], which is an important characteristic for fast stream learning problems. The reason for this is basically that binary classification problems contain significantly lower number of samples, as each sub-problem uses only a small subset of samples.

3.3. Fundamentals

Data streams are one of the fundamental reasons for the necessity of applying evolving, adaptive models in general and evolving fuzzy systems in particular. This is simply because streams are theoretically an infinite sequence of samples, which cannot be processed at once within a batch process, even not in modern computers with high virtual memory capacities. Data streams may not necessarily be online based on permanent recordings, measurements or sample gatherings, but can also arise due to a block- or sample-wise loading of batch data sites, e.g., in case of *very large databases* (VLDB)⁶ or in case of *big data* problems (White, 2012); in this context, they are also often referred as *pseudo-streams*. In particular, a data stream (or pseudo-stream) is characterized by the following properties (Gama, 2010):

- The data samples or data blocks are continuously arriving online over time. The frequency depends on the frequency of the measurement recording process.
- The data samples are arriving in a specific order, over which the system has no control.
- Data streams are usually not bounded in a size; i.e., a data stream is alive as long as some interfaces, devices or components at the system are switched on and are collecting data.
- Once a data sample/block is processed, it is usually discarded immediately, afterwards.

Changes in the process such as new operation modes, system states, varying environmental influences etc. are usually implicitly also effecting the data stream in way that for instance *drifts* or *shifts* may arise (see [Section 3.4.1](#)), or new regions in the feature/system variable space are explored (knowledge expansion).

Formally, a stream can be defined as an infinite sequence of samples $(\vec{x}_1, \vec{y}_1), (\vec{x}_2, \vec{y}_2), (\vec{x}_3, \vec{y}_3), \dots$, where \vec{x} denotes the vector containing all input features (variables) and \vec{y} the output variables which should be predicted. In case of unsupervised learning problems, \vec{y} disappears—note that, however, in the context of fuzzy systems, only supervised regression and classification problems are studied. Often $y = \vec{y}$, i.e., single output systems are encouraged, especially as it is often possible to decast a MIMO (multiple input multiple output problem) system into single independent MISO (multiple input single output problem) systems (e.g., when the outputs are independent).

Handling streams for modeling tasks in an appropriate way requires the usage of *incremental learning* algorithms, which are deduced from the concept of incremental heuristic search (Koenig *et al.*, 2004). These algorithms possess the property to build and learn models in step-wise manner rather than with a whole dataset at once. From formal mathematical point of view, an incremental model update I of the former model f_N (estimated from the N initial samples) is defined by

$$f_{N+m} = I(f_N, (\vec{x}_{N+1}, \dots, \vec{x}_{N+m}, \vec{y}_{N+1}, \dots, \vec{y}_{N+m})).$$

So, the incremental model update is done by just taking the new m samples and the old model, but not using any prior data. Hereby, the whole model may also include some additional statistical help measures, which needs to be updated synchronously to the ‘real’ model. If $m = 1$, we speak about *incremental learning in sample mode* or *sample-wise incremental learning*, otherwise about *incremental learning in block mode* or *block-wise incremental learning*. If the output vector starts to be missing in the data stream samples, but a supervised model has been trained already before which is then updated with the stream samples either in unsupervised manner or by using its own predictions, then someone speaks about *semi-supervised* (online) learning (Chapelle *et al.*, 2006).

Two update modes in the incremental learning process are distinguished:

- (1) Update of the model parameters: In this case, a fixed number of parameters $\Phi_N = \{\phi_1, \dots, \phi_l\}_N$ of the original model f_N is updated by the incremental learning process and the outcome is a new parameter setting Φ_{N+m} with the same number of parameters, i.e., $|\Phi_{N+m}| = |\Phi_N|$. Here, we also speak about a *model adaptation* respectively a *model refinement* with new data.
- (2) Update of the whole model structure: This case leads to the evolving learning concept, as the number of the parameters may change and also the number of structural components may change automatically (e.g., rules are added or pruned in case of fuzzy systems) according to the characteristics of the new data samples $\tilde{x}_{N+1, \dots, N+m}$. This means that usually (but not necessarily) $|\Phi_{N+m}| \neq |\Phi_N|$ and $C_{N+m} \neq C_N$ with C the number of structural components. The update of the whole model structure also may include an update of the input structure, i.e., input variables/features may be exchanged during the incremental learning process—see also [Section 3.4.2](#).

An important aspect in incremental learning algorithms is the so-called *plasticity-stability dilemma* (Abraham and Robins, 2005), which describes the problem of finding an appropriate tradeoff between flexible model updates and structural convergence. This strongly depends on the nature of the stream: in some cases, a more intense update is required than in others (drifting versus life-long concepts in the stream). If an algorithm converges to an optimal solution or at least to the same solution as the hypothetical batch solution (obtained by using all data up to a certain sample at once), it is called a *recursive algorithm*. Such an algorithm is usually beneficial as long as no drifts arise, which make the older learned relations obsolete (see [Section 3.4.1](#)).

An *initial batch mode training step* with the first amount of training samples is, whenever possible, usually preferable to *incremental learning from scratch*, i.e., a building up of the model sample per sample from the beginning. This is because within a batch mode phase, it is possible to carry out validation procedures [such as cross-validation (Stone, 1974) or bootstrapping (Efron and Tibshirani, 1993)] in connection with search

techniques for finding an optimal set of parameters for the learning algorithm in order to achieve a good generalization quality. The obtained parameters are then usually reliable start parameters for the incremental learning algorithm to evolve the model further. When performing incremental learning from scratch, the parameters have to be set to some blind default values, which may be not necessarily appropriate for the given data stream mining problem.

In a pure online learning setting, however, incremental learning from scratch is indispensable. Then, often start default parameters of the learning engines need to be parameterized. Thus, it is beneficial that the algorithms require as less as possible parameters (see [Section 3.3.5](#)). Overcoming unlucky settings of parameters can be sometimes achieved with dynamic structural changes such as component-based split-and-merge techniques (as described in [Section 3.6.2](#)).

3.3.1. Recursive Learning of Linear Parameters

A lot of the EFS approaches available in literature (see [Section 3.3.5](#)) use the TS-type fuzzy systems architecture with linear parameters in the consequents. The reason lies in the highly accurate and precise models which can be achieved with these systems (Lughofer, 2011b), and therefore enjoy a wide attraction in several application fields, see [Sections 3.2.2](#) and [3.6.5](#). Also, within several classification variants, TS-fuzzy systems may be used as regression-based binary classifiers, e.g., in all-pairs technique (Lughofer and Buchtala, 2013) as well as in one-versus-rest classification schemes (Angelov *et al.*, 2008). Sometimes, singleton numerical values in the consequents (native Sugeno systems) or higher order polynomials (Takagi–Sugeno–Kang) are used in the consequents. These just change the number of parameters to learn but not the way how to learn them.

The currently available EFS techniques rely on the optimization of the least-squares error criterion, which is defined as the squared deviation between observed outputs y_1, \dots, y_N and predicted outputs $\hat{y}_1, \dots, \hat{y}_N$; thus:

$$J = \|y - \hat{y}\|_{L_2} = \sum_{k=1}^N (y(k) - \sum_{i=1}^C l_{i;\vec{w}}(\vec{x}(k))\Psi_i(\vec{x}(k)))^2 \rightarrow \min_{\vec{w}}. \quad (27)$$

This problem can be written as a classical linear least squares problem with a weighted regression matrix containing the global regressors

$$\vec{r}_i(k) = [\Psi_i(\vec{x}(k)) \ x_1(k)\Psi_i(\vec{x}(k)) \cdots x_p(k)\Psi_i(\vec{x}(k))], \quad (28)$$

for $i = 1, \dots, C$, with C the current number of rules and k the k th data sample denoting the k th row. For this problem, it is well known that a recursive solution exists which converges to the optimal one within each incremental learning step, see Ljung (1999) and Lughofer (2011b). Also [Chapter 2](#) for its detailed derivation in the context of evolving TS

fuzzy systems.

However, the problem with this *global learning* approach is that it does not offer any flexibility regarding rule evolution and pruning, as these cause a change in the size of the regressors and thus a dynamic change in the dimensionality of the recursive learning problem, which leads to a disturbance of the parameters in the other rules and to the loss of optimality. Therefore, the authors in Angelov *et al.* (2008) emphasize the usage of *local learning* approach which learns and updates the consequent parameters for each rule separately. Adding or deleting a new rule therefore does not affect the convergence of the parameters of all other rules; thus, optimality in least squares sense is preserved. The local learning approach leads to a weighted least squares formulation for each rule given by (without loss of generality the i th):

$$J_i = \sum_{k=1}^N \Psi_i(\vec{x}(k))(y(k) - \hat{y}_i(k))^2 \rightarrow \min_{\vec{w}_i} \quad i = 1, \dots, C, \quad (29)$$

with $\hat{y}_i(k) = l_{i;\vec{w}_i}(\vec{x}(k))$

This problem can be written as a classical weighted least squares problem, where the weighting matrix is a diagonal matrix containing the basis function values Ψ_i for each input sample. Again, an exact recursive formulation can be derived [see Lugofer (2011b) and [Chapter 2](#)], which is termed as *recursive fuzzily weighted least squares* (RFWLS). As RFWLS is so fundamental and used in many EFS approaches, we explicitly deploy the update formulas (from the k th to the $k + 1$ st cycle):

$$\vec{w}_i(k+1) = \vec{w}_i(k) + \gamma(k)(y(k+1) - \vec{r}^T(k+1)\vec{w}_i(k)), \quad (30)$$

$$\gamma(k) = P_i(k+1)\vec{r}(k+1) = \frac{P_i(k)\vec{r}(k+1)}{\frac{\lambda}{\Psi_i(\vec{x}(k+1))} + \vec{r}^T(k+1)P_i(k)\vec{r}(k+1)}, \quad (31)$$

$$P_i(k+1) = \frac{1}{\lambda}(I - \gamma(k)\vec{r}^T(k+1))P_i(k), \quad (32)$$

with $P_i(k) = (R_i(k)^T Q_i(k) R_i(k))^{-1}$ the inverse weighted Hessian matrix and $\vec{r}(k+1) = [1 \ x_1(k+1) \ x_2(k+1) \ \dots \ x_p(k+1)]^T$ the regressor values of the $k + 1$ st data sample, which is the same for all i rules, and λ a forgetting factor, with default value equal to 1 (no forgetting) — see [Section 3.4.1](#) for a description and meaning of its usage. Whenever $\lambda < 1$, the following function is minimized: $J_i = \sum_{k=1}^N \lambda^{N-k} \Psi_i(\vec{x}(k))(y(k) - \hat{y}_i(k))^2$, instead of Equation (29), thus samples which appeared a long time ago are almost completely outweighed. Obviously, the actual weight of the sample is Ψ_i (the membership degree to Rule i), thus a sample receives a low weight when it does not fall into rule i : then, the Kalman filter gain $\gamma(k)$ in Equation (31) becomes a value close to 0 and the update of P_i and \vec{w}_i is marginal. Again, Equation (30) converges to the optimal solution within one incremental learning step.

The assurance of convergence to optimality is guaranteed as long as there is not structural change in the rules' antecedents. However, due to rule center movements or resettings in the incremental learning phase (see [Section 3.3.4](#)), this is usually not the case. Therefore, a kind of sub-optimality is caused whose deviation degree to the real optimality could have been bounded for some EFS approaches such as FLEXFIS (Lughofer, 2008) and PANFIS (Pratama *et al.*, 2014a).

Whenever a new rule is evolved by a rule evolution criterion, the parameters and inverse weighted Hessian matrix (required for an exact update) have to be initialized. In Ljung (1999), it is emphasized to set \vec{w} to 0 and P_i to αI with α big enough. However, this is for the purpose of a global modeling approach starting with a faded out regression surface over the whole input space. In local learning, the other rules defining other parts of the feature space remain untouched. Thus, setting the hyper-plane of the new rule which may appear somewhere inbetween the other rules to 0 would lead to an undesired muting of one local region and to discontinuities in the online predictions (Cernuda *et al.*, 2012). Thus, it is more beneficial to inherit the parameter vector and the inverse weighted Hessian matrix from the most nearby lying rule (Cernuda *et al.*, 2012).

Recent extensions of RFWLS are as follows:

- In PANFIS (Pratama *et al.*, 2014a), an additional constant α is inserted, conferring a noteworthy effect to foster the asymptotic convergence of the system error and weight vector being adapted, which acts like a binary function. In other words, the constant α is in charge to regulate the current belief of the weight vectors \vec{w}_i and depends on the approximation and the estimation errors. It is 1 whenever the approximation error is bigger than the system error, and 0 otherwise. Thus, adaptation takes fully place in the first case and completely not place in the second case (which may have advantages in terms of flexibility and computation time). A similar concept is used in the improved version of SOFNN, see Leng *et al.* (2012).
- Generalized version of RFWLS (termed as *FWGRLS*) as used in GENEFIS (Pratama *et al.*, 2014b): This exploits the generalized RLS as derived in Xu *et al.* (2006) and adopts it to the local learning context in order to favor from its benefits as discussed above. The basic difference to RFWLS is that it adds a weight decay regularization term in the least squares problem formulation in order to punish more complex models. In a final simplification step, it ends up with similar formulas as in Equations (30)–(32), but with the difference to subtract the term $\alpha P_i(k + 1) \nabla \phi(\vec{w}_i(k))$ in Equation (30), with α a regularization parameter and ϕ the weight decay function: one of the most popular ones in literature is the quadratic function defined as $\phi(\vec{w}) = \frac{1}{2} \|\vec{w}\|^2$, thus $\nabla \phi = \vec{w}$.
- In some approaches [e.g., eMG (Lemos *et al.*, 2011a) or rGK (Dovzan and Skrjanc, 2011)], weights Ψ_i of the data samples are integrated in the second term of Equation (30) as well, which however do not exactly follow the original derivation of recursive

weighted least-squares (Aström and Wittenmark, 1994; Ljung, 1999), but leads to a similar effect.

Alternatively, Cara *et al.* (2013) proposes a different learning scheme for singleton consequent parameters (in a Sugeno fuzzy system) within an evolving fuzzy controller design, which relies on the prediction error of the current model. The update of the i th rule singleton consequent parameter w_{i0} becomes:

$$w_{i0}(k+1) = w_{i0}(k) + C\mu_i(\vec{x}(k+1))(y(k+1) - \hat{y}(k+1)), \quad (33)$$

with μ_i the activation degree of the i th rule as present in the previous time step k (before being updated with the new sample $\vec{x}(k+1)$), and C a normalization constant. Hence, instead of γ , μ_i is used as update gain which is multiplied with the normalization constant.

3.3.2. Recursive Learning of Nonlinear Parameters

Nonlinear parameters occur in every model architecture as defined throughout [Section 3.2.2](#), mainly only in the fuzzy sets included in the rules' antecedent parts — except for the extended version of TS fuzzy systems ([Section 3.2.2.3](#)), where they also appear in the consequents. Often, the parameters in the fuzzy sets define their centers c and characteristic spreads σ , but often the parameters may appear in a different form, for instance, in case of sigmoid functions they define the slope and the point of gradient change. Thus, we generally refer to a nonlinear parameter as ϕ and a whole set of nonlinear parameters as Φ . The incremental update of nonlinear parameters is necessary in order to adjust and move the fuzzy sets and rules composed by the sets to the actual distribution of the data in order to achieve always the correct, well-placed positions. An example is provided in [Figure 3.6](#) where the initial data cloud (circles) in the left upper part changes slightly the position due to new data samples (rectangles). Leaving the original rule (marked with an ellipsoid) untouched, would cause a misplacement of the rule. Thus, it is beneficial to adjust the rule center and its spread accordingly to the new samples. This figure also shows the case of a rule evolution in the lower right part (new samples significantly away from the old rule contour) — as will be discussed in the subsequent section.

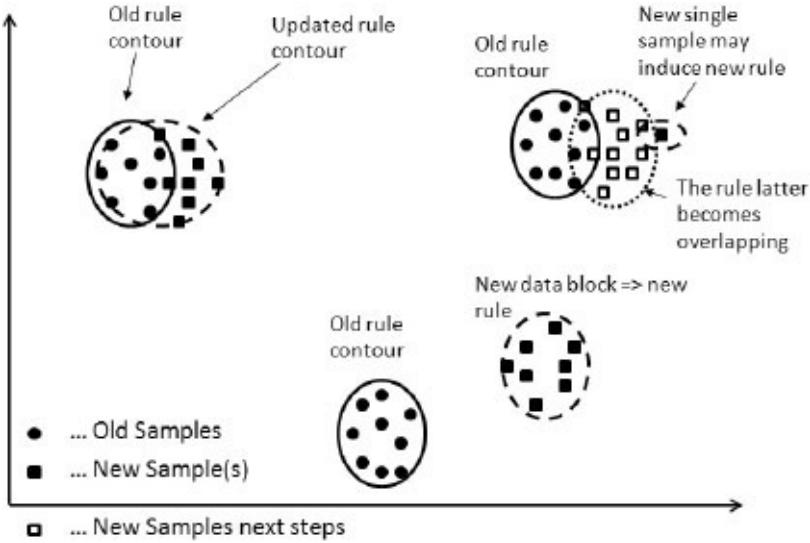


Figure 3.6: Three cases affecting rule contours (antecedents): The left upper part shows a case where a rule movement is demanded to appropriately cover the joint partition (old and new samples), the lower right part shows a case where a new rule should be evolved and the upper right part shows a case where sample-wise incremental learning may trigger a new rule which may turn out to be superfluous later (as future samples are filling up the gap forming one cloud) → (back-)merge requested as discussed in [Section 3.3.4](#).

A possibility to update the nonlinear parameters in EFS is again, similar to the consequent parameters, by applying a numerical incremental optimization procedure. Relying on the least squares optimization problem as in case of recursive linear parameter updates, its formulation in dependency of the nonlinear parameters Φ becomes:

$$J = J(\Phi) = \sum_{k=1}^N \|(\mathbf{y}_k - \hat{\mathbf{y}}(\Phi))\|_{L_2} \rightarrow \min_{\Phi; [\vec{w}]}!. \quad (34)$$

In case of TS fuzzy systems, for instance, $\hat{\mathbf{y}}(\Phi) = \sum_{i=1}^C l_i(\vec{x})\Psi_i(\Phi)(\vec{x})$. Then, the linear consequent parameters \vec{w} needs to be synchronously optimized to the nonlinear parameters (thus, in optional braces), in order to guarantee optimal solution. This can be done either in an alternating nested procedure, i.e., perform an optimization step for nonlinear parameters first, see below, and then optimizing the linear ones, e.g., by Equation (30), or within one joint update formula, e.g., when using one Jacobian matrix on all parameters, see below).

Equation (34) is still a free optimization problem, thus any numerical, gradient-based or Hessian-based technique for which a stable incremental algorithm can be developed is a good choice: this is the case for steepest descent, Gauss–Newton method and Levenberg–Marquardt. Interestingly, a common parameter update formula can be deduced for all three variants (Ngia and Sjöberg, 2000):

$$\Phi(k+1) = \Phi(k) + \mu(k)P(k)^{-1}\psi(\vec{x}(k), \Phi)e(\vec{x}(k), \Phi), \quad (35)$$

where $\psi(\vec{x}(k), \Phi) = \frac{\partial \mathbf{y}}{\partial \Phi}(\vec{x}(k))$ the partial derivative of the current model y after each nonlinear parameter evaluated at the current input sample $\vec{x}(k)$, $e(\vec{x}(k), \Phi)$ is the residual in the k th sample: $e(\vec{x}(k), \Phi) = \mathbf{y}_k - \hat{\mathbf{y}}_k$, and $\mu(k)P(k)^{-1}$ the learning gain with $P(k)$ an

approximation of the Hessian matrix, which is substituted in different ways:

- For the steepest descent algorithm, $P(k) = I$, thus the update depends only on first order derivative vectors; furthermore, $\mu(k) = \frac{\mu}{\|\vec{r}(k)\|^2}$ with $\vec{r}(k)$ the current regression vector.
- For Gauss–Newton, $\mu(k) = 1 - \lambda$ and $P(k) = (1 - \lambda)H(k)$ with $H(k)$ the Hessian matrix which can be approximated by $Jac^T(k)Jac(k)$ with Jac the Jacobian matrix (including the derivatives w.r.t. all parameters in all rules for all samples up to the k th) resp. by $Jac^T(k)diag(\Psi_i(\vec{x}(k)))Jac(k)$ in case of the weighted version for local learning (see also [Section 3.3.1](#)) — note that the Jacobian matrix reduces to the regression matrix R in case of linear parameters, as the derivatives are the original input variables (thus, $H = R^T R$ in case of recursive linear parameter learning resulting in the native (slow) recursive least squares without inverse matrix update). Additionally to updating the parameters according to Equation (35), the update of the matrix P is required, which is given by:

$$P(k) = \lambda P(k-1) + (1 - \lambda) \psi(\vec{x}(k), \Phi) \psi(\vec{x}(k), \Phi)^T. \quad (36)$$

- For Levenberg–Marquardt, $P(k) = (1 - \lambda)H(k) + \alpha I$ with $H(k)$ as in case of Gauss–Newton and again $\mu(k) = 1 - \lambda$. The update of the matrix P is done by:

$$P(k) = \lambda P(k-1) + (1 - \lambda) (\psi(\vec{x}(k), \Phi) \psi(\vec{x}(k), \Phi)^T + \alpha I). \quad (37)$$

Using matrix inversion lemma (Sherman and Morrison, 1949) and some reformulation operations to avoid matrix inversion in each step (P^{-1} is required in Equation (35)) leads to the well-known recursive Gauss–Newton approach, which is e.g., used in Komijani *et al.* (2012) for recursively updating the kernel widths in the consequents and also for fine-tuning the regularization parameter. It also results in the recursive least squares approach in case of linear parameters (formulas for the local learning variant in Equation (30)). In case of recursive Levenberg Marquardt (RLM) algorithm, a more complex reformulation option is requested to approximate the update formulas for $P(k)^{-1}$ directly (without intermediate inversion step). This leads to the recursive equations as successfully used in the EFP method by Wang and Vrbanek (2008) for updating centers and spreads in Gaussian fuzzy sets (multivariate Gaussian rules), see also Lughofer (2011b) and [Chapter 2](#).

3.3.3. Learning of Consequents in EFC

The most common choice in EFC design for consequent learning is simply to use the class majority voting for each rule separately. This can be achieved by incrementally counting the number of samples falling into each class k and rule i , h_{ik} (the rule which is the nearest one in the current data stream process). The class with majority count $k* = argmax_{k=1}^K h_{ik}$ is the consequent class of the corresponding (i th) rule in case of the classical architecture in

Equation (16). The confidences in each class per rule can be obtained by the relative frequencies among all classes $f_{ik} = \frac{h_{ik}}{\sum_{k=1}^K h_{ik}}$ in case of extended architecture in Equation (17). For multi-model classifiers, the same strategy can be applied within each single binary classifier. An enhanced confidence calculation scheme will be handled under the scope of *reliability* in [Section 3.4.5](#).

3.3.4. Incremental Partitioning of the Feature Space (Rule Learning)

A fundamental issue in *evolving systems*, which differs them from *adaptive systems* is that they possess the capability to change their structure during online, incremental learning cycles — adaptive systems are only able to update their parameters as described in the preliminary two sub-sections. The evolving technology addresses the dynamic expansion and contraction of the rule-base. Therefore, almost all of the EFS approaches foresee two fundamental concepts for incremental partitioning of the feature space (only some foresee only the first option):

- *Rule evolution*: It addresses the problem when and how to evolve new rules on-the-fly and on demand → knowledge expansion.
- *Rule pruning*: It addresses the problem when and how to prune rules in the rule-base on-the-fly and on demand → knowledge contraction, rule-base simplification.

The first issue guarantees to include new system states, operation modes, process characteristics in the models to enrich their knowledge and expand them to so far unexplored regions in the feature space. The second issue guarantees that a rule-base cannot grow forever and become extremely large, hence is responsible for smart computation times and compactness of the rule-base which may be beneficial for interpretability reasons, see [Section 3.5](#). Also, it is a helpful engine for preventing model over-fitting, especially in case when rules are evolved close to each other or are moving together over time, thus turning out to be superfluous at a later stage. Whenever new rules are evolved, the incremental update of their parameters (as described in the preliminary sub-sections) can begin and continue in the subsequent cycles.

The current state-of-the-art in EFS is that both concepts are handled in different ways in different approaches, see Lughofe (2011b) and “*Evolving Systems*” Journal by Springer⁷ for recently published approaches. Due to space limitations of this book chapter in the whole encyclopedia, it is not possible to describe the various options for rule evolution and pruning anchored in the various EFS approaches. Therefore, we outline the most important directions, which enjoy some common understanding and usage in various approaches.

One concept which is widely used is the *incremental clustering* technique [see Bouchachia (2011)] for a survey of methods], which searches for an optimal grouping of the data into several clusters, ideally following the natural distribution of data clouds. In

particular, the aim is that similar samples contribute to the (formation of the) same cluster while different samples should fall into different clusters (Gan *et al.*, 2007). In case when using clustering techniques emphasizing clusters with *convex shapes* (e.g., ellipsoids), these can then be directly associated with rules. Due to their projection onto the axes, the fuzzy sets appearing in the rule antecedents can be obtained. The similarity concept applied in the various approaches differ: some are using distance-oriented criteria [e.g., DENFIS (Kasabov and Song, 2002), FLEXFIS (Lughofer, 2008) or eFuMo (Zdsar *et al.*, 2014)], some are using density-based criteria [e.g., eTS (Angelov and Filev, 2004) and its extension eTS+ Angelov (2010), or Almaksour and Anquetil (2011)] and some others are using statistical-oriented criteria [e.g., ENFM (Soleimani *et al.*, 2010)]; this also affects the rule evolution criterion, often being a threshold (e.g., a maximal allowed distance) which decides whether a new rule is evolved or not. Distance-based criteria may be more prone to outliers than density-based and statistical-based ones; on the other hand, the latter ones can be quite lazy until new rules are evolved (e.g., a significant new dense area is required such that a new rule is evolved there). A summary of EFS approaches and which one applies which criterion will be given in [Section 3.3.5](#).

Fundamental and quite common to many incremental clustering approaches is the update of the centers \vec{c} defining the cluster prototype given by

$$\vec{c}(N+1) = \frac{N\vec{c}(N) + \vec{x}(N+1)}{N+1}, \quad (38)$$

and the update of the inverse covariance matrix Σ^{-1} defining the shape of clusters given by

$$\Sigma^{-1}(N+1) = \frac{\Sigma^{-1}(N)}{1-\alpha} - \frac{\alpha}{1-\alpha} \frac{(\Sigma^{-1}(N)(\vec{x} - \vec{c}))(\Sigma^{-1}(N)(\vec{x} - \vec{c}))^T}{1 + \alpha((\vec{x} - \vec{c})^T \Sigma^{-1}(N)(\vec{x} - \vec{c}))},$$

with $\alpha = \frac{1}{N+1}$ and N the number of samples seen so far. Usually, various clusters/rules are updated, each representing an own covariance matrix Σ_i^{-1} , thus the symbol $N = k_i$ then represents the number of samples “seen by the corresponding cluster so far”, i.e., falling into the corresponding cluster so far (also denoted as the *support of the cluster*).

Other concepts rely

- On the degree of the coverage of the current input sample, i.e., when the coverage is low, a new rule is evolved [e.g., SOFNN (Leng *et al.*, 2005), the approach in Leng *et al.* (2012)].
- On the system error criteria such as the one-step-ahead prediction, i.e., when this is high, a new rule is evolved (e.g., as in SOFNN (Leng *et al.*, 2005) or the approach in Leng *et al.* (2012)) or even a split is performed as in AHLTNM (Kalhor *et al.*, 2010).
- On the rule significance in terms of (expected) statistical contribution and influence, i.e., when a new rule is expected to significantly influence the output of the system it is actually added to the system [e.g., SAFIS (Rong *et al.*, 2006) and its extensions (Rong

et al., 2011; Rong, 2012) PANFIS (Pratama *et al.*, 2014a), GENEFIS (Pratama *et al.*, 2014b)).

- On Yager’s participatory learning concept (Yager, 1990), comparing the arousal index and the compatibility measure with thresholds [as in ePL (Lima *et al.*, 2010; Lemos *et al.*, 2013), eMG (Lemos *et al.*, 2011a)].
- On the goodness of fit tests based on statistical criteria (e.g., F-statistics) for candidate splits. The leafs are replaced with a new subtree, inducing an expansion of the hierarchical fuzzy model [e.g., in Lemos *et al.* (2011b) or incremental LOLIMOT (Local Linear Model Tree) (Hametner and Jakubek, 2013)].

Furthermore, most of the approaches which are applying an adaptation of the rule contours, e.g., by recursive adaptation of the nonlinear antecedent parameters, are equipped with a merging strategy for rules. Whenever rules are forced to move together due to the nature of data stream samples falling in-between these, they may become inevitably overlapping, see the upper right case in [Figure 3.6](#) for an example. The rule evolution concepts cannot omit such occasions in advance, as streaming samples are loaded in the same timely order as they are appearing/recorded in the system — sometimes, originally it may seem that two clusters are contained in the data, which may turn out later as erroneous assumption. Various criteria have been suggested to identify such occurrences and to eliminate them, see Lughofer (2013) and [Section 3.2](#) for a recent overview. In Lughofer and Hüllermeier (2011); Lughofer *et al.* (2011a), a generic concept has been defined for recognizing such overlapping situations based on fuzzy set and rule level. It is applicable for most of the conventional EFS techniques as relying on a geometric-based criterion employing a rule inclusion metric. This has been expanded in Lughofer *et al.* (2013) to the case of adjacent rules in the feature space showing the same trend in the antecedents and consequents, thus guaranteeing a kind of joint homogeneity. Generic rule merging formulas have been established in Lughofer *et al.* (2011a) and go hand in hand with consistency assurance, especially when equipped with specific merging operations for rule consequent parts, see also [Section 3.5](#).

3.3.5. EFS Approaches

In this section, we provide an overview of most important EFS approaches developed since the invention of evolving fuzzy systems approximately 10 years ago. Due to space limitations and the wide variety and manifold of the approaches, we are not able to give a compact summary about the basic methodological concepts in each of these. Thus, we restrict ourselves to report a rough comparison, which is based on the main characteristics and properties of the EFS approaches. This comparison is provided in [Table 3.1](#). Additionally, we demonstrate a pseudo-code in Algorithm 3.1, which shows more or less a common denominator of which steps are performed within the learning engines of the EFS approaches.

Algorithm 3.1. Key Steps in an Evolving Fuzzy Systems Learning Engine

- (1) Load new data sample \vec{x} .
- (2) Pre-process data sample (e.g., normalization).
- (3) **If** rule-base is empty, initialize the first rule with its center to the data sample $\vec{c} = \vec{x}$ and its spread (range of influence) to some small value; go to Step (1).
- (4) **Else**, perform the following Steps (5–10):
 - (5) Check if rule evolution criteria are fulfilled
 - (a) If yes, evolve a new rule ([Section 3.3.4](#)) and perform body of Step (3) (without if-condition).
 - (b) If no, proceed with next step.
 - (6) Update antecedents parts of (some or all) rules ([Sections 3.3.4](#) and [3.3.2](#)).
 - (7) Update consequent parameters (of some or all) rules ([Sections 3.3.3](#) and [3.3.1](#)).
 - (8) Check if the rule pruning/merging criteria are fulfilled
 - (a) If yes, prune or merge rules ([Section 3.3.4](#)); go to Step (1).
 - (b) If no, proceed with next step.
 - (9) Optional: Perform corrections of parameters towards more optimality.
 - (10) Go to Step (1).

One comment refers to the update of antecedents and consequents: some approaches may only update those of *some* rules (e.g., the rule corresponding to the winning cluster), others may always update those of *all* rules. The former may have some advantages regarding the prevention of the *unlearning effect* in parts where actual samples do not fall (Lughofer, 2010a), the latter achieves significance and thus reliability in the rule parameters faster (more samples are used for updating).

Table 3.1: Comparison of properties and characteristics of important EFS approaches.

Method	Architecture	Ant. Learning	Cons. Learning	Rule Pr.	Forget.	Dim. Red.	# of P.	Note
AHLTNM (Kalhor <i>et al.</i> , 2010)	TS fuzzy system	Incremental Split-and-Merge (own)	RFWLS	Yes	Yes, cons. only	No	1–2	
DENFIS (Kasabov and Song, 2002)	Neuro-fuzzy system	Evolving Clustering (ECM)	RFWLS with distance weights	No	No	No	1–2	One of the first approaches
EFC-AP (Lughofer and Buchtala, 2013)	All-pairs classifiers	Evolving Clustering (eVQ)	RFWLS or Consequent labeling	Yes	No	No	1–2	Very high accuracy, first all-pairs EFC
EFP (Wang and Vrbanek, 2008)	Mamdani + TS fuzzy system	Evolving Clustering (own) + RLM	RLM + RLS (global)	Yes	Yes, cons. only	No	1–2	First attempt of recursive learning of nonlinear params
eFPT (Shaker <i>et al.</i> , 2013)	Tree-based structure	Dynamic substitution with neighbor tree	None	Yes	No	Yes	3	Good interpretability, exploits architecture from Huang <i>et al.</i> (2008)
eFT (Lemos <i>et al.</i> , 2011b)	Tree-based structure	Replacement of leaves with subtrees	RFWLS double-weight	No	No	Yes	4	First approach of evolving fuzzy decision tree
eFuMo (Zdsar <i>et al.</i> , 2014)	Dynamic TS fuzzy system (lags)	Evolving Clustering (Gustafson-Kessel)	RFWLS double-weight.	Yes	Yes, cons. + No ante.		3–5	Capable of splitting of rules
eMG (Lemos <i>et al.</i> , 2011a)	Generalized TS fuzzy system	Participatory Learning	RFWLS double-weight	Yes	No	No	4	First approach using generalized rules
ENFM (Soleimani <i>et al.</i> , 2010)	Generalized TS fuzzy system	Recursive Clustering (Gath-Geva)	RFWLS	Yes	Yes, cons. only	No	3	
eClass (Angelov <i>et al.</i> , 2008)	single model, one-versus-rest classifiers and MIMO	Evolving clustering (eClustering)	RFWLS + or Cons. labeling	No	Yes, cons. only	No	1–2	First one-vs-rest in EFC
ePL (Lima <i>et al.</i> , 2010)	TS fuzzy system	Participatory Learning	RLS (global)	Yes	No	No	5	First usage of participatory (Yager, 1990)
eT2FIS (Tung <i>et al.</i> , 2013)	Type-2 Mamdani	Native activation levels, parameter updates	same as antecedent learning	Yes	No	No	3	First evolving Type-2 Mamdani system
eTS(+) (Angelov, 2010; Angelov and Filev, 2004)	TS fuzzy system	Evolving clustering (eClustering)	RFWLS double-weight	Yes	Yes, later in (Lughofer and Angelov, 2011)	Yes	1–2	One of the pioneering methods (2004)
eTS-LS-SVM (Komijani <i>et al.</i> , 2012)	Extended TS fuzzy system	Recursive clustering, suitability	Recursive Gauss–Newton	Yes	No	No	6 (2 + 4)	First approach for evolving extended TS fuzzy system

Although many of these have different facets with a large variety of pros and cons, which cannot be strictly arranged in an ordered manner to say one method is for sure better than the other, the number of parameters (last but one column) gives somehow a bit clarification about the usability resp. the effort to tune the method and finally, to let it run. Tendentially, the more parameters a method has, the more sensitive it is to a particular result and the higher the effort to get it successfully run in an industrial installation. In case when mentioning “X – Y” number of parameters it means that Y parameters are the case when forgetting is parameterized (fixed forgetting factor), which is often an optional

choice in many applications.

For further details about the concrete algorithms and concepts of the approaches listed in [Tables 3.1](#) and [3.2](#), please refer to Lughofer (2011b), describing approaches in a compact detailed form from the origin of EFS up to June 2010 and to recently published ones (since July 2010) in “*Evolving Systems*” Journal by Springer⁸ as well as papers in the recent special issues “Evolving Soft Computing Techniques and Applications” (Bouchachia *et al.*, 2014) in *Applied Soft Computing* Journal (Elsevier) and “Online Fuzzy Machine Learning and Data Mining” (Bouchachia *et al.*, 2013) in *Information Sciences* Journal (Elsevier), and also some recent regular contributions in “IEEE Transactions on Fuzzy Systems”⁹ and “IEEE Transactions on Neural Networks and Learning Systems”¹⁰ (neuro-fuzzy approaches).

3.4. Stability and Reliability

Two important issues when learning from data streams are the assurance of stability during the incremental learning process and the investigation of reliability of model outputs in terms of predictions, forecasts, quantifications, classification statements etc. These usually leads to an enhanced *robustness* of the evolved models. Stability is usually guaranteed by all the aforementioned approaches listed in [Tables 3.1](#) and [3.2](#) as long as the data streams from which the models are learnt appear in a quite “smooth, expected” fashion. However, specific occasions such as *drifts* and *shifts* (Klinkenberg, 2004) or high noise levels may appear in these, which require a specific handling within the learning engine of the EFS approaches. Another problem is dedicated to high-dimensional streams, usually stemming from large-scale time-series (Morreale *et al.*, 2013) embedded in the data, and mostly causing a *curse of dimensionality* effect, which leads to over-fitting and downtrends in model accuracy. As can be realized from Column #7 in [Tables 3.1](#) and [3.2](#), only a few approaches embed any online dimensionality reduction procedure so far in order to diminish this effect. Although high noise levels can be automatically handled by RFWLS, its spin-offs and modifications as well as by the antecedent learning engines discussed in [Sections 3.3.2](#) and [3.3.4](#), reliability aspects in terms of increasing the certainty in model outputs respecting the noise are weakly discussed. Drift handling is included in some of the methods by forgetting strategies, but these are basically only applied for consequent learning in Takagi–Sugeno type fuzzy models (see Column 6 in [Tables 3.1](#) and [3.2](#)).

Table 3.2: Comparison of properties and characteristics of important EFS approaches.

Method	Architecture	Ant. Learning	Cons. Learning	Rule Pr.	Forget.	Dim. Red.	# of P.	Note
FLEXFIS (++) (Lughofer, 2008, 2012b)	TS fuzzy system	Evolving clustering (eVQ)	RFWLS	Yes	Yes, cons. + ante.	No	1–2	First approach forgetting in ante, enhanced robustness
FLEXFIS-Class (Angelov <i>et al.</i> , 2008; Lughofer, 2011c)	Single model, one-vs-rest classifiers	Evolving clustering (eVQ)	RFWLS	No	Yes, cons. only	Yes (smooth)	1–2	First one-versus-rest in EFC
FPA (Wang <i>et al.</i> , 2013)	Single model classifiers	Incremental con-based optimization	Weight vector update	No	No	No	2	
Gen-Smart-EFS (GS-EFS) (Lughofer <i>et al.</i> , 2013)	Generalized TS fuzzy system	Evolving clustering (extended eVQ)	RFWLS	Yes	Yes, cons. only	Yes (smooth)	1–2	First joint concept dim. red. for gen. rules + pruning
GENEFIS (Pratama <i>et al.</i> , 2014b)	Generalized TS fuzzy system	Gen. adaptive resonance theory + stat. influence	FWGRLS	Yes	No	Yes (crisp)	3–4	Gen. rules + online dim. red.

HAW-NFS (Bodyanskiy and Vynokurova, 2013)	Neuro-fuzzy system	inc opt. [mod. of (Kaczmarz, 1993)]	RLS (global)	No	No	No	3	First wavelet-based neuro-fuzzy
LOLIMOT inc. (Hamelner and Jakubek, 2013)	Tree-based structure	Node replacement (recursive split)	Recursive nonlinear least squares	No	No	No	1–2	First approach for an incremental LOLIMOT (Nelles, 2001)
PANFIS (Pratama <i>et al.</i> , 2014a)	Generalized TS fuzzy system	Extended Self-Organizing Map, statistical influence	FWGRLS	Yes	No	No	2	First proj. concept of gen. rules, stability proofs
rGK (Dovzan and Skrjanc, 2011)	TS fuzzy system	Recursive clustering (GK)	RLS (global), RFWLS (local)	No	Yes, cons. only	No	3–6	No rule evolution
(E)SAFIS (Rong <i>et al.</i> , 2014, 2006)	TS fuzzy system (MIMO)	Statistical influence, distance criterion	RLS (global), inc. stability with Lyapunov	Yes	No	No	3	One of the pioneering methods (2005–2006)
SAFIS-Class (Rong <i>et al.</i> , 2011)	Single model using TS fuzzy system	Statistical influence, distance criterion	RLS (global)	Yes	No	No	6	
SEIT2FNN (Juang and Tsao, 2008)	Type-2 TS fuzzy system	Incremental steepest descent	RLS (global)	No	Yes, cons. only	No	3	First evolving type-2 fuzzy system
SOFMLS (Rubio, 2009)	Mamdani	Evolving clustering (nearest neighborhood)	Modified RLS with increased stability	Yes	No	No	4	First approach of an evolving Mamdani fuzzy system
SOFNN (imp) (Leng <i>et al.</i> , 2005, 2012)	Neuro-fuzzy system	Coverage and system error criteria, rule enlargement	Modified RLS (global) with weights	Yes	No	No	2	One of the pioneering methods (2005)

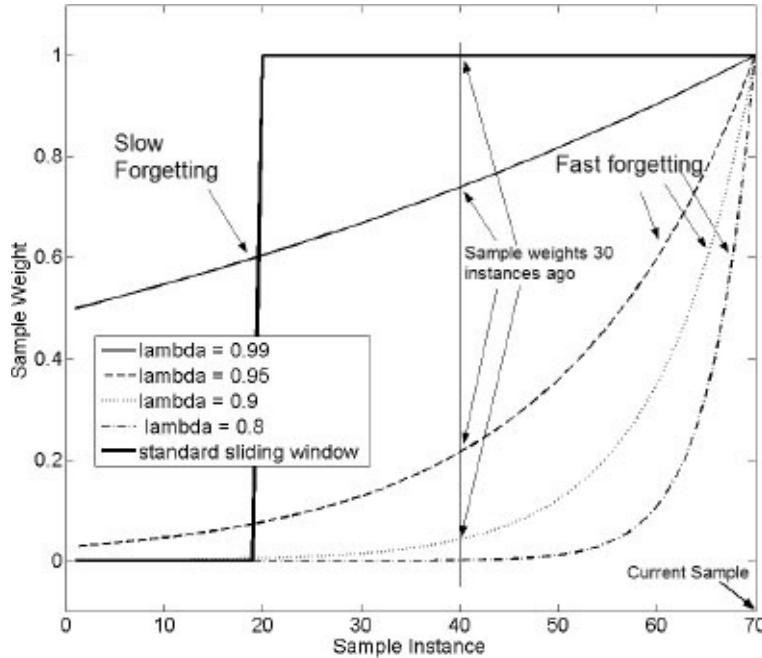
This section is dedicated to a summary of recent developments in stability, reliability and robustness of EFS which can be generically used in combination with most of the approaches listed in Tables 3.1 and 3.2.

3.4.1. Drift Handling in Streams

Drifts in data streams refer to a gradual evolution of the underlying data distribution and therefore of the learning concept over time (Tsymbal, 2004; Widmer and Kubat, 1996) and are frequent occurrences in nowadays non-stationary environments (Sayed-Mouchaweh and Lugofer, 2012). Drifts can happen because the system behavior, environmental conditions or target concepts dynamically change during the online learning process, which makes the relations, concepts contained in the old data samples obsolete. Such situations are in contrary to new operation modes or system states which should be integrated into the models with the same weight as the older ones in order to extend the models, but keeping the relations in states seen before untouched (still valid for future predictions). Drifts, however, usually mean that the older learned relations (as parts of a model) are not valid any longer and thus should be incrementally out-weighted, ideally in a smooth manner to avoid catastrophic forgetting (French, 1999; Moe-Helgesen and Strand, 2005).

A smooth forgetting concept for consequent learning employing the idea of exponential forgetting (Aström and Wittenmark, 1994), is used in approximately half of

the EFS approaches listed in [Tables 3.1](#) and [3.2](#) (refer to Column #6). The strategy in all of these is to integrate a forgetting factor $\lambda \in [0, 1[$ for strengthening the influence of newer samples in the Kalman gain γ —see Equation (31). [Figure 3.7](#) shows the weight impact of samples obtained for different forgetting factor values. This is also compared to a standard sliding window technique, which weighs all samples up to a certain point of time in the past equally, but forgets all others before completely \rightarrow non-smooth forgetting. This variant is also termed as decremental learning (Bouillon *et al.*, 2013; Cernuda *et al.*, 2014), as the information contained in older samples falling out of the sliding window is fully unlearned (= decremented) from the model. The effect of this forgetting factor integration is that changes of the target concept in regression problems can be tracked appropriately, i.e., a movement and shaping of the current regression surface towards the new output behavior is enforced, see Lugofer and Angelov (2011). Decremental learning may allow more flexibility (only the latest N samples are really used for model shaping), but increases the likelihood of catastrophic forgetting.



[Figure 3.7](#): Smooth forgetting strategies achieving different weights for past samples; compared to a sliding window with fixed width \rightarrow complete forgetting of older samples (decremental learning).

Regarding a drift handling in the antecedent part, several techniques may be used such as a reactivation of rule centers and spreads from a converged situation by an increase of the learning gain: this is conducted in Lugofer and Angelov (2011) for the two EFS approaches eTS and FLEXFIS and has the effect that rules are helped out from their stucked, converged positions to cover the new data cloud appearance of the drifted situation. In eFuMo, the forgetting in antecedent learning is integrated as the degree of the weighted movement of the rule centers \vec{c}_i towards a new data sample \vec{x}_{N+1} :

$$\begin{aligned} \vec{c}_i(N+1) &= \vec{c}_i(N) + \Delta\vec{c}_i(N+1) \quad \text{with } \Delta\vec{c}_i(N+1) \\ &= \frac{\mu_i(\vec{x}_{N+1})^\eta (\vec{x}_{N+1} - \vec{c}_i(N))}{s_i(N+1)}, \end{aligned} \tag{39}$$

with $s_i(N+1) = \lambda s_i(N) + \mu_i (\vec{x}_{N+1})^\eta$ the sum of the past memberships $\mu_i(\vec{x}_j)$, $j = 1, \dots, N$, η the fuzziness degree also used as parameters in fuzzy c-means, and λ the forgetting factor. Forgetting is also integrated in the inverse covariance matrix and determinant update defining the shape of the clusters. All other EFS techniques in [Tables 3.1](#) and [3.2](#) do not embed an antecedent forgetting.

An important investigation is the question when to trigger forgetting and when to apply the conventional life-long learning concept (all samples equally weighted) (Hamker, 2001). In Shaker and Lughofer (2014), it could be analyzed that when using a permanent (fixed) forgetting factor respectively an increased model flexibility in case when no drift happens, the accuracy of the evolved models may decrease over time. Thus, it is necessary to install a drift indication concept, which is able to detect drifts and in ideal cases also to quantify their intensity levels; based on these, it is possible to increase or decrease the degree of forgetting, also termed as *adaptive forgetting*. A recent approach for EFS which performs this by using an extended version of Page–Hinkley test (Mouss *et al.*, 2004), (a widely known and respected test in the field of drift handling in streams (Gama, 2010; Sebastiao *et al.*, 2013) is demonstrated in Shaker and Lughofer (2014). It is also the first attempt to localize the drift intensity by quantifying drift effects in different local parts of the features space with different intensities and smoothly: EFS is a perfect model architecture to support such a local smooth handling (fuzzy rules with certain overlaps).

3.4.2. Online Curse of Dimensionality Reduction

For models including localization components as is the case of evolving fuzzy systems (in terms of rules), it is well known that *curse of dimensionality* is very severe in case when a high number of variables are used as model inputs (Pedrycz and Gomide, 2007), e.g., in large-scale time-series, recorded in multi-sensor networks (Chong and Kumar, 2003). This is basically because in high-dimensional spaces, someone cannot speak about locality any longer (on which these types of models rely), as all samples are moving to the edges of the joint feature space — see Hastie *et al.* (2009) and [Chapter 1](#) for a detailed analysis of this problem.

Therefore, the reduction of the dimensionality of the learning problem is highly desired. In data stream sense, to ensure an appropriate reaction onto the system dynamics, the feature reduction should be conducted online and be open for *anytime changes*. A possibility is to track the importance levels of features over time and to cut out that ones which become unnecessary — as has been used in connection with EFS for regression problems in Angelov (2010); Pratama *et al.* (2014b) and for classification problems in a first attempt in Bouchachia and Mittermeir (2006). However, features which are unimportant at an earlier point of time may become important at a later stage (*feature reactivation*). This means that crisply cutting out some features with the usage of online feature selection and ranking approaches such as Li (2004); Ye *et al.* (2005) can fail to

represent the recent feature dynamics appropriately. Without a re-training cycle, which, however slows down the process and causes additional sliding window parameters, this would lead to discontinuities in the learning process (Lughofer, 2011b), as parameters and rule structures have been learnt on different feature spaces before.

An approach which is addressing input structure changes incrementally on-the-fly is presented in Lughofer (2011c) for classification problems using classical single model and one-versus-rest based multi-model architectures (in connection with FLEXFIS-Class learning engine). It operates on a global basis, hence features are either seen as important or unimportant for the whole model. The basic idea is that feature weights $\lambda_1, \dots, \lambda_p \in [0, 1]$ for the p features included in the learning problem are calculated based on a stable separability criterion (Dy and Brodley, 2004):

$$J = \text{trace}(S_w^{-1} S_b), \quad (40)$$

with S_w the within scatter matrix modeled by the sum of the covariance matrices for each class, and S_b the between scatter matrix, modeled by the sum of the degree of mean shift between classes. The criterion in Equation (40) is applied (1). Dimension-wise to see the impact of each feature separately — note that in this case it reduces to a ratio of two variances — and (2). For the remaining $p - 1$ feature subspace in order to gain the quality of separation when excluding each feature. In both cases, p criteria J_1, \dots, J_p according to Equation (40) are obtained. For normalization purposes to $[0, 1]$, finally the feature weights are defined by:

$$\lambda_j = 1 - \frac{J_j - \min_{j=1,\dots,p}(J_j)}{\max_{j=1,\dots,p}(J_j) - \min_{j=1,\dots,p}(J_j)}. \quad (41)$$

To be applicable in online learning environments, updating the weights in incremental mode is achieved by updating the within-scatter and between-scatter matrices using the recursive covariance matrix formula (Hisada *et al.*, 2010). This achieves a smooth change of feature weights = feature importance levels over time with new incoming samples. Features may become out-weighted (close to 0) and reactivated (weights significantly larger than 0) at a later stage without “disturbing” the parameter and rule structure learning process. Hence, this approach is also denoted as *smooth and soft online dimension reduction*—the term *softness* comes from the decreased weights instead of a crisp deletion. Down-weighted features then play a marginal role during the learning process, e.g., the rule evolution criterion relies more on the highly weighted features.

The feature weights concept has been recently employed in Lughofer *et al.* (2014), in the context of data stream regression problems, there with the usage of generalized rules as defined in [Section 3.2.2.2](#), instead of axis-parallel ones as used in Lughofer (2011c). The features weights are calculated by a combination of future-based expected statistical contributions of the features in all rules and their influences in the rules’ hyper-planes (measured in terms of gradients), see Lughofer *et al.* (2014).

3.4.3. Incremental Smoothing

Recently, a concept for incremental smoothing of consequent parameters has been introduced in Rosemann *et al.* (2009), where a kind of *incremental regularization* is conducted after each learning step. This has the effect to decrease the level of over-fitting whenever the noise level in the data is high. Indeed, when applying the local recursive least squares estimator as in Equations (30)–(32) and some of its modifications, the likelihood of over-fitting is small due to the enforcement of the local approximation and interpretation property [as analyzed in Angelov *et al.* (2008)], but still may be present. The approach in Rosemann *et al.* (2009) accomplishes the smoothing by correcting the consequent functions of the updated rule(s) by a template T measuring the violation degree subject to a meta-level property on the neighboring rules.

Finally, it should be highlighted that this strategy assures smoother consequent hyperplanes over nearby lying or even touching rules, thus increases the likelihood of further rule reduction through extended simplicity assurance concepts (adjacency, homogeneity, trend-continuation criteria) as discussed in Lughofe (2013) and successfully applied to obtain compact rule bases in data stream regression problems in Lughofe *et al.* (2013), employing generalized fuzzy rules, defined in [Section 3.2.2.2](#).

3.4.4. Convergence Analysis/Ensurance

Another important criterion when applying EFS is some sort of convergence of the parameters included in the fuzzy systems over time (in case of a regular stream without a drift etc.)—this accounts for the stability aspect in the stability-plasticity dilemma (Hamker, 2001), which is important in the life-long learning context. This is for instance accomplished in the FLEXFIS approach, which, however only guarantees a sub-optimality subject to a constant (thus guaranteeing finite solutions) due to a quasi-monotonic decreasing intensity of parameter updates, but is not able to provide a concrete level of this sub-optimality, see Lughofe (2008) and Lughofe (2011b) and [Chapter 3](#). In the approach by Rubio (2010), a concrete upper bound on the identification error is achieved by the modified least squares algorithm to train both, parameters and structures, is achieved with the support of Lyapunov function. The upper bound depends on the actual certainty of the model output. Another approach which is handling the problem of constraining the model error while assuring parameter convergence simultaneously is applied within the PANFIS learning engine (Pratama *et al.*, 2014a): This is achieved with the usage of an extended recursive least squares approach.

3.4.5. Reliability

Reliability deals with the *interpretation of the model predictions* in terms of classification statements, quantification outputs or forecasted values in time series. Reliability points to the trustworthiness of the predictions of current query points which may be basically

influenced by two factors:

- The quality of the training data w.r.t data stream samples seen so far.
- The nature and localization of the current query point for which a prediction should be obtained.

The trustworthiness/certainty of the predictions may support/influence the users/operators during a final decision finding — for instance, a query assigned to a class may cause a different user's reaction when the trustworthiness about this decision is low, compared to when it is high. In this sense, it is an essential add-on to the actual model predictions which may also influence its further processing.

The first factor basically concerns the noise level in measurement data. This also may cover aspects in the direction of uncertainties in users' annotations in classification problems: a user with lower experience level may cause more inconsistencies in the class labels, causing overlapping classes, finally increasing conflict cases (see below); similar occasions may happen in case when several users annotate samples on the same systems, but have different opinions in borderline cases.

The second factor concerns the position of the current query point with respect to the definition space of the model. A model can show a perfect trend with little uncertainty, but a new query point appears far away from all training samples seen so far, yielding a severe degree of extrapolation when conducting the model inference process. In a classification context, a query point may also fall close in a highly overlapped region or close to the decision boundary between two classes. The first problem is denoted as *ignorance*, the second as *conflict* (Hüllermeier and Brinker, 2008; Lughofer, 2012a). A visualization of these two occasions is shown in Figure 3.8(a) (for conflict) and 3.8(b) (for ignorance). The conflict case is due to a sample falling in-between two classes and the ignorance case due to a query point falling outside the range of training samples, which is indeed linearly separable, but by several possible decision boundaries [also termed as the *variability of the version space* (Hüllermeier and Brinker, 2008)].

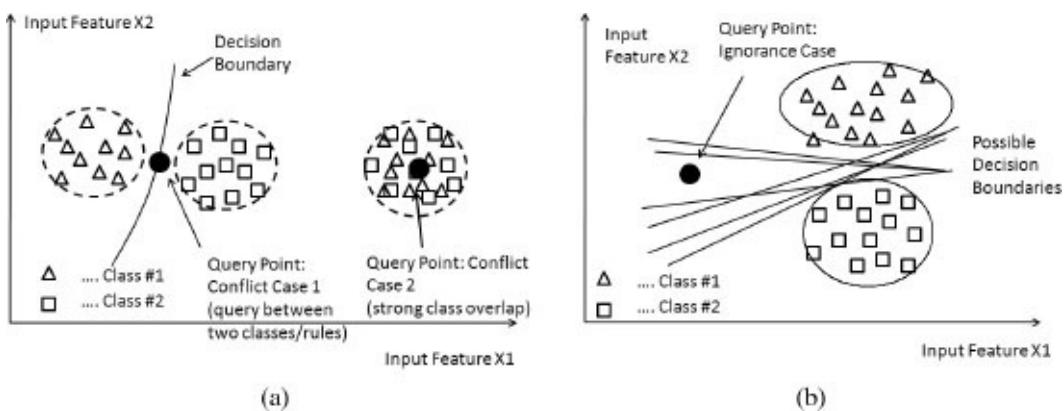


Figure 3.8: (a) Two conflict cases: query point falls inbetween two distinct classes and within the overlap region of two classes; (b) Ignorance case as query point lies significantly away from the training samples, thus increasing the *variability of the version space* (Hüllermeier and Brinker, 2008); in both figures, rules modeling the two separate classes are shown by an ellipsoid, the decision boundaries indicated by solid lines.

In the regression context, the estimation of parameters through RWFLS and modifications usually can deal with high noise levels in order to find a nonoverfitting trend of the approximation surface. However, it is not possible to represent uncertainty levels of the model predictions later on. These can be modeled by the so-called error bars or confidence intervals (Smithson, 2003). In EFS, they have been developed based on parameter uncertainty in Lughofer and Guardiola (2008a) [applied to online fault detection in Serdio *et al.* (2014a)] and in an extended scheme in Skrjanc (2009) (for chemometric calibration purposes). The latter is based on a funded deduction from statistical noise and quantile estimation theory (Tschumitschew and Klawonn, 2012). Both are applicable in connection with local (LS) learning of TS consequent parameters.

In the classification context, conflict and ignorance can be reflected and represented by means of fuzzy classifiers in a quite natural way (Hühn and Hüllermeier, 2009). These concepts have been only recently tackled once in the field of evolving fuzzy classifiers (EFC), see Lughofer and Buchtala (2013), where multiple binary classifiers are trained in an all-pairs context for obtaining simper decision boundaries in multi-class classification problems (see [Section 3.2.6.3](#)). On a single rule level, the confidence in a prediction can be obtained by the confidence in the different classes coded into the consequents of the rules having the form of Equation (17). This provides a perfect conflict level (close to 0.5 → high conflict, close to 1 → low conflict) in case of overlapping classes within a single rule. If a query point \vec{x} falls in-between rules with different majority classes (different maximal confidence levels), then the extended weighted classification scheme in Equation (19) is requested to represent a conflict level. If the confidence in the final output class L , $conf_{f_L}$, is close to 0.5, conflict is high, when it is close to 1, conflict is low. In case of all-pairs architecture, Equation (19) can be used to represent conflict levels in the binary classifiers. Furthermore, an overall conflict level on the final classifier output is obtained by Lughofer and Buchtala (2013):

$$conflict_{deg} = \frac{score_k}{score_k + score_l}, \quad (42)$$

with k and l the classes with the highest scores.

The ignorance criterion can be resolved in a quite natural way, represented by a degree of extrapolation, thus:

$$Ign_{deg} = 1 - \max_{i=1}^C \mu_i(\vec{x}), \quad (43)$$

with C the number of rules currently contained in the evolved fuzzy classifier. In fact, the degree of extrapolation is a good indicator of the degree of ignorance, but not necessarily sufficient, see Lughofer (2012a), for an extended analysis and further concepts. However, integrating the ignorance levels into the preference relation scheme of all-pairs evolving fuzzy classifiers according to Equation (24) for obtaining the final classification statement,

helped to boost the accuracies of the classifier significantly, as then classifiers which show a strongly extrapolated situation in the current query are down-weighted towards 0, thus masked-out, in the scoring process. This leads to an out-performance of incremental machine learning classifiers from MOA framework¹¹ (Bifet *et al.*, 2010) on several large-scale problems, see Lughofer and Buchtala (2013). The overall ignorance level of an all-pairs classifier is the minimal ignorance degree calculated by Equation (43) over all binary classifiers.

3.5. Interpretability

Improved transparency and interpretability of the evolved models may be useful in several real-world applications where the operators and experts intend to gain a deeper understanding of the interrelations and dependencies in the system. This may enrich their knowledge and enable them to interpret the characteristics of the system on a deeper level. Concrete examples are decision support systems or classification systems, which sometimes require the knowledge why certain decisions have been made, e.g., see Wetter (2000): Insights into these models may provide answers to important questions (e.g., providing the health state of a patient) and support the user in taking appropriate actions. Another example is the substitution of expensive hardware with *soft sensors*, referred to as *eSensors* in an evolving context (Angelov and Kordon, 2010a; Macias-Hernandez and Angelov, 2010): The model has to be linguistically or physically understandable, reliable, and plausible to an expert, before it can be substituted for the hardware. In often cases, it is beneficial to provide further insights into the control behavior (Nelles, 2001).

Interpretability, apart from pure complexity reduction, has been addressed very little in the evolving systems community so far (under the scope of data stream mining). A recent position paper published in *Information Sciences* journal Lughofe (2013) summarizes the achievements in EFS, provides avenues for new concepts as well as concrete new formulas and algorithms and points out open issues as important future challenges. The basic common understanding is that complexity reduction as a key prerequisite for compact and therefore transparent models is handled in most of the EFS approaches, which can be found nowadays in literature (please also refer to Column “Rule pruning” in [Tables 3.1](#) and [3.2](#)), whereas other important criteria [known to be important from conventional batch offline design of fuzzy systems (Casillas *et al.*, 2003; Gacto *et al.*, 2011)], are more or less loosely handled in EFS. These criteria include:

- Distinguishability and Simplicity
- Consistency
- Coverage and Completeness
- Local Property and Addition-to-One-Unity
- Feature Importance Levels
- Rule Importance Levels
- Interpretation of Consequents
- Interpretability of Input–Output Behavior
- Knowledge Expansion

Distinguishability and *simplicity* are handled under the scope of complexity reduction, where the difference between these two lies in the occurrence of the degree of overlap of

rules and fuzzy sets: Redundant rules and fuzzy sets are highly overlapping and therefore indistinguishable, thus should be merged, whereas obsolete rules or close rules showing similar approximation/classification trends belong to an unnecessary complexity which can be simplified (due to pruning). Figure 3.9 visualizes an example of a fuzzy partition extracted in the context of house price estimation (Lughofer *et al.*, 2011b), when conducting native precise modeling (left) and when conducting some simplification steps according to merging, pruning and constrained-based learning (right). Only in the right case, it is possible to assign linguistic labels to the fuzzy sets and hence to achieve interpretation quality.

Consistency addresses the problem of assuring that no contradictory rules, i.e., rules which possess similar antecedents but dissimilar consequents, are present in the system. This can be achieved by merging redundant rules, i.e., those one which are similar in their antecedents, with the usage of the participatory learning concept introduced by Yager (1990). An appropriate merging of the linear parameter vectors \vec{w} is given by Lughofer and Hüllermeier (2011):

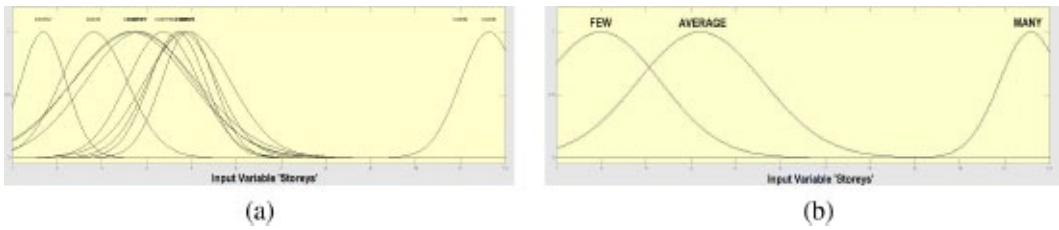


Figure 3.9: (a) Weird un-interpretable fuzzy partition for an input variable in house pricing; (b) The same partition achieved when conducting merging, pruning options of rules and sets during the incremental learning phase → assignments of linguistic labels possible.

$$\vec{w}_{\text{new}} = \vec{w}_{R_1} + \alpha \cdot \text{Cons}(R_1, R_2) \cdot (\vec{w}_{R_2} - \vec{w}_{R_1}), \quad (44)$$

where $\alpha = k_{R_2}/(k_{R_1} + k_{R_2})$ represents the *basic learning rate* with k_{R_i} the support of the more supported rule R_i and $\text{Cons}(R_1, R_2)$ the *compatibility measure* between the two rules within the participatory learning context. The latter is measured by a consistency degree between antecedent and consequents of the two rules. It relies on the exponential proportion between rule antecedent similarity degree (overlap) and rule consequent similarity degree.

Coverage and *completeness* refer to the problem of a well-defined coverage of the input space by the rule-base. Formally, ϵ -completeness requires that for each new incoming data sample there exists at least one rule to which this sample has a membership degree of at least ϵ . A specific re-adjustment concept of fuzzy sets and thus rules is presented in Lughofer (2013), which restricts the re-adjustment level in order to keep the accuracy high. An alternative, more profound option for data stream regression problems, is offered as well in Lughofer (2013), which integrates a punishment term for ϵ -completeness into the least squares optimization problem. Incremental optimization techniques based on gradients of the extended optimization function may be applied in

order to approach but not necessarily fully assure ϵ -completeness. On the other hand, the joint optimization guarantees a reasonable tradeoff between model accuracy and model coverage of the feature space.

Local property and *addition-to-one-unity* have been not considered so far, but will go a step further by ensuring fuzzy systems where only maximal 2^p rules fire at the same time (Bikdash, 1999), with p the input dimensionality. From our point of experience, this cannot be enforced by significantly loosing some model accuracy, as it requires significant shifts of rules away from the real position of the data clouds/density swarms.

Feature importance levels are an outcome of the online dimensionality reduction concept discussed in [Section 3.4.2](#). With their usage, it is possible to obtain an interpretation on input structure level which features are more important and which ones are less important. Furthermore, they may also lead to a reduction of the rule lengths thus increasing rule transparency and readability, as features with weights smaller than ϵ have a very low impact on the final model output and therefore can be eliminated from the rule antecedent and consequent parts when showing the rule-base to experts/operators.

Rule importance levels could serve as essential interpretation component as they tend to show the importance of each rule in the rule-base. Furthermore, rule weights may be used for a smooth rule reduction during learning procedures, as rules with low weights can be seen as unimportant and may be pruned or even re-activated at a later stage in an online learning process (*soft rule pruning mechanism*). This strategy may be beneficial when starting with an expert-based system, where originally all rules are fully interpretable (as designed by experts/users), however, some may turn out to be superfluous over time for the modeling problem at hand (Lughofer, 2013). Furthermore, rule weights can be used to handle inconsistent rules in a rule-base, see e.g., Pal and Pal (1999); Cho and Park (2000), thus serving for another possibility to tackle the problem of consistency (see above). The usage of rule weights and their updates during incremental learning phases, was, to our best knowledge, not studied so far in the evolving fuzzy community. In Lughofer (2013), a first concept was suggested to adapt the rule weights, integrated as nonlinear parameters in the fuzzy systems architecture, based on incremental optimization procedures, see also [Section 3.3.2](#).

Interpretation of consequents is assured by nature in case of classifiers with consequent class labels plus confidence levels; in case of TS fuzzy systems, it is assured as soon as local learning of rule consequents is used (Angelov *et al.*, 2008; Lughofer, 2011b) [Chapter 2](#) and Lughofer (2013). Please also refer to [Section 3.3.1](#): Then, a snuggling of the partial linear trends along the real approximation surface is guaranteed, thus giving rise in which parts of the feature space the model will react in which way and intensity (gradients of features).

Interpretability of Input–Output Behavior refers to the understanding which output(s) will be produced when showing the system concrete input queries. For instance, a model

with constant output has a maximal input–output interpretability (as being very predictable what outcome will be produced for different input values), however, usually suffers from predictive accuracy (as long as the behavior of the system to be modeled is non-constant). Most actively firing rules can be used as basis for this analysis.

Knowledge expansion refers to the automatic integration of new knowledge arising during the online process on demand and on-the-fly, also in order to expand the interpretation range of the models, and is handled by all conventional EFS approaches through rule evolution and/or splitting, see [Tables 3.1](#) and [3.2](#).

3.5.1. Visual Interpretability

Visual interpretability refers to an interesting alternative to *linguistic interpretability* (as discussed above), namely, to the representation of a model in a graphical form. In our context, this approach could be especially useful if models evolve quickly, since monitoring a visual representation might then be easier than following a frequently changing linguistic description. Under this scope, alternative “interpretability criteria” may then become interesting which are more dedicated to the timely development of the evolving fuzzy model — for instance, a trajectory of rule centers showing their movement over time, or trace paths showing birth, growth, pruning and merging of rules; first attempts in this direction have been conducted in Hentzgen *et al.* (2013), employing the concept of *rule chains*. These have been significantly extended in Hentzgen *et al.* (2014) by setting up a visualization framework with a grown-up user front-end (GUI), integrating various similarity, coverage and overlap measures as well as specific techniques for an appropriate catchy representation of high-dimensional rule antecedents and consequents. Internally, it uses the FLEXFIS++ approach (Lughofer, 2012b) as incremental learning engine.

3.6. Usability and Applications

In order to increase the usability of evolving fuzzy systems, several issues are discussed in this section, ranging from the reduction of annotation efforts in classification settings through a higher plug-and-play capability (more automatization, less tuning) to the decrease of computational resources and as well as to online evaluation measures for supervising modeling performance. At the end of this section, a list of real-world applications making use of evolving fuzzy systems will be discussed.

Finally, the increase of the usability together with the assurance of interpretability serves as basis for a successful future development of the *human-inspired evolving machines (HIEM)* concept as discussed in Lughofer (2011a), which is expected to be the *next generation of evolving intelligent systems*¹² —the aim is to enrich the pure machine learning systems with human knowledge and feelings, and to form a joint concept of *active learning and teaching* in terms of a higher-educated computational intelligence useable in artificial intelligence.

3.6.1. Single-Pass Active Learning

3.6.1.1. For classification problems

In online classification tasks, all evolving and incremental classifier variants require provision of the *ground truth* in form of *true class labels* for incoming samples to guarantee smooth and well-defined updates for increased classifiers' performance. Otherwise, classifiers' false predictions self-reinforce and are back-propagated into their structure and parameters, leading to a deterioration of their performance over time (Gama, 2010; Sayed-Mouchaweh and Lughofer, 2012). The problem, however, is that the true class labels of new incoming samples are usually not included in the stream neither provided automatically by the system. Mostly, operators or experts have to provide the ground truth which requires considerable manpower and fast manual responses in case of online processes. Therefore, in order to attract the operators and users to work and communicate with the system, thus to assure classifier usability in online mode, decreasing the number of required samples for evolving and improving a classifier over time is essential.

This task can be addressed by *active learning* (Settles, 2010), a technique where the learner itself has control over which samples are used to update the classifiers (Cohn *et al.*, 1994). Conventional active learning approaches operate fully in batch mode: (1) New samples are selected iteratively and sequentially from a pool of training data; (2) The true class labels of the selected samples are queried from an expert; and (3) Finally, the classifier is re-trained based on the new samples together with those previously selected. In an online setting, such iteration phases over a pool of data samples are not practicable. Thus, a requirement is that the sample selection process operates autonomously in a

single-pass manner, omitting time-intensive re-training phases.

Several first attempts have been made in connection with linear classifiers (Chu *et al.*, 2011; Sculley, 2007). A nonlinear approach which employs both, single fuzzy model architecture with extended confidence levels in the rule consequents [as defined in Equation (17)] and the all-pairs concept as defined in [Section 3.2.6.3](#), is demonstrated in Lughofe (2012a). There, the actual evolved fuzzy classifier itself decides for each sample whether it helps to improve the performance and, when indicated, requests the true class label for updating its parameters and structure. In order to obtain the certainty level for each new incoming data sample (query point), two reliability concepts are explored: *conflict* and *ignorance*, both motivated and explained in detail in [Section 3.4.5](#): when one of the two cases arises for a new data stream sample, a class label is requested from operators. A common understanding based on several results on high-dimensional classification streams (binary and multi-class problems) was that a very similar tendency of accuracy trend lines over time can be achieved when using only 20–25% of the data samples in the stream for classifier updates, which are selected based on the single-pass active learning policy. Upon random selection, the performance deteriorates significantly. Furthermore, a batch active learning scheme based on re-training cycles using SVMs classifiers (Schölkopf and Smola, 2002) (lib-SVM implementation¹³) could be significantly out-performed in terms of accumulated one-step-ahead accuracy (see [Section 3.6.4](#) for its definition).

3.6.1.2. For regression problems

Online active learning may be also important in case of regression problems whenever for instance the measurements of a supervised target are quite costly to obtain. An example is the gathering of titration values within a spin-bath analytics at a viscose production process (courtesy to Lenzing GmbH), which are for the purpose to supervise the regulation of the substances H_2SO_4 , Na_2SO_4 , and $ZnSO_4$ (Cernuda *et al.*, 2014). There, active learning is conducted in *incremental and decremental stages* with the help of a sliding window-based approach (sample selection for incoming as well as outgoing points) and using TS fuzzy models connected with PLS. It indeed exceeds the performance of conventional equidistant and costly model updates, but is not fully operating in a single-pass manner (a window of samples is required for re-estimation of statistics, etc.). Single-pass strategies have been, to our best knowledge, not handled in data stream regression problems, neither in connection with evolving fuzzy systems.

3.6.2. Toward a Full Plug-and-Play Functionality

The *plug-and-play functionality* of online incremental learning methods is one of the most important properties in order to prevent time-intensive pre-training cycles in batch mode and to support an easy usability for experts and operators. The situation in the EFS

community is that all EFS approaches as listed in [Tables 3.1](#) and [3.2](#) allow the possibility to incrementally learn the models from scratch. However, all of these require at least one or a few learning parameters guiding the engines to correct, stable models — see Column #8 in these tables. Sometimes, a default parametrization exists, but is sub-optimal for upcoming new future learning tasks, as having been optimized based on streams from past processes and application scenarios. Cross-validation (Stone, 1974) or boot-strapping (Efron and Tibshirani, 1993) may help to guide the parameters to good values during the start of the learning phase (carried out on some initial collected samples), but, apart that these iterative batch methods are eventually too slow (especially when more than two parameters need to be adjusted), a stream may turn out to change its characteristics later on (e.g., due to a drift, see [Section 3.4.1](#)). This usually would require a dynamic update of the learning parameters, which is not supported by any EFS approach so far and has to be specifically developed in connection with the concrete learning engine.

An attempt to overcome such an unlucky or undesired parameter setting is presented in Lughofe and Sayed-Mouchaweh (2015) for evolving cluster models, which, however, may be easily adopted to EFS approaches, especially to those ones performing an incremental clustering process for rule extraction. Furthermore, the prototype-based clusters in Lughofe and Sayed-Mouchaweh (2015) are axis-parallel defining local multivariate Gaussian distributions in the feature space. Thus, when using Gaussian fuzzy sets in connection with product t-norm, the same rule shapes are induced. The idea in Lughofe and Sayed-Mouchaweh (2015) is based on dynamic split-and-merge concepts for clusters (rules) which are either moving together forming a homogenous joint region (\rightarrow merge requested) or are internally containing two or more distinct data clouds, thus already housing some internal heterogeneity (\rightarrow split requested), see [Figure 3.10](#) (Cluster #4) for an example, also showing the internal structure of Cluster #4 in the right image. Both occurrences may arise either due to the nature of the stream or often due to a wrong parametrization of the learning engine (e.g., a too low threshold such that new rules are evolved too early). The main difficulty lies on the identification of when to merge and when to split: parameter-free options are discussed in Lughofe and Sayed-Mouchaweh (2015). Opposed to other *joint* merging and splitting concepts in some EFS approaches, one strength of the approach in Lughofe and Sayed-Mouchaweh (2015) is that it can be used independently from the concrete learning engine. The application of the unsupervised automatic splitting and merging concepts to supervised streaming problems under the scope of EFS/EFC may thus be an interesting and fruitful future challenge.

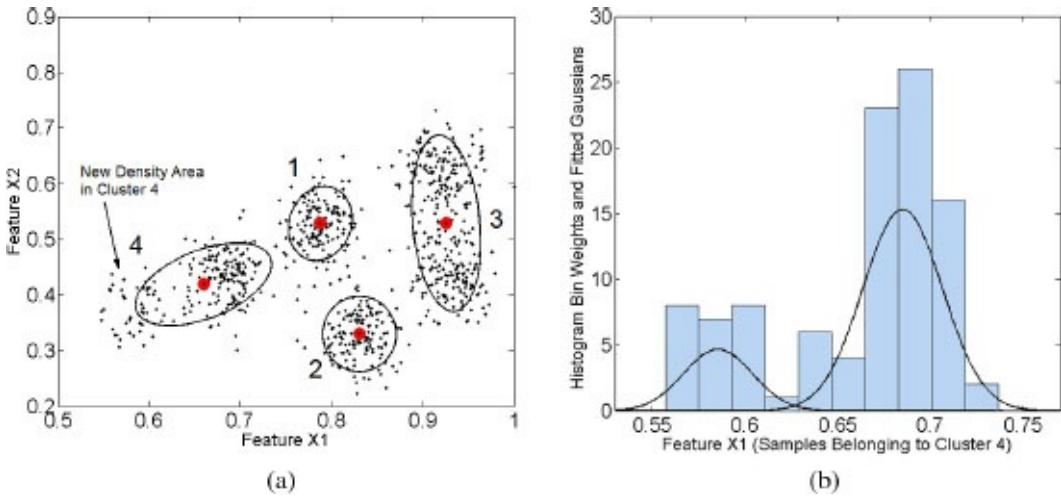


Figure 3.10: (a) The cluster structure after 800 samples, Cluster #4 containing already a more distinct density area; (b) Its corresponding histogram along Feature X1, showing the clear implicit heterogenous nature.

3.6.3. On Improving Computational Demands

When dealing with online data stream processing problems, usually the computational demands required for updating the fuzzy systems are an essential criteria whether to install and use the component or not. It can be a kick-off criterion, especially in real-time systems, where the update is expected to terminate in real-time, i.e., before the next sample is loaded, the update cycle should be finished. Also, the model predictions should be in-line the real-time demands, but these are known to be very fast in case of fuzzy inference scheme (significantly below milliseconds) (Kruse *et al.*, 1994; Pedrycz and Gomide, 2007; Piegat, 2001). An extensive evaluation and especially a comparison of computational demands for a large variety of EFS approaches over a large variety of learning problems with different number of classes, different dimensionality etc. is unrealistic, also because most of the EFS approaches are hardly downloadable or to obtain from the authors. A loose attempt in this direction has been made by Komijani *et al.* (2012), where they classify various EFS approaches in terms of computation speed into three categories: *low, medium, high*.

Interestingly, the consequent update is more or less following the complexity of $O(Cp^2)$ with p the dimensionality of the feature space and C the number of rules, when local learning is used (as for most EFS approaches, compare in [Tables 3.1](#) and [3.2](#)), and following the complexity of $O((Cp)^2)$ when global learning is applied. The quadratic terms p^2 resp. $(Cp)^2$ are due to the multiplication of the inverse Hessian with the actual regressor vector in Equation (31), and because their sizes are $(p+1) \times (p+1)$ and $p+1$ in case of local learning (storing the consequent parameters of one rule) resp. $(C(p + 1)) \times (C(p + 1))$ and $C(p + 1)$ in case of global learning (storing the consequent parameters of all rules). Regarding antecedent learning, rule evolution and pruning, most of the EFS approaches try to be restricted to have at most cubic complexity in terms of the number of rules plus the number of inputs. This may guarantee some sort of smooth termination in an online process, but it is not a necessary prerequisite and has to be inspected for the particular

learning problem at hand.

However, some general remarks on the improvement of computational demands can be given: first of all, the reduction of *unnecessary complexity* such as merging of redundant overlapping rules and pruning of obsolete rules (as discussed in [Section 3.3.4](#)) is always beneficial for speeding up the learning process. This also ensures that fuzzy systems are not growing forever, thus restricted in their expansion and virtual memory requests. Second, some fast version of incremental optimization techniques could be adopted to fuzzy systems estimation, for instance, there exists a fast RLS algorithm for recursively estimating linear parameters in near linear time ($O(n \log(n))$), but with the costs of some stability and convergence, see Douglas (1996); Gay (1996) or Merched and Sayed (1999). Another possibility for decreasing the computation time for learning is the application of active learning for exquisitely selecting only a subset of samples, based on which the model will be updated, please also refer to [Section 3.6.1](#).

3.6.4. Evaluation Measures

Evaluation measures may serve as indicators about the actual state of the evolved fuzzy systems, pointing to its accuracy and trustworthiness in predictions. In a data streaming context, the temporal behavior of such measures plays an essential role in order to track the model development over time resp. to realize down-trends in accuracy at an early stage (e.g., caused by drifts), and to react appropriately (e.g., conducting a re-modeling phase, changes at the system setup, etc.). Furthermore, the evaluation measures are indispensable during the development phase of EFS approaches. In literature dealing with incremental and data-stream problems (Bifet and Kirkby, 2011), basically three variants of measuring the (progress of) model performance are suggested:

- Interleaved-test-and-then-train.
- Periodic hold out test.
- Fully-train-and-then-test.

Interleaved-test-and-then-train, also termed as *accumulated one-step ahead error/accuracy*, is based on the idea to measure model performance in one-step ahead cycles, i.e., based on one newly loaded sample only. In particular, the following steps are carried out:

- (1) Load a new sample (the N th).
- (2) Predict its target \hat{y} using the current evolved fuzzy systems.
- (3) Compare the prediction \hat{y} with the true target value y and update the performance measure pm :

$$pm(y, \hat{y})(N) \leftarrow upd(pm(y, \hat{y})(N - 1)). \quad (45)$$

(4) Update the evolved fuzzy system (arbitrary approach).

(5) Erase sample and go to Step (1).

This is a rather optimistic approach, assuming that the target response is immediately available for each new sample after its prediction. Often, it may be delayed (Marrs *et al.*, 2012; Subramanian *et al.*, 2013), postponing the update of the model performance to a later stage. Furthermore, in case of single sample updates its prediction horizon is minimal that makes it difficult to really provide a clear distinction between training and test data, hence weakening their independence. In this sense, this variant is sometimes too optimistic, under-estimating the true model error. On the other hand, all training samples are also used for testing, thus it is quite practicable for small streams.

The periodic holdout procedure can “look ahead” to collect a batch of examples from the stream for use as test examples. In particular, it uses each odd data block for learning and updating the model and each even data block for eliciting the model performance on this latest block; thereby, the data block sizes may be different for training and testing and may vary in dependence of the actual stream characteristics. In this sense, a lower number of samples is used for model updating/tuning than in case of *interleaved-test-and-then-train* procedure. In experimental test designs, where the streams are finite, it is thus more practicable for longer streams. On the other hand, this method would be preferable in scenarios with concept drift, as it would measure a model’s ability to adapt to the *latest trends* in the data — whereas in *interleaved-test-and-then-train* procedure all the data seen so far is reflected in the current accuracy/error, becoming less flexible over time. Forgetting may be integrated into Equation (45), but this requires an additional tuning parameter (e.g., a window size in case of sliding windows). The following steps are carried out in a periodic holdout process:

(1) Load a new data block $X_N = \vec{x}_{N*m+1}, \dots, \vec{x}_{N*m+m}$ containing m samples.

(2) If N is odd:

(a) Predict the target values $\hat{y}_1, \dots, \hat{y}_m$ using the current evolved fuzzy systems.

(b) Compare the predictions $\hat{y}_{N*m+1}, \dots, \hat{y}_{N*m+m}$ with the true target values $y_{N*m+m}, \dots, y_{N*m+m}$ and calculate the performance measure (one or more of Equation (46) to Equation (51)) using \bar{y} and \hat{y} .

(c) Erase block and go to Step (1).

(3) Else (N is even):

(a) Update the evolved fuzzy system (arbitrary approach) with all samples in the buffer using the real target values.

(b) Erase block and go to Step (1).

Last but not least, an alternative to the online evaluation procedure is to evolve the

model an a certain training set and then evaluate it on an independent test set, termed as *fully-train-and-then-test*. This procedure is most heavily used in many applications of EFS, especially in nonlinear system identification and forecasting problems, as summarized in [Table 3.3](#). It extends the prediction horizon of *interleaved-test-and-then-train* procedure with the size of the test set, but only does this in one occasion (at the end of learning). Therefore, it is not useable in drift cases or severe changes during online incremental learning processes and should be only used during development and experimental phases.

Regarding appropriate performance measures, the most convenient choice in classification problems is the number of correctly classified samples (accuracy). In the time instance N (processing the N th sample), the update of the performance measure as in [Equation \(45\)](#) is then conducted by

$$Acc(N) = \frac{Acc(N-1) * (N-1) + I(\hat{y}, y)}{N}, \quad (46)$$

with $Acc(0) = 0$ and I the indicator function, i.e., $I(a, b) = 1$ whenever $a = b$, otherwise $I(a, b) = 0$, \hat{y} the predicted class label and y the real one. It can be used in the same manner for eliciting the accuracy on whole data blocks as in the periodic hold out case. Another important measure is the so-called *confusion matrix* (Stehman, 1997), which is defined as:

$$C = \begin{bmatrix} N_{11} & N_{12} & \dots & N_{1K} \\ N_{21} & N_{22} & \dots & N_{2K} \\ \vdots & \vdots & \vdots & \vdots \\ N_{K1} & N_{K2} & \dots & N_{KK} \end{bmatrix}, \quad (47)$$

with K the current number of classes, where the diagonal elements denote the number of class j samples which have been correctly classified as class j samples and the element N_{ij} denotes the number of class i samples which are wrongly classified to class j . These can be simply updated by counting.

Furthermore, often someone may be not only interested how strong the samples are miss-classified, but also how certain they are classified (either correctly or wrongly). For instance, a high classification accuracy with a lot of certain classifier outputs may have a higher value than with a lot of uncertain outputs. Furthermore, a high uncertainty degree in the statements point to a lot of conflict cases (compare with [Equation \(19\)](#) and [Figure 3.5](#)), i.e., a lot of samples falling into class overlap regions. Therefore, a measure telling the uncertainty degree over samples seen so far is of great interest — a widely used measure is provided in (Amor *et al.*, 2004):

$$Rel_N = 1 - \frac{1}{K} \sum_{k=1}^K |conf_k(N) - y_k(N)|, \quad (48)$$

where $y_k(i) = 1$ if k is the class the current sample N belongs to, and $y_k(i) = 0$ otherwise, and $\text{con } f_k$ the certainty level in class k , which can be calculated by Equation (19), for instance. It can be accumulated in the same manner as the accuracy above.

In case of regression problems, the most common choices are the root mean squared error (RMSE), the mean absolute error (MAE) or the average percentual deviation (APE). Their updates are achieved by:

$$\text{RMSE}(N) = \sqrt{\frac{1}{N} * ((N - 1) * (\text{RMSE}(N - 1)^2) + (y - \hat{y})^2)}, \quad (49)$$

$$\text{MAE}(N) = \frac{1}{N} * ((N - 1) * \text{MAE}(N - 1) + |y - \hat{y}|), \quad (50)$$

$$\text{APE}(N) = \frac{1}{N} * \left((N - 1) * \text{APE}(N - 1) + \frac{|y - \hat{y}|}{\text{range}(y)} \right). \quad (51)$$

Their batch calculation for even blocks in periodic hold out test is following the standard procedure and thus can be realized from Wikipedia. Instead of calculating the concrete error values, often the observed versus predicted curves over time and their correlation plots are shown. This gives an even better impression under which circumstances and at which points of time the model behaves in which way. A systematic error shift can be also realized.

Apart from accuracy criteria, other measures rely on the complexity of the models. In most EFS application cases (refer to [Table 3.3](#)), the development of the number of rules over time is plotted as a two-dimensional function. Sometimes, also the number of fuzzy sets are reported as additionally criteria which depend on the rule lengths. These mostly depend on the dimensionality of the learning problem. In case of embedded feature selection, there may be a big drop in the number of fuzzy sets once some features are discarded resp. out-weighted (compare with [Section 3.4.2](#)). [Figure 3.11](#) shows a typical accumulated accuracy curve over time in the left image (using an all-pairs EFC including active learning option with different amount of samples for updating) and a typical development of the number of rules in the right one: At the start, rules are evolved and accumulated, later some rules turned out to be superfluous and hence are back-pruned and merged. This guarantees an anytime flexibility.

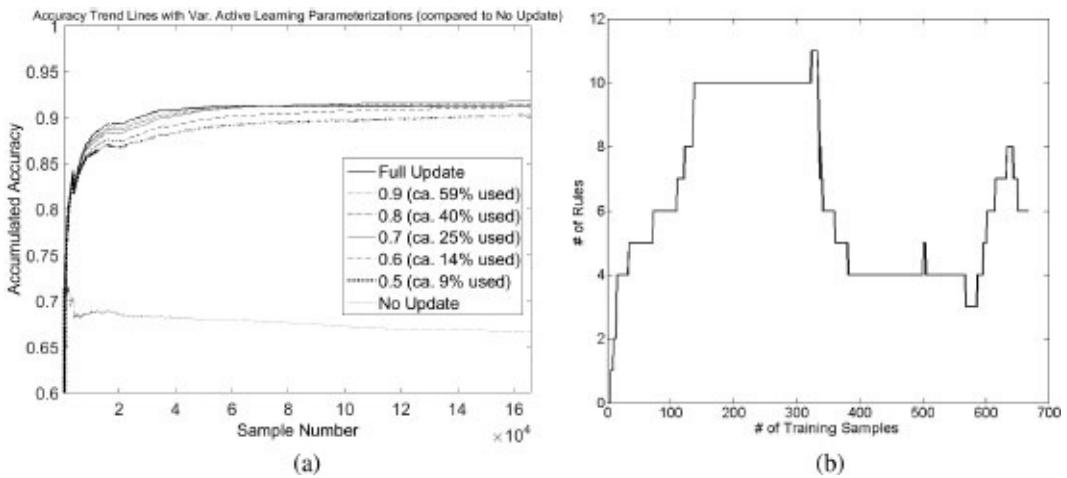


Figure 3.11: (a) Typical accumulated accuracy increasing over time in case of full update and active learning variants (reducing the number of samples used for updating while still maintaining high accuracy); (b) Typical evolution of the number of rules over time.

3.6.5. Real-World Applications of EFS — Overview

Due to space restrictions, a complete description of application scenarios in which EFSs have been successfully implemented and used so far is simply impossible. Thus, we restrict ourselves to a compact summary within [Table 3.3](#) showing application types and classes and indicating which EFS approaches have been used in the circumstance of which application type. In all of these cases, EFS(C) helped to increase automatization capability, improving performance of the models and finally increasing the useability of the whole systems; in some cases, no modeling has been (could be) applied before at all.

Table 3.3: Application classes in which EFS approaches have been successfully applied so far.

Application Type/Class (alphabetically)	EFS approaches (+ refs)	Comment
Active learning / human– machine interaction	FLEXFIS-Class (Lughofer <i>et al.</i> , 2009; Lughofer, 2012d), EFC-AP (Lughofer, 2012a), FLEXFIS-PLS (Cernuda <i>et al.</i> , 2014)	Reducing the annotation effort and measurement costs in industrial processes
Adaptive online control	evolving PID and MRC controllers in (Angelov and Skrjanc, 2013), eFuMo (Zdsar <i>et al.</i> , 2014), rGK (Dovzan <i>et al.</i> , 2012), self-evolving NFC (Cara <i>et al.</i> , 2013), adaptive controller in (Rong <i>et al.</i> , 2014).	Design of fuzzy controllers which can be updated and evolved on-the-fly
Bioinformatics	EFuNN (Kasabov, 2002)	Specific applications such as ribosome binding site (RBS) identification, gene profiling
Chemometric Modeling and Process Control	FLEXFIS++ (Cernuda <i>et al.</i> , 2013, 2012); the approach in Bodyanskiy and Vynokurova (2013)	The application of EFS onto processes in chemical industry (high-dim. NIR spectra)

EEG signals classification and processing	eTS (Xydeas <i>et al.</i> , 2006), epSNNr (Nuntalid <i>et al.</i> , 2011)	Time-series modeling with the inclusion of time delays
Evolving Smart Sensors (eSensors)	eTS+ (Macias-Hernandez and Angelov, 2010) (gas industry), (Angelov and Kordon, 2010a, 2010b) (chemical process industry), FLEXFIS (Lughofer <i>et al.</i> , 2011c) and PANFIS (Pratama <i>et al.</i> , 2014a) (NOx emissions)	Evolving predictive and forecasting models in order to substitute cost-intensive hardware sensors
Forecasting and prediction (general)	AHLTNM (Kalhor <i>et al.</i> , 2010) (daily temp.), eT2FIS (Tung <i>et al.</i> , 2013) (traffic flow), eFPT (Shaker <i>et al.</i> , 2013) (Statlog from UCI), eFT (Lemos <i>et al.</i> , 2011b) and eMG (Lemos <i>et al.</i> , 2011a) (short-term electricity load), FLEXFIS+ (Lughofer <i>et al.</i> , 2011b) and GENEFIS (Pratama <i>et al.</i> , 2014b) (house prices), LOLIMOT inc. (Hametner and Jakubek, 2013) (maximum cylinder pressure), rGK (Dovzan <i>et al.</i> , 2012) (sales prediction) and others	Various successful implementations of EFS
Financial domains	eT2FIS (Tung <i>et al.</i> , 2013), evolving granular systems (Leite <i>et al.</i> , 2012b), ePL (Maciel <i>et al.</i> , 2012), PANFIS (Pratama <i>et al.</i> , 2014a), SOFNN (Prasad <i>et al.</i> , 2010)	Time-series modeling with the inclusion of time delays
Identification of dynamic benchmark problems	DENFIS (Kasabov and Song, 2002), eT2FIS (Tung <i>et al.</i> , 2013), eTS+ (Angelov, 2010), FLEXFIS (Lughofer, 2008), SAFIS (Rong, 2012), SEIT2FNN (Juang and Tsao, 2008), SOFNN (Prasad <i>et al.</i> , 2010)	Mackey-Glass, Box-Jenkins, etc.
Online fault detection and condition monitoring	eMG for classification (Lemos <i>et al.</i> , 2013), FLEXFIS++ (Lughofer and Guardiola, 2008b; Serdio <i>et al.</i> , 2014a), rGK (Dovzan <i>et al.</i> , 2012)	EFS applied as SysID models for extracting residuals
Online monitoring	eTS+ (Macias-Hernandez and Angelov, 2010) (gas industry)	Supervision of system behaviors
Robotics	eTS+ (Zhou and Angelov, 2007)	In the area of self-localization
Time-series modeling	DENFIS (Widiputra <i>et al.</i> , 2012), ENFM (Soleimani <i>et al.</i> , 2010)	Local modeling of multiple time-series versus instance-based

	and eTS-LS-SVM (Komijani <i>et al.</i> , 2012) (sun spot)	learning
User behavior identification	eClass and eTS (Angelov <i>et al.</i> , 2012; Iglesias <i>et al.</i> , 2010), eTS+ (Andreu and Angelov, 2013), FPA (Wang <i>et al.</i> , 2013)	Analysis of the user's behaviors in multi-agent systems, on computers, indoor environments etc.
Video processing	eTS, eTS+ (Angelov <i>et al.</i> , 2011; Zhou and Angelov, 2006)	Including real-time object id., obstacles tracking and novelty detection
Visual quality control	EFC-AP (Lughofer and Buchtala, 2013), FLEXFIS-Class (Eitzinger <i>et al.</i> , 2010; Lughofer, 2010b), pClass (Pratama <i>et al.</i> , 2014c)	Image classification tasks based on feature vectors

Acknowledgments

This work was funded by the research programme at the LCM GmbH as part of a K2 project. K2 projects are financed using funding from the Austrian COMET-K2 programme. The COMET K2 projects at LCM are supported by the Austrian federal government, the federal state of Upper Austria, the Johannes Kepler University and all of the scientific partners which form part of the K2-COMET Consortium. This publication reflects only the author's views.

References

- Abonyi, J., Babuska, R. and Szeifert, F. (2002). Modified Gath–Geva fuzzy clustering for identification of Takagi–Sugeno fuzzy models. *IEEE Trans. Syst., Man Cybern. Part B*, 32(5), pp. 612–621.
- Abonyi, J. (2003). *Fuzzy Model Identification for Control*. Boston, U.S.A.: Birkhäuser.
- Abraham, W. and Robins, A. (2005). Memory retention: The synaptic stability versus plasticity dilemma. *Trends Neurosci.*, 28(2), pp. 73–78.
- Affenzeller, M., Winkler, S., Wagner, S. and Beham, A. (2009). *Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications*. Boca Raton, Florida: Chapman & Hall.
- Allwein, E., Schapire, R. and Singer, Y. (2001). Reducing multiclass to binary: a unifying approach for margin classifiers. *J. Mach. Learn. Res.*, 1, pp. 113–141.
- Almaksour, A. and Anquetil, E. (2011). Improving premise structure in evolving Takagi–Sugeno neuro-fuzzy classifiers. *Evolving Syst.*, 2, pp. 25–33.
- Amor, N., Benferhat, S. and Elouedi, Z. (2004). Qualitative classification and evaluation in possibilistic decision trees. In *Proc. FUZZ-IEEE Conf.*, Budapest, Hungary, pp. 653–657.
- Andreu, J. and Angelov, P. (2013). Towards generic human activity recognition for ubiquitous applications. *J. Ambient Intell. Human Comput.*, 4, pp. 155–156.
- Angelov, P. (2010). Evolving Takagi–Sugeno fuzzy systems from streaming data, eTS+. In Angelov, P., Filev, D. and Kasabov, N. (eds.), *Evolving Intelligent Systems: Methodology and Applications*. New York: John Wiley & Sons, pp. 21–50.
- Angelov, P. and Filev, D. (2004). An approach to online identification of Takagi–Sugeno fuzzy models. *IEEE Trans. Syst. Man Cybern., Part B: Cybern.*, 34(1), pp. 484–498.
- Angelov, P. and Kasabov, N. (2005). Evolving computational intelligence systems. In *Proc. 1st Int. Workshop on Genet. Fuzzy Syst.*, Granada, Spain, pp. 76–82.
- Angelov, P. and Kordon, A. (2010a). Evolving inferential sensors in the chemical process industry. In Angelov, P., Filev, D. and Kasabov, N. (eds.), *Evolving Intelligent Systems: Methodology and Applications*. New York: John Wiley & Sons, pp. 313–336.
- Angelov, P. and Kordon, A. (2010b). Adaptive inferential sensors based on evolving fuzzy models: An industrial case study. *IEEE Trans. Syst., Man Cybern., Part B: Cybern.*, 40(2), pp. 529–539.
- Angelov, P. and Skrjanc, I. (2013). Robust evolving cloud-based controller for a hydraulic plant. In *Proc. 2013 IEEE Conf. Evolving Adapt. Intell. Syst. (EAIS)*. Singapore, pp. 1–8.
- Angelov, P., Filev, D. and Kasabov, N. (2010). *Evolving Intelligent Systems—Methodology and Applications*. New York: John Wiley & Sons.
- Angelov, P., Ledezma, A. and Sanchis, A. (2012). Creating evolving user behavior profiles automatically. *IEEE Trans. Knowl. Data Eng.*, 24(5), pp. 854–867.
- Angelov, P., Lugofer, E. and Zhou, X. (2008). Evolving fuzzy classifiers using different model architectures. *Fuzzy Sets Syst.*, 159(23), pp. 3160–3182.
- Angelov, P., Sadeghi-Tehran, P. and Ramezani, R. (2011). An approach to automatic real-time novelty detection, object identification, and tracking in video streams based on recursive density estimation and evolving Takagi–Sugeno fuzzy systems. *Int. J. Intell. Syst.*, 26(3), pp. 189–205.
- Aström, K. and Wittenmark, B. (1994). *Adaptive Control Second Edition*. Boston, MA, USA: Addison-Wesley Longman Publishing Co. Inc.
- Babuska, R. (1998). *Fuzzy Modeling for Control*. Norwell, Massachusetts: Kluwer Academic Publishers.
- Balas, V., Fodor, J. and Varkonyi-Koczy, A. (2009). *Soft Computing based Modeling in Intelligent Systems*. Berlin, Heidelberg: Springer.
- Bifet, A., Holmes, G., Kirkby, R. and Pfahringer, B. (2010). MOA: Massive online analysis. *J. Mach. Learn. Res.*, 11, pp. 1601–1604.

- Bifet, A. and Kirkby, R. (2011). Data stream mining — a practical approach. Technical report, University of Waikato, Japan, Department of Computer Sciences.
- Bikdash, M. (1999). A highly interpretable form of sugeno inference systems. *IEEE Trans. Fuzzy Syst.*, 7(6), pp. 686–696.
- Bishop, C. (2007). *Pattern Recognition and Machine Learning*. New York: Springer.
- Bodyanskiy, Y. and Vynokurova, O. (2013). Hybrid adaptive wavelet-neuro-fuzzy system for chaotic time series identification, *Inf. Sci.*, 220, pp. 170–179.
- Bouchachia, A. (2009). Incremental induction of classification fuzzy rules. In *IEEE Workshop Evolving Self-Dev. Intell. Syst. (ESDIS) 2009*. Nashville, U.S.A., pp. 32–39.
- Bouchachia, A. (2011). Evolving clustering: An asset for evolving systems. *IEEE SMC NewsL.*, 36.
- Bouchachia, A. and Mittermeir, R. (2006). Towards incremental fuzzy classifiers. *Soft Comput.*, 11(2), pp. 193–207.
- Bouchachia, A., Lughofer, E. and Mouchaweh, M. (2014). Editorial to the special issue: Evolving soft computing techniques and applications. *Appl. Soft Comput.*, 14, pp. 141–143.
- Bouchachia, A., Lughofer, E. and Sanchez, D. (2013). Editorial to the special issue: Online fuzzy machine learning and data mining. *Inf. Sci.*, 220, pp. 1–4.
- Bouillon, M., Anquetil, E. and Almaksour, A. (2013). Decremental learning of evolving fuzzy inference systems: Application to handwritten gesture recognition. In Perner, P. (ed.), *Machine Learning and Data Mining in Pattern Recognition*, 7988, *Lecture Notes in Computer Science*. New York: Springer, pp. 115–129.
- Breiman, L., Friedman, J., Stone, C. and Olshen, R. (1993). *Classification and Regression Trees*. Boca Raton: Chapman and Hall.
- Cara, A., Herrera, L., Pomares, H. and Rojas, I. (2013). New online self-evolving neuro fuzzy controller based on the tase-nf model. *Inf. Sci.*, 220, pp. 226–243.
- Carr, V. and Tah, J. (2001). A fuzzy approach to construction project risk assessment and analysis: construction project risk management system. *Adv. Eng. Softw.*, 32(10–11), pp. 847–857.
- Casillas, J., Cordon, O., Herrera, F. and Magdalena, L. (2003). *Interpretability Issues in Fuzzy Modeling*. Berlin, Heidelberg: Springer Verlag.
- Castro, J. and Delgado, M. (1996). Fuzzy systems with defuzzification are universal approximators. *IEEE Trans. Syst. Man Cybern. Part B: Cybern.*, 26(1), pp. 149–152.
- Cernuda, C., Lughofer, E., Hintenau, P., Märzinger, W., Reischer, T., Pawlicek, M. and Kasberger, J. (2013). Hybrid adaptive calibration methods and ensemble strategy for prediction of cloud point in melamine resin production. *Chemometr. Intell. Lab. Syst.*, 126, pp. 60–75.
- Cernuda, C., Lughofer, E., Mayr, G., Röder, T., Hintenau, P., Märzinger, W. and Kasberger, J. (2014). Incremental and decremental active learning for optimized self-adaptive calibration in viscose production. *Chemometr. Intell. Lab. Syst.*, 138, pp. 14–29.
- Cernuda, C., Lughofer, E., Suppan, L., Röder, T., Schmuck, R., Hintenau, P., Märzinger, W. and Kasberger, J. (2012). Evolving chemometric models for predicting dynamic process parameters in viscose production. *Anal. Chim. Acta*, 725, pp. 22–38.
- Chapelle, O., Schoelkopf, B. and Zien, A. (2006). *Semi-Supervised Learning*. Cambridge, MA: MIT Press.
- Cho, J. and Park, D. (2000). Novel fuzzy logic control based on weighting of partially inconsistent rules using neural network. *J. Intell. Fuzzy Syst.*, 8(2), pp. 99–110.
- Chong, C.-Y. and Kumar, S. (2003). Sensor networks: Evolution, opportunities, and challenges. *Proc. IEEE*, 91(8), pp. 1247–1256.
- Chu, W., Zinkevich, M., Li, L., Thomas, A. and Zheng, B. (2011). Unbiased online active learning in data streams. In *Proc. KDD 2011*. San Diego, California.
- Cleveland, W. and Devlin, S. (1988). Locally weighted regression: An approach to regression analysis by local fitting. *J. Am. Stat. Assoc.*, 84(403), pp. 596–610.
- Cohn, D., Atlas, L. and Ladner, R. (1994). Improving generalization with active learning. *Mach. Learn.*, 15(2), pp. 201–

- Day, N. E. (1969). Estimating the components of a mixture of normal distributions. *Biometrika*, 56(463–474).
- Diehl, C. and Cauwenberghs, G. (2003). SVM incremental learning, adaptation and optimization. In *Proc. Int. Joint Conf. Neural Netw.*, Boston, 4, pp. 2685–2690.
- Douglas, S. (1996). Efficient approximate implementations of the fast affine projection algorithm using orthogonal transforms. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Atlanta, Georgia, pp. 1656–1659.
- Dovzan, D. and Skrjanc, I. (2011). Recursive clustering based on a Gustafson–Kessel algorithm. *Evolving Syst.*, 2(1), pp. 15–24.
- Dovzan, D., Logar, V. and Skrjanc, I. (2012). Solving the sales prediction problem with fuzzy evolving methods. In *WCCI 2012 IEEE World Congr. Comput. Intell.*, Brisbane, Australia.
- Duda, R., Hart, P. and Stork, D. (2000). *Pattern Classification, Second Edition*. Southern Gate, Chichester, West Sussex, England: Wiley-Interscience.
- Dy, J. and Brodley, C. (2004). Feature selection for unsupervised learning. *J. Mach. Learn. Res.*, 5, pp. 845–889.
- Efron, B. and Tibshirani, R. (1993). *An Introduction to the Bootstrap*. Boca Raton, Florida: Chapman & Hall/CRC.
- Eitzinger, C., Heidl, W., Lugofer, E., Raiser, S., Smith, J., Tahir, M., Sannen, D. and van Brussel, H. (2010). Assessment of the influence of adaptive components in trainable surface inspection systems. *Mach. Vis. Appl.*, 21(5), pp. 613–626.
- Fürnkranz, J. (2001). Round robin rule learning. In *Proc. Int. Conf. Mach. Learn. (ICML 2011)*, Williamstown, MA, pp. 146–153.
- Fürnkranz, J. (2002). Round robin classification. *J. Mach. Learn. Res.*, 2, pp. 721–747.
- French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends Cogn. Sci.*, 3(4), pp. 128–135.
- Fuller, R. (1999). *Introduction to Neuro-Fuzzy Systems*. Heidelberg, Germany: Physica-Verlag.
- Gacto, M., Alcalá, R. and Herrera, F. (2011). Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures. *Inf. Sci.*, 181(20), pp. 4340–4360.
- Gama, J. (2010). *Knowledge Discovery from Data Streams*. Boca Raton, Florida: Chapman & Hall/CRC.
- Gan, G., Ma, C. and Wu, J. (2007). *Data Clustering: Theory, Algorithms, and Applications (Asa-Siam Series on Statistics and Applied Probability)*. U.S.A.: Society for Industrial & Applied Mathematics.
- Gay, S. L. (1996). Dynamically regularized fast recursive least squares with application to echo cancellation. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Atlanta, Georgia, pp. 957–960.
- Hametner, C. and Jakubek, S. (2013). Local model network identification for online engine modelling. *Inf. Sci.*, 220, pp. 210–225.
- Hamker, F. (2001). RBF learning in a non-stationary environment: the stability-plasticity dilemma. In Howlett, R. and Jain, L. (eds.), *Radial Basis Function Networks 1: Recent Developments in Theory and Applications*, Heidelberg, New York: Physica Verlag, pp. 219–251.
- Hastie, T., Tibshirani, R. and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference and Prediction, Second Edition*. New York Berlin Heidelberg: Springer.
- Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation, 2nd Edition*. Upper Saddle River, New Jersey: Prentice Hall Inc.
- He, H. and Garcia, E. (2009). Learning from imbalanced data. *IEEE Trans. Knowl. Data Eng.*, 21(9), pp. 1263–1284.
- Hentzgen, S., Strickert, M. and Hüllermeier, E. (2014). Visualization of evolving fuzzy rule-based systems. *Evolving Syst.*, DOI: 10.1007/s12530-014-9110-4, on-line and in press.
- Henzgen, S., Strickert, M. and Hüllermeier, E. (2013). Rule chains for visualizing evolving fuzzy rule-based systems. In *Advances in Intelligent Systems and Computing*, 226, *Proc. Eighth Int. Conf. Comput. Recognit. Syst. CORES 2013*. Cambridge, MA: Springer, pp. 279–288.
- Herrera, L., Pomares, H., Rojas, I., Valenzuela, O. and Prieto, A. (2005). TaSe, a taylor series-based fuzzy system model that combines interpretability and accuracy. *Fuzzy Sets Syst.*, 153(3), pp. 403–427.

- Hisada, M., Ozawa, S., Zhang, K. and Kasabov, N. (2010). Incremental linear discriminant analysis for evolving feature spaces in multitask pattern recognition problems. *Evolving Syst.*, 1(1), pp. 17–27.
- Ho, W., Tung, W. and Quek, C. (2010). An evolving Mamdani–Takagi–Sugeno based neural-fuzzy inference system with improved interpretability–accuracy. In *Proc. WCCI 2010 IEEE World Congr. Comput. Intell.*, Barcelona, pp. 682–689.
- Holmlund, L. and Ostergaard, J. (1982). Control of a cement kiln by fuzzy logic. *Fuzzy Inf. Decis. Process.*, pp. 398–409.
- Huang, Z., Gedeon, T. D. and Nikravesh, M. (2008). Pattern trees induction: A new machine learning method. *IEEE Trans. Fuzzy Syst.*, 16(4), pp. 958–970.
- Hüllermeier, E. and Brinker, K. (2008). Learning valued preference structures for solving classification problems. *Fuzzy Sets Syst.*, 159(18), pp. 2337–2352.
- Hühn, J. and Hüllermeier, E. (2009). FR3: A fuzzy rule learner for inducing reliable classifiers, *IEEE Trans. Fuzzy Syst.*, 17(1), pp. 138–149.
- Iglesias, J., Angelov, P., Ledezma, A. and Sanchis, A. (2010). Evolving classification of agent's behaviors: a general approach. *Evolving Syst.*, 1(3), pp. 161–172.
- Ishibuchi, H. and Nakashima, T. (2001). Effect of rule weights in fuzzy rule-based classification systems. *IEEE Trans. Fuzzy Syst.*, 9(4), pp. 506–515.
- Jang, J.-S. (1993). ANFIS: Adaptive-network-based fuzzy inference systems. *IEEE Trans. Syst. Man Cybern.*, 23(3), pp. 665–685.
- Juang, C. and Tsao, Y. (2008). A self-evolving interval type-2 fuzzy neural network with online structure and parameter learning. *IEEE Trans. Fuzzy Syst.*, 16(6), pp. 1411–1424.
- Kaczmarz, S. (1993). Approximate solution of systems of linear equations. *Int. J. Control.*, 53, pp. 1269–1271.
- Kalhor, A., Araabi, B. and Lucas, C. (2010). An online predictor model as adaptive habitually linear and transiently nonlinear model. *Evolving Syst.*, 1(1), pp. 29–41.
- Karer, G. and Skrjanc, I. (2013). *Predictive Approaches to Control of Complex Systems*. Berlin, Heidelberg: Springer Verlag.
- Karnik, N. and Mendel, J. (2001). Centroid of a type-2 fuzzy set. *Inf. Sci.*, 132(1–4), pp. 195–220.
- Kasabov, N. (2002). *Evolving Connectionist Systems — Methods and Applications in Bioinformatics, Brain Study and Intelligent Machines*. London: Springer Verlag.
- Kasabov, N. K. and Song, Q. (2002). DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction. *IEEE Trans. Fuzzy Syst.*, 10(2), pp. 144–154.
- Klement, E., Mesiar, R. and Pap, E. (2000). *Triangular Norms*. New York: Kluwer Academic Publishers.
- Klinkenberg, R. (2004). Learning drifting concepts: example selection vs. example weighting, *Intell. Data Anal.*, 8(3), pp. 281–300.
- Koenig, S., Likhachev, M., Liu, Y. and Furcy, D. (2004). Incremental heuristic search in artificial intelligence. *Artif. Intell. Mag.*, 25(2), pp. 99–112.
- Komijani, M., Lucas, C., Araabi, B. and Kalhor, A. (2012). Introducing evolving Takagi–Sugeno method based on local least squares support vector machine models. *Evolving Syst.* 3(2), pp. 81–93.
- Kruse, R., Gebhardt, J. and Palm, R. (1994). *Fuzzy Systems in Computer Science*. Wiesbaden: Verlag Vieweg.
- Kuncheva, L. (2000). *Fuzzy Classifier Design*. Heidelberg: Physica-Verlag.
- Leekwijck, W. and Kerre, E. (1999). Defuzzification: criteria and classification. *Fuzzy Sets Syst.* 108(2), pp. 159–178.
- Leite, D., Ballini, R., Costa, P. and Gomide, F. (2012a). Evolving fuzzy granular modeling from nonstationary fuzzy data streams. *Evolving Syst.* 3(2), pp. 65–79.
- Leite, D., Costa, P. and Gomide, F. (2012b). Interval approach for evolving granular system modeling. In Sayed-Mouchaweh, M. and Lughofner, E. (eds.), *Learning in Non-Stationary Environments: Methods and Applications*. New York: Springer, pp. 271–300.

- Lemos, A., Caminhas, W. and Gomide, F. (2011a). Multivariable gaussian evolving fuzzy modeling system. *IEEE Trans. Fuzzy Syst.*, 19(1), pp. 91–104.
- Lemos, A., Caminhas, W. and Gomide, F. (2011b). Fuzzy evolving linear regression trees. *Evolving Syst.*, 2(1), pp. 1–14.
- Lemos, A., Caminhas, W. and Gomide, F. (2013). Adaptive fault detection and diagnosis using an evolving fuzzy classifier. *Inf. Sci.*, 220, pp. 64–85.
- Leng, G., McGinnity, T. and Prasad, G. (2005). An approach for on-line extraction of fuzzy rules using a self-organising fuzzy neural network. *Fuzzy Sets Syst.*, 150(2), pp. 211–243.
- Leng, G., Zeng, X.-J. and Keane, J. (2012). An improved approach of self-organising fuzzy neural network based on similarity measures. *Evolving Syst.*, 3(1), pp. 19–30.
- Leondes, C. (1998). *Fuzzy Logic and Expert Systems Applications (Neural Network Systems Techniques and Applications)*. San Diego, California: Academic Press.
- Li, Y. (2004). On incremental and robust subspace learning. *Pattern Recognit.*, 37(7), pp. 1509–1518.
- Liang, Q. and Mendel, J. (2000). Interval type-2 fuzzy logic systems: Theory and design. *IEEE Trans. Fuzzy Syst.*, 8(5), pp. 535–550.
- Lima, E., Hell, M., Ballini, R. and Gomide, F. (2010). Evolving fuzzy modeling using participatory learning. In Angelov, P., Filev, D. and Kasabov, N. (eds.), *Evolving Intelligent Systems: Methodology and Applications*. New York: John Wiley & Sons, pp. 67–86.
- Lippmann, R. (1991). A critical overview of neural network pattern classifiers. In *Proc. IEEE Workshop Neural Netw. Signal Process.*, pp. 266–275.
- Ljung, L. (1999). *System Identification: Theory for the User*. Upper Saddle River, New Jersey: Prentice Hall PTR, Prentic Hall Inc.
- Lughofer, E., Smith, J. E., Caleb-Solly, P., Tahir, M., Eitzinger, C., Sannen, D. and Nuttin, M. (2009). Human-machine interaction issues in quality control based on on-line image classification. *IEEE Trans. Syst. Man Cybern., Part A: Syst. Humans*, 39(5), pp. 960–971.
- Lughofer, E. (2008). FLEXFIS: A robust incremental learning approach for evolving TS fuzzy models. *IEEE Trans. Fuzzy Syst.*, 16(6), pp. 1393–1410.
- Lughofer, E. (2010a). Towards robust evolving fuzzy systems. In Angelov, P., Filev, D. and Kasabov, N. (eds.), *Evolving Intelligent Systems: Methodology and Applications*. New York: John Wiley & Sons, pp. 87–126.
- Lughofer, E. (2010b). On-line evolving image classifiers and their application to surface inspection. *Image Vis. Comput.*, 28(7), 1065–1079.
- Lughofer, E. (2011a). Human-inspired evolving machines — the next generation of evolving intelligent systems? *SMC NewsL.*, 36.
- Lughofer, E. (2011b). *Evolving Fuzzy Systems — Methodologies, Advanced Concepts and Applications*. Berlin, Heidelberg: Springer.
- Lughofer, E. (2011c). On-line incremental feature weighting in evolving fuzzy classifiers. *Fuzzy Sets Syst.*, 163(1), pp. 1–23.
- Lughofer, E. (2012a). Single-pass active learning with conflict and ignorance. *Evolving Syst.*, 3(4), pp. 251–271.
- Lughofer, E. (2012b). Flexible evolving fuzzy inference systems from data streams (FLEXFIS++). In Sayed-Mouchaweh, M. and Lughofer, E. (eds.), *Learning in Non-Stationary Environments: Methods and Applications*. New York: Springer, pp. 205–246.
- Lughofer, E. and Sayed-Mouchaweh, M. (2015). Autonomous data stream clustering implementing incremental split-and-merge techniques — Towards a plug-and-play approach. *Inf. Sci.*, 204, pp. 54–79.
- Lughofer, E. (2012d). Hybrid active learning (HAL) for reducing the annotation efforts of operators in classification systems. *Pattern Recognit.*, 45(2), pp. 884–896.
- Lughofer, E. (2013). On-line assurance of interpretability criteria in evolving fuzzy systems — achievements, new concepts and open issues. *Inf. Sci.*, 251, pp. 22–46.
- Lughofer, E. and Angelov, P. (2011). Handling drifts and shifts in on-line data streams with evolving fuzzy systems.

- Appl. Soft Comput.*, 11(2), pp. 2057–2068.
- Lughofer, E. and Buchtala, O. (2013). Reliable all-pairs evolving fuzzy classifiers. *IEEE Trans. Fuzzy Syst.*, 21(4), pp. 625–641.
- Lughofer, E. and Guardiola, C. (2008a). Applying evolving fuzzy models with adaptive local error bars to on-line fault detection. In *Proc. Genet. Evolving Fuzzy Syst., 2008*. Witten-Bommerholz, Germany, pp. 35–40.
- Lughofer, E. and Guardiola, C. (2008b). On-line fault detection with data-driven evolving fuzzy models. *J. Control Intell. Syst.*, 36(4), pp. 307–317.
- Lughofer, E. and Hüllermeier, E. (2011). On-line redundancy elimination in evolving fuzzy regression models using a fuzzy inclusion measure. In *Proc. EUSFLAT 2011 Conf.*, Aix-Les-Bains, France: Elsevier, pp. 380–387.
- Lughofer, E., Bouchot, J.-L. and Shaker, A. (2011a). On-line elimination of local redundancies in evolving fuzzy systems. *Evolving Syst.*, 2(3), pp. 165–187.
- Lughofer, E., Trawinski, B., Trawinski, K., Kempa, O. and Lasota, T. (2011b). On employing fuzzy modeling algorithms for the valuation of residential premises. *Inf. Sci.*, 181(23), pp. 5123–5142.
- Lughofer, E., Macian, V., Guardiola, C. and Klement, E. (2011c). Identifying static and dynamic prediction models for NO_x emissions with evolving fuzzy systems. *Appl. Soft Comput.*, 11(2), pp. 2487–2500.
- Lughofer, E., Cernuda, C. and Pratama, M. (2013). Generalized flexible fuzzy inference systems. In *Proc. ICMLA 2013 Conf.*, Miami, Florida, pp. 1–7.
- Lughofer, E., Cernuda, C., Kindermann, S. and Pratama, M. (2014). Generalized smart evolving fuzzy systems. *Evolving Syst.*, online and in press, doi: 10.1007/s12530-015-9132-6
- Macias-Hernandez, J. and Angelov, P. (2010). Applications of evolving intelligent systems to the oil and gas industry. In Angelov, P., Filev, D. and Kasabov, N. (eds.), *Evolving Intelligent Systems: Methodology and Applications*. New York: John Wiley & Sons, pp. 401–421.
- Maciel, L., Lemos, A., Gomide, F. and Ballini, R. (2012). Evolving fuzzy systems for pricing fixed income options. *Evolving Syst.*, 3(1), pp. 5–18.
- Mamdani, E. (1977). Application of fuzzy logic to approximate reasoning using linguistic systems. *Fuzzy Sets Syst.* 26(12), pp. 1182–1191.
- Marrs, G., Black, M. and Hickey, R. (2012). The use of time stamps in handling latency and concept drift in online learning. *Evolving Syst.*, 3(2), pp. 203–220.
- Mendel, J. (2001). *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*. Upper Saddle River: Prentice Hall.
- Mendel, J. and John, R. (2002). Type-2 fuzzy sets made simple. *IEEE Trans. Fuzzy Syst.*, 10(2), pp. 117–127.
- Mercer, J. (1909). Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. R. Soc. A*, 209, pp. 441–458.
- Merched, R. and Sayed, A. (1999). Fast RLS laguerre adaptive filtering. In *Proc. Allerton Conf. Commun., Control Comput.*, Allerton, IL, pp. 338–347.
- Moe-Helgesen, O.-M. and Stranden, H. (2005). Catastrophic forgetting in neural networks. Technical report. Trondheim, Norway: Norwegian University of Science and Technology.
- Morreale, P., Holtz, S. and Goncalves, A. (2013). Data mining and analysis of large scale time series network data. In *Proc. 27th Int. Conf. Adv. Inf. Netw. Appl. Workshops (WAINA)*. Barcelona, Spain, pp. 39–43.
- Mouss, H., Mouss, D., Mouss, N. and Sefouhi, L. (2004). Test of Page–Hinkley, an approach for fault detection in an agro-alimentary production system. In *Proc. Asian Control Conf.*, 2, pp. 815–818.
- Nakashima, Y. Y. T., Schaefer, G. and Ishibuchi, H. (2006). A weighted fuzzy classifier and its application to image processing tasks. *Fuzzy Sets Syst.*, 158(3), pp. 284–294.
- Nauck, D. and Kruse, R. (1998). NEFCLASS-X — a soft computing tool to build readable fuzzy classifiers. *BT Technol. J.*, 16(3), pp. 180–190.
- Nelles, O. (2001). *Nonlinear System Identification*. Berlin: Springer.

- Ngia, L. and Sjöberg, J. (2000). Efficient training of neural nets for nonlinear adaptive filtering using a recursive Levenberg–Marquardt algorithm. *IEEE Trans. Signal Process*, 48(7), pp. 1915–1926.
- Nguyen, H., Sugeno, M., Tong, R. and Yager, R. (1995). *Theoretical Aspects of Fuzzy Control*. New York: John Wiley & Sons.
- Nuntalid, N., Dhoble, K. and Kasabov, N. (2011). EEG classification with BSA spike encoding algorithm and evolving probabilistic spiking neural network. In *Neural Inf. Process., LNCS 7062*. Berlin Heidelberg: Springer Verlag, pp. 451–460.
- Pal, N. and Pal, K. (1999). Handling of inconsistent rules with an extended model of fuzzy reasoning. *J. Intell. Fuzzy Syst.*, 7, pp. 55–73.
- Pedrycz, W. and Gomide, F. (2007). *Fuzzy Systems Engineering: Toward Human-Centric Computing*. Hoboken, New Jersey: John Wiley & Sons.
- Piegat, A. (2001). *Fuzzy Modeling and Control*. Heidelberg, New York: Physica Verlag, Springer Verlag Company.
- Prasad, G., Leng, G., McGuinnity, T. and Coyle, D. (2010). Online identification of self-organizing fuzzy neural networks for modeling time-varying complex systems. In Angelov, P., Filev, D. and Kasabov, N. (eds.), *Evolving Intelligent Systems: Methodology and Applications*. New York: John Wiley & Sons, pp. 201–228.
- Pratama, M., Anavatti, S., Angelov, P. and Lughofer, E. (2014a). PANFIS: A novel incremental learning machine. *IEEE Trans. Neural Netw. Learn. Syst.*, 25(1), pp. 55–68.
- Pratama, M., Anavatti, S. and Lughofer, E. (2014b). GENEFIS: Towards an effective localist network. *IEEE Trans. Fuzzy Syst.*, 22(3), pp. 547–562.
- Pratama, M., Anavatti, S. and Lughofer, E. (2014c). pClass: An effective classifier to streaming examples. *IEEE Transactions on Fuzzy Systems*, 23(2), pp. 369–386.
- Quinlan, J. R. (1994). *C4.5: Programs for Machine Learning*. San Francisco, CA: Morgan Kaufmann Publishers.
- Reichmann, O., Jones, M. and Schildhauer, M. (2011). Challenges and opportunities of open data in ecology. *Science*, 331(6018), pp. 703–705.
- Reveiz, A. and Len, C. (2010). Operational risk management using a fuzzy logic inference system. *J. Financ. Transf.*, 30, pp. 141–153.
- Riaz, M. and Ghafoor, A. (2013). Spectral and textural weighting using Takagi–Sugeno fuzzy system for through wall image enhancement. *Prog. Electromagn. Res. B.*, 48, pp. 115–130.
- Rong, H.-J. (2012). Sequential adaptive fuzzy inference system for function approximation problems. In Sayed-Mouchaweh, M. and Lughofer, E. (eds.), *Learning in Non-Stationary Environments: Methods and Applications*. New York: Springer.
- Rong, H.-J., Han, S. and Zhao, G.-S. (2014). Adaptive fuzzy control of aircraft wing-rock motion. *Appl. Soft Comput.*, 14, pp. 181–193.
- Rong, H.-J., Sundararajan, N., Huang, G.-B. and Saratchandran, P. (2006). Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction. *Fuzzy Sets Syst.*, 157(9), pp. 1260–1275.
- Rong, H.-J., Sundararajan, N., Huang, G.-B. and Zhao, G.-S. (2011). Extended sequential adaptive fuzzy inference system for classification problems. *Evolving Syst.*, 2(2), pp. 71–82.
- Rosemann, N., Brockmann, W. and Neumann, B. (2009). Enforcing local properties in online learning first order TS fuzzy systems by incremental regularization. In *Proc. IFSA-EUSFLAT 2009*. Lisbon, Portugal, pp. 466–471.
- Rubio, J. (2009). SOFMLS: Online self-organizing fuzzy modified least square network. *IEEE Trans. Fuzzy Syst.*, 17(6), pp. 1296–1309.
- Rubio, J. (2010). Stability analysis for an on-line evolving neuro-fuzzy recurrent network. In Angelov, P., Filev, D. and Kasabov, N. (eds.), *Evolving Intelligent Systems: Methodology and Applications*. New York: John Wiley & Sons, pp. 173–199.
- Saminger-Platz, S., Mesiar, R. and Dubois, D. (2007). Aggregation operators and commuting. *IEEE Trans. Fuzzy Syst.*, 15(6), pp. 1032–1045.
- Sayed-Mouchaweh, M. and Lughofer, E. (2012). *Learning in Non-Stationary Environments: Methods and Applications*.

New York: Springer.

- Schölkopf, B. and Smola, A. (2002). *Learning with Kernels — Support Vector Machines, Regularization, Optimization and Beyond*. London, England: MIT Press.
- Sculley, D. (2007). Online active learning methods for fast label efficient spam filtering. In *Proc. Fourth Conf. Email AntiSpam*. Mountain View, California.
- Sebastiao, R., Silva, M., Rabico, R., Gama, J. and Mendonca, T. (2013). Real-time algorithm for changes detection in depth of anesthesia signals. *Evolving Syst.*, 4(1), pp. 3–12.
- Senge, R. and Hüllermeier, E. (2011). Top-down induction of fuzzy pattern trees. *IEEE Trans. Fuzzy Syst.*, 19(2), pp. 241–252.
- Serdio, F., Lughofer, E., Pichler, K., Buchegger, T. and Efendic, H. (2014a). Residual-based fault detection using soft computing techniques for condition monitoring at rolling mills. *Inf. Sci.*, 259, pp. 304–320.
- Serdio, F., Lughofer, E., Pichler, K., Pichler, M., Buchegger, T. and Efendic, H. (2014b). Fault detection in multi-sensor networks based on multivariate time-series models and orthogonal transformations. *Inf. Fusion*, 20, pp. 272–291.
- Settles, B. (2010). Active learning literature survey. Technical report, Computer Sciences Technical Report 1648. Madison: University of Wisconsin.
- Shaker, A. and Hüllermeier, E. (2012). IBLStreams: a system for instance-based classification and regression on data streams. *Evolving Syst.*, 3, pp. 239–249.
- Shaker, A. and Lughofer, E. (2014). Self-adaptive and local strategies for a smooth treatment of drifts in data streams. *Evolving Syst.*, 5(4), pp. 239–257.
- Shaker, A., Senge, R. and Hüllermeier, E. (2013). Evolving fuzzy patterns trees for binary classification on data streams. *Inf. Sci.*, 220, pp. 34–45.
- Sherman, J. and Morrison, W. (1949). Adjustment of an inverse matrix corresponding to changes in the elements of a given column or a given row of the original matrix. *Ann. Math. Stat.*, 20, p. 621.
- Shilton, A., Palaniswami, M., Ralph, D. and Tsoi, A. (2005). Incremental training of support vector machines. *IEEE Trans. Neural Netw.*, 16(1), pp. 114–131.
- Silva, A. M., Caminhas, W., Lemos, A. and Gomide, F. (2014). A fast learning algorithm for evolving neo-fuzzy neuron. *Appl. Soft Comput.*, 14(B), pp. 194–209.
- Skrjanc, I. (2009). Confidence interval of fuzzy models: An example using a waste-water treatment plant. *Chemometri. Intell. Lab. Syst.*, 96, pp. 182–187.
- Smithson, M. (2003). *Confidence Intervals*. SAGE University Paper, Series: Quantitative Applications in the Social Sciences. Thousand Oaks, California.
- Smola, A. and Schölkopf, B. (2004). A tutorial on support vector regression. *Stat. Comput.*, 14, pp. 199–222.
- Soleimani, H., Lucas, K. and Araabi, B. (2010). Recursive Gath-Geva clustering as a basis for evolving neuro-fuzzy modeling. *Evolving Syst.*, 1(1), pp. 59–71.
- Stehman, V. (1997). Selecting and interpreting measures of thematic classification accuracy. *Remote Sens. Environ.*, 62(1), pp. 77–89.
- Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *J. R. Stat. Soc.*, 36(1), pp. 111–147.
- Subramanian, K., Savita, R. and Suresh, S. (2013). A meta-cognitive interval type-2 fuzzy inference system classifier and its projection based learning algorithm. In *Proc. IEEE EAIS 2013 Workshop (SSCI 2013 Conf.)*, Singapore, pp. 48–55.
- Sun, H. and Wang, S. (2011). Measuring the component overlapping in the gaussian mixture model. *Data Min. Knowl. Discov.*, 23, pp. 479–502.
- Takagi, T. and Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. Syst., Man Cybern.*, 15(1), pp. 116–132.
- Tschumitschew, K. and Klawonn, F. (2012). Incremental statistical measures. In Sayed-Mouchaweh, M. and Lughofer, E. (eds.), *Learning in Non-Stationary Environments: Methods and Applications*. New York: Springer, pp. 21–55.

- Tsymbal, A. (2004). The problem of concept drift: definitions and related work. Technical Report TCD-CS-2004-15. Trinity College Dublin, Ireland, Department of Computer Science.
- Tung, S., Quek, C. and Guan, C. (2013). eT2FIS: An evolving type-2 neural fuzzy inference system. *Inf. Sci.*, 220, pp. 124–148.
- Wang, L. and Mendel, J. (1992). Fuzzy basis functions, universal approximation and orthogonal least-squares learning. *IEEE Trans. Neural Netw.*, 3(5), pp. 807–814.
- Wang, L., Ji, H.-B. and Jin, Y. (2013). Fuzzy passive-aggressive classification: A robust and efficient algorithm for online classification problems, *Inf. Sci.*, 220, pp. 46–63.
- Wang, W. and Vrbanek, J. (2008). An evolving fuzzy predictor for industrial applications. *IEEE Trans. Fuzzy Syst.*, 16(6), pp. 1439–1449.
- Werbos, P. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis. Harvard University, USA: Appl. Math.
- Wetter, T. (2000). Medical decision support systems. In *Medical Data Analysis*. Berlin/Heidelberg: Springer, pp. 458–466.
- White, T. (2012). *Hadoop: The Definitive Guide*. O'Reilly Media.
- Widiputra, H., Pears, R. and Kasabov, N. (2012). Dynamic learning of multiple time series in a nonstationary environment. In Sayed-Mouchaweh, M. and Lughofer, E. (eds.), *Learning in Non-Stationary Environments: Methods and Applications*. New York: Springer, pp. 303–348.
- Widmer, G. and Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts, *Mach. Learn.*, 23(1), pp. 69–101.
- Wu, X., Kumar, V., Quinlan, J., Gosh, J., Yang, Q., Motoda, H., MacLachlan, G., Ng, A., Liu, B., Yu, P., Zhou, Z.-H., Steinbach, M., Hand, D. and Steinberg, D. (2006). Top 10 algorithms in data mining. *Knowl. Inf. Syst.* 14(1), pp. 1–37.
- Xu, Y., Wong, K. and Leung, C. (2006). Generalized recursive least square to the training of neural network. *IEEE Trans. Neural Netw.*, 17(1), pp. 19–34.
- Xydeas, C., Angelov, P., Chiao, S. and Reoulas, M. (2006). Advances in eeg signals classification via dependant hmm models and evolving fuzzy classifiers. *Int. J. Comput. Biol. Med.*, special issue on Intell. Technol. Bio-inf. Med., 36(10), pp. 1064–1083.
- Yager, R. R. (1990). A model of participatory learning. *IEEE Trans. Syst., Man Cybern.*, 20(5), pp. 1229–1234.
- Ye, J., Li, Q., Xiong, H., Park, H., Janardan, R. and Kumar, V. (2005). IDR, QR: An incremental dimension reduction algorithms via QR decomposition. *IEEE Trans. Knowl. Data Eng.*, 17(9), pp. 1208–1222.
- Zaanen, A. (1960). *Linear Analysis*. Amsterdam: North Holland Publishing Co.
- Zadeh, L. (1965). Fuzzy sets. *Inf. Control*, 8(3), pp. 338–353.
- Zadeh, L. (1975). The concept of a linguistic variable and its application to approximate reasoning. *Inf. Sci.*, 8(3), pp. 199–249.
- Zavoianu, A. (2010). *Towards Solution Parsimony in an Enhanced Genetic Programming Process*. PhD thesis. Linz, Austria: Johannes Kepler University Linz.
- Zdsar, A., Dovzan, D. and Skrjanc, I. (2014). Self-tuning of 2 DOF control based on evolving fuzzy model. *Appl. Soft Comput.*, 19, pp. 403–418.
- Zhou, X. and Angelov, P. (2006). Real-time joint landmark recognition and classifier generation by an evolving fuzzy system. In *Proc. FUZZ-IEEE 2006*. Vancouver, Canada, pp. 1205–1212.
- Zhou, X. and Angelov, P. (2007). Autonomous visual self-localization in completely unknown environment using evolving fuzzy rule-based classifier. In *2007 IEEE Int. Conf. Comput. Intell. Appl. Def. Secur.*, Honolulu, Hawaii, USA, pp. 131–138.

¹http://en.wikipedia.org/wiki/Big_data.

²http://en.wikipedia.org/wiki/Very_large_database.

³http://en.wikipedia.org/wiki/Evolving_intelligent_system.

⁴<http://www.springer.com/physics/complexity/journal/12530>.

⁵http://en.wikipedia.org/wiki/Incremental_heuristic_search.

⁶http://en.wikipedia.org/wiki/Very_large_database.

⁷<http://www.springer.com/physics/complexity/journal/12530>.

⁸<http://www.springer.com/physics/complexity/journal/12530>.

⁹<http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=91>.

¹⁰<http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=5962385>.

¹¹<http://moa.cms.waikato.ac.nz/>.

¹²http://en.wikipedia.org/wiki/Evolving_intelligent_system.

¹³<http://www.csie.ntu.edu.tw/cjlin/libsvm/>.

Chapter 4

Modeling Fuzzy Rule-Based Systems

Rashmi Dutta Baruah and Diganta Baruah

The objective of this chapter is to familiarize the reader with various approaches used for fuzzy modeling. It is expected that the discussion presented in the chapter would enable the reader to understand, compare, and choose between the different fuzzy modeling alternatives, both in knowledge-based and automated approaches, so as to apply it to model real-world systems. It is assumed that the reader has a basic knowledge of fuzzy set theory and linear algebra. However, illustrative examples are provided throughout the text to comprehend the basic concepts. The chapter is organized in five sections. In the first section, fuzzy systems are introduced very briefly as a foundation to start the discussion on the fuzzy modeling techniques. Next, an overview of design issues is presented in the same section. [Section 4.2](#) describes the knowledge-based methods and [Section 4.3](#) discusses the automated methods. The automated methods include template-based method, neuro-fuzzy approaches, genetic-fuzzy approaches, and clustering-based approaches. [Section 4.4](#) presents a brief description on another automated approach, viz., online approach that has recently grabbed a lot of attention from the researchers. Finally, in [Section 4.5](#), a short summary of the chapter is presented.

4.1. Introduction

In simple terms, a system that uses fuzzy sets and fuzzy logic in some form can be considered as a fuzzy system. For example, the system's input and output can be represented using fuzzy sets or the system itself can be defined in terms of fuzzy *if-then* rules. In this chapter, the focus is on fuzzy systems that are based on fuzzy *if-then* rules which use fuzzy set theory to make decisions or draw conclusions. Such systems are often referred to as *fuzzy rule-based* (FRB) systems. For simplicity, here we will be referring to a FRB system as fuzzy system. Fuzzy systems can be broadly classified into two families: Mamdani-type (Mamdani, 1997) and Takagi– Sugeno-type (Takagi and Sugeno, 1985). In the Mamdani-type, also called linguistic systems, rules are represented as:

$$\text{if } x_1 \text{ is } A_1 \text{ and } x_2 \text{ is } A_2 \text{ and } \dots \text{ and } x_n \text{ is } A_n \text{ then } y \text{ is } B, \quad (1)$$

where x_i , $i = 1, 2, \dots, n$, is the i th input variable, and A_i and B are the linguistic terms (e.g., Small, Large, High, Low, etc.) defined by fuzzy sets, and y is the output associated with the given rule.

The rule structure of the Takagi–Sugeno-type (TSK), also called functional type, is usually given as:

$$\text{if } x_1 \text{ is } A_1 \text{ and } x_2 \text{ is } A_2 \text{ and } \dots \text{ and } x_n \text{ is } A_n \text{ then } y = f(x_1, x_2, \dots, x_n), \quad (2)$$

where x_i , $i = 1, 2, \dots, n$, is the i th input variable, A_i is the antecedent fuzzy set, y is the output of the rule, and f is a real valued function. If f is a constant then the rule is of zero-order type and the corresponding system is called zero-order TSK system, and if f is first order polynomial then the rule is of first-order type and the resulting system is called first-order system as given below:

- Zero-order TSK rule

$$\text{if } x_1 \text{ is } A_1 \text{ and } x_2 \text{ is } A_2 \text{ and } \dots \text{ and } x_n \text{ is } A_n \text{ then } y = a, \quad (3)$$

where a is a real number.

- First-order TSK rule

$$\begin{aligned} & \text{if } x_1 \text{ is } A_1 \text{ and } x_2 \text{ is } A_2 \text{ and } \dots \text{ and } x_n \text{ is } A_n \text{ then } y \\ &= a_0 + a_1 x_1 + a_2 x_2 + \dots + a_n x_n, \end{aligned} \quad (4)$$

where a_i is the i th consequence parameter.

Thus, in the Mamdani-type, the consequent of each rule is a fuzzy set whereas in the Sugeno-type the consequent is a function of input variables. Due to this difference, the inference mechanism of determining the output of the system in both the categories somewhat varies.

The early approaches to FRB system design involve representing the knowledge and experience of a human expert, associated with a particular system, in terms of *if-then* rules. To avoid the difficult task of knowledge acquisition and to improve the system performance another alternative is to use expert knowledge as well as system generated input-output data. This fusion of expert knowledge and data can be done in many ways. For example, one way is to combine linguistic rules from human expert and rules learnt by numerical data, and another way is to derive the initial structure and parameters from expert knowledge and optimize the parameters using input-output data by applying machine learning techniques. The common approaches used for learning and adjusting the parameters from the training data are neural networks and genetic algorithms (GAs). These approaches are often referred to as *automated methods* or *data-driven methods*. Due to the complexity of systems and availability of huge data, presently the automated approaches are commonly used for fuzzy modeling.

The majority of applications of fuzzy systems are in the area of process control (Fuzzy Logic Control- FLC), decision-making (Fuzzy Expert Systems), estimation (prediction, forecasting), and pattern recognition (classification). Regardless of the type of application most systems are based on simple fuzzy *if-then* rules, and so have a common structure. Thus, the basic design steps are same for all such FRB systems. In general, the design of a fuzzy system or fuzzy system modeling is a multi-step process that involves the following:

- Formulation of the problem.
- Specification of the fuzzy sets.
- Generation of rule set.
- Selection of fuzzy inference and defuzzification mechanism.

In the knowledge-based approach, all the steps are completed based on intuition and experience. The automated methods usually require designer's involvement only in problem formulation and selection of fuzzy inference and defuzzification mechanism. In the following sections, we describe all the design steps by considering a simple control application.

4.2. Knowledge-Based Approach

The knowledge-based approach can be applied to linguistic fuzzy models. We illustrate the steps to obtain a linguistic fuzzy model for a simple cooling fan system. The problem is to control the regulator knob of a cooling fan inside a room based on two inputs temperature and humidity. The speed of fan increases as the knob is turned right and decreases when it is turned left. The scenario is depicted in [Figure 4.1](#).

4.2.1. Formulation of the Problem

In problem formulation step, the problem is defined in terms of the system input, the required output, and the objectives. For example, to model a fuzzy control system typically one needs to state: What is to be controlled? What are the input variables? What kind of response is expected from the control system? What are the possible system failure states? The selection of input and output variables and the range of the variables are mainly dependent on the problem at hand and the designer sensibility.

For example, consider the cooling fan system problem given in [Figure 4.1](#). For this problem, the inputs are temperature and humidity measures and the output is the action in terms of turning the knob right or left in small or big steps.

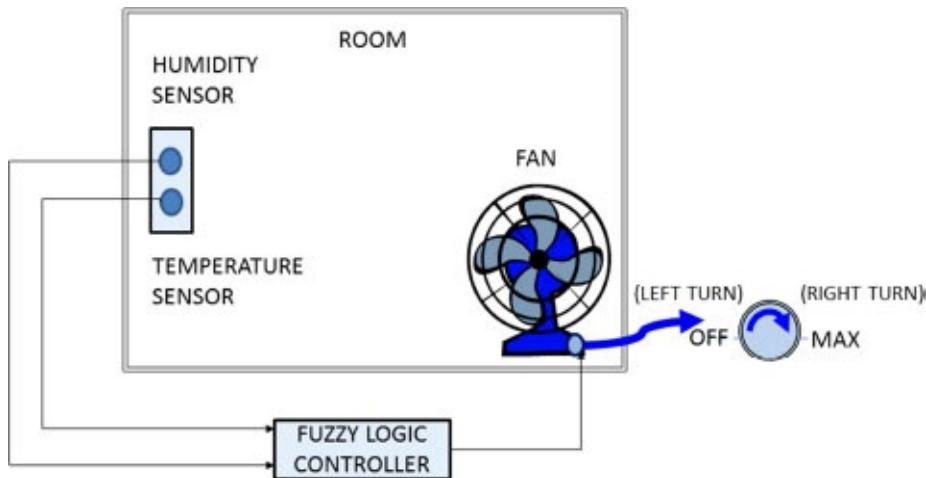


Figure 4.1: Cooling fan system.

Table 4.1: Temperature ranges and linguistic values.

Temperature range ($^{\circ}\text{C}$)	Linguistic values
15–19	Low
18–25	Normal
24–30	High

4.2.2. Specification of Fuzzy Sets

The fuzzy sets are specified in terms of membership functions. For each input and output variable, the linguistic values (or fuzzy terms or labels) are specified before defining the membership functions. For the system depicted in [Figure 4.1](#), if the range of the input

variable temperature is 15–30°C, then the linguistic values, *Low*, *Normal*, and *High* can be associated with it (see [Table 4.1](#)). Now, the membership functions for temperature can be defined as shown in [Figure 4.2](#). Similarly, the membership functions for the input humidity and the output action is shown in [Figures 4.3](#) and [4.4](#). It is assumed that the range of relative *humidity* is 20–70% and the knob can be rotated in degrees from –3 to 3.

The number and shape of the membership functions depend on application domain, experts' knowledge and are chosen subjectively or (and) are generated automatically. An important observation is that usually three to seven membership functions are used. As the number grows, it becomes difficult to manage especially in case of manual design. Many fuzzy logic control problems assume linear membership functions usually Triangular in shape, the other commonly used membership functions are Trapezoidal and Gaussian. The other modeling issues can be the amount of overlapping between the membership functions and the distribution of membership functions (even or uneven distribution) ([Figures 4.5a](#) and [4.5b](#)). Also, any point in the range of the input variables has to be covered by at least one fuzzy set participating to at least one rule ([Figure 4.5c](#)). It is recommended that each membership function overlaps only with the closest neighboring membership function ([Figure 4.5d](#)). In uneven distribution some of the membership functions can cover smaller range to achieve precision ([Figure 4.5b](#)). Currently, more focus is towards identification of these parameters automatically using machine learning techniques.

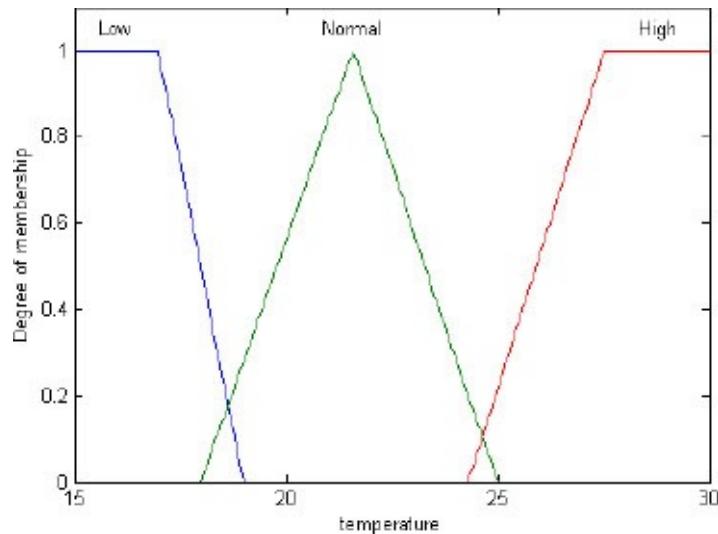


Figure 4.2: Fuzzy membership functions for input variable temperature.

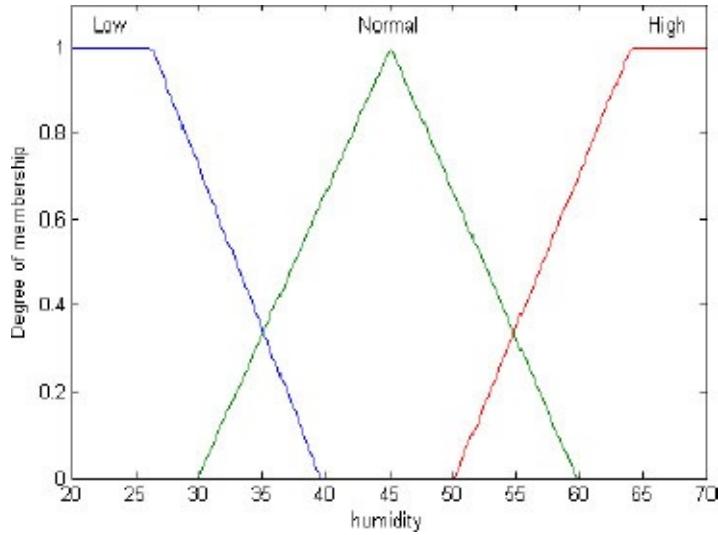


Figure 4.3: Fuzzy membership functions for input variable humidity.

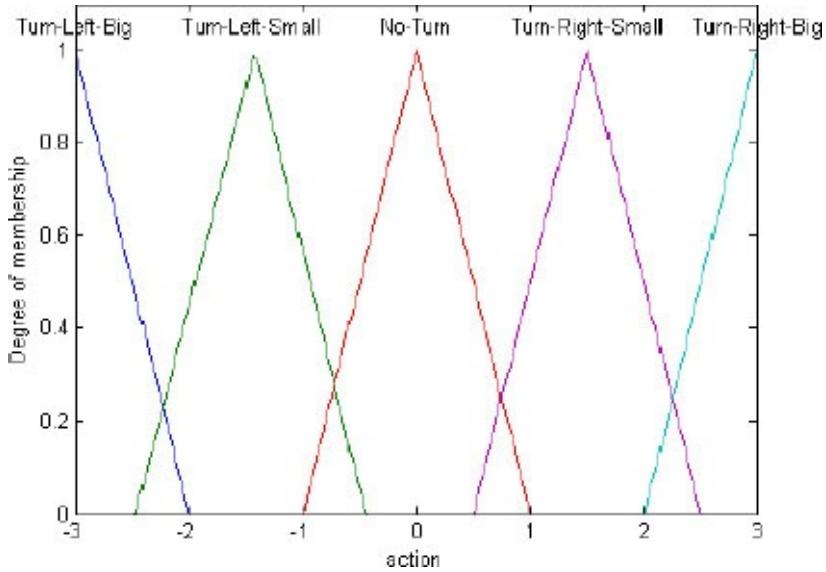


Figure 4.4: Fuzzy membership functions for output variable action.

4.2.3. Generation of Rule Set

A rule maps the input to output and the set of rules mainly defines the fuzzy model. For a simple system with two inputs and single output, the rules can be enumerated with the help of a matrix. For example, for the system shown in [Figure 4.1](#), the matrix is given in [Table 4.2](#) and the corresponding rules are given in [Figure 4.6](#). For a system with more than two inputs and outputs the rules can be presented in a tabular form. For example, [Table 4.3](#) gives the partial rule-set of the fan control system when another input *rate-of-change of temperature* (ΔT) is added.

4.2.4. Selection of Fuzzy Inference and Defuzzification Mechanism

After generating the rule-set, it is required to specify how the system would calculate the final output for a given input. This can be specified in terms of inference and defuzzification mechanism. The designer needs to specify how to infer or compute the fuzzy operator in the antecedent part, how to get the fuzzy output from each rule, and

finally how to aggregate these outputs to get a single crisp output.

Usually the fuzzy rules involve more than one fuzzy set connected with fuzzy operators (AND, OR, NOT). The inference mechanism depends on the fuzzy combination operators used in the rules. These fuzzy combination operators are also referred to as *T-norms*. There are many ways to compute these operators. A designer needs to specify which one to apply to the specific system. For example, the AND operator is expressed as,

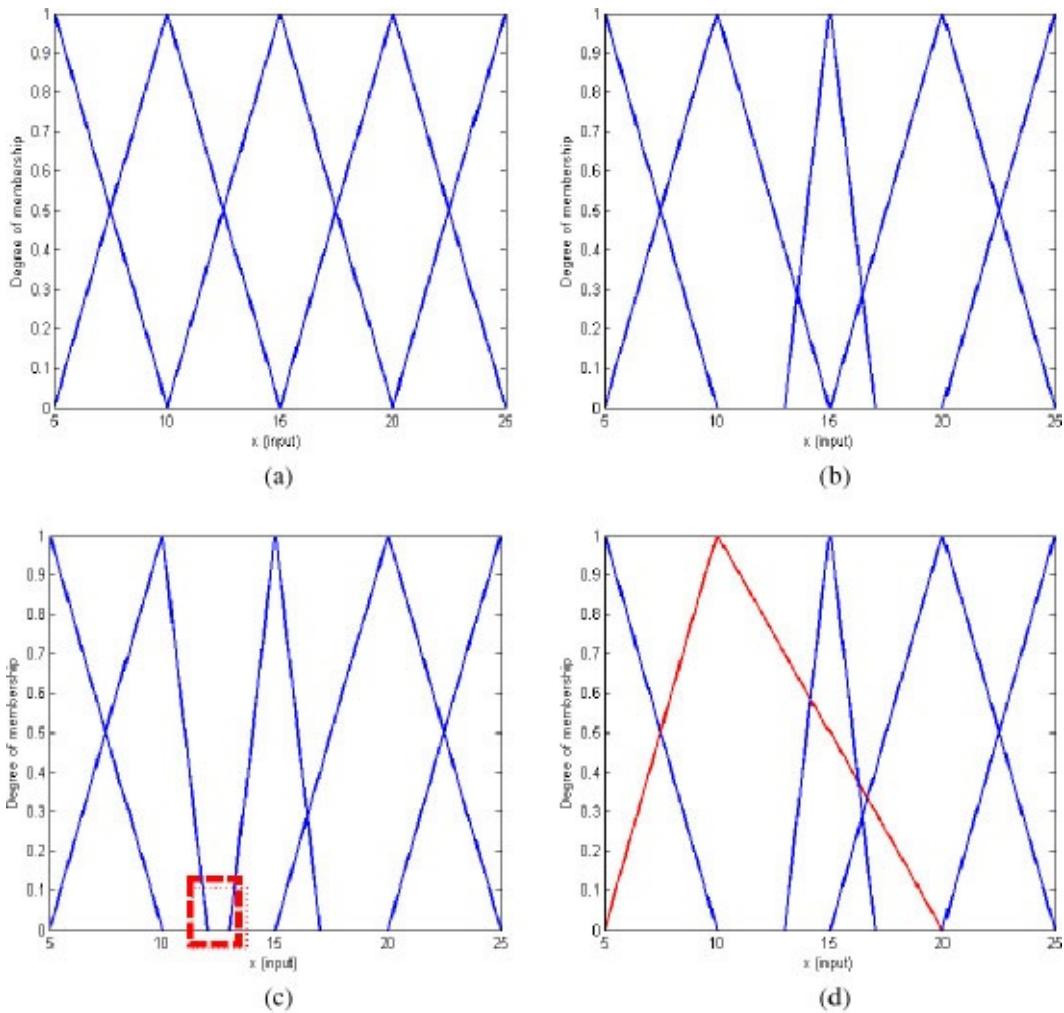


Figure 4.5: (a) MFs evenly distributed, (b) MFs unevenly distributed, (c) MFs not covering all the input points, (d) MFs overlapping with more than one neighbor.

Table 4.2: Rule matrix for cooling fan system.

Temperature(T)/ humidity (H)	Low	Normal	High
Low	Turn left big	No turn	Turn right small
Normal	Turn left small	No turn	Turn right small
High	Turn right small	Turn right small	Turn right big

- Rule1: *if T is low and H is low then turn the knob left big.*
- Rule2: *if T is low and H is normal then no need to turn the knob.*
- Rule3: *if T is low and H is high then turn the knob right small.*
- Rule4: *if T is normal and H is low then no need to turn the knob.*
- Rule5: *if T is normal and H is normal then no need to turn the knob.*
- Rule6: *if T is normal and H is high then turn the knob right small.*
- Rule7: *if T is high and H is low then turn the knob right small.*
- Rule8: *if T is high and H is normal then turn the knob right small.*
- Rule9: *if T is high and H is high then turn the knob right big.*

Figure 4.6: Rule set for fan speed control.

Table 4.3: Rule table for cooling fan system considering three inputs.

Input 1 temperature	Input 2 humidity	Input 3 ΔT	Output action
Low	Low	Decreasing	Turn left big
Low	Low	Steady	Turn left big
Low	Low	Increasing	Turn right small

$\mu_{A \cap B} = T(\mu_A(x), \mu_B(x))$, $\mu_A(x)$ is membership of x in fuzzy set A and $\mu_B(x)$ is membership of x in fuzzy set B.

The two most commonly used methods to compute AND are:

- $T(\mu_A(x), \mu_B(x)) = \min(\mu_A(x), \mu_B(x))$, this method is widely known as Zadeh method.
- $T(\mu_A(x), \mu_B(x)) = \mu_A(x) \cdot \mu_B(x)$ this method is Product method that computes the fuzzy AND by simply multiplying the two membership values.

Example 2.1. Consider the scenario presented in [Figure 4.1](#) and assume that the present room temperature is 18.5°C and humidity is 36.5%. Also, consider that the Zadeh (or minimum) method is used as fuzzy combination operator.

For a given input, first its degree of membership to each of the input fuzzy sets is determined; this step is often referred to as fuzzification. After fuzzification, the firing strength of each rule can be determined in the following way:

$$\begin{aligned} \text{Rule 1: } & T(\mu_{\text{Low}}(18.5), \mu_{\text{Low}}(36.5)) \\ &= \min(\mu_{\text{Low}}(18.5), \mu_{\text{Low}}(36.5)) = \min(0.5, 0.25) = 0.25, \end{aligned}$$

$$\begin{aligned} \text{Rule 2: } & T(\mu_{\text{Low}}(18.5), \mu_{\text{Normal}}(36.5)) \\ &= \min(\mu_{\text{Low}}(18.5), \mu_{\text{Normal}}(36.5)) = \min(0.5, 0.43) = 0.43, \end{aligned}$$

$$\begin{aligned} \text{Rule 3: } & T(\mu_{\text{Low}}(18.5), \mu_{\text{High}}(36.5)) \\ &= \min(\mu_{\text{Low}}(18.5), \mu_{\text{High}}(36.5)) = \min(0.5, 0) = 0. \end{aligned}$$

Thus, for the given input, Rule 1 applies (or triggers or fires) at 25%, Rule 2 applies at

43%, and Rule 3 at 0%, i.e., it does not apply at all. This means the *then* part or the action of Rule 1, Rule 2 fires at strength of 0.25 and 0.43 respectively. The firing strengths of the remaining rules are determined in a similar way, and the value is 0 for Rule 4 to Rule 9.

After determining the firing strength of each rule, the consequent of the rule is obtained or the rule conclusion is inferred. The commonly used method to obtain the rule consequent is by clipping the output membership function at the rule strength. [Figure 4.7](#) shows the degree of membership of input to each of the input fuzzy sets and the inferred conclusions using the clipping method.

There are number of defuzzification techniques such as centroid method which is also known as center of area or center of gravity (COG) method, weighted average method, Maximum membership method (or height method), Mean (first or last) of maxima method, etc. Each method has its own advantages and disadvantages. For example, both the maximum membership method and weighted average method are computationally faster and simpler. However, in terms of accuracy, weighted average is better as maximum membership accounts only for rules, which are triggered at the maximum membership level. The COG method is most widely used and provides better results, however computational overhead is more and it has disadvantage of not allowing control actions towards the extremes of the action (output) range (Ross, 2010).

Example 2.2. Consider the aggregated output shown in [Figures 4.8](#) and [4.9](#). Application of COG method results in crisp output of -0.6, i.e., the knob is required to be turned 0.6 degrees to the left.

The COG method is given by $z^* = \frac{\int \mu_{\text{output}}(z) \cdot z dz}{\int \mu_{\text{output}}(z) dz}$, where z^* is the defuzzified or crisp output, μ_{output} is the aggregated resultant membership function of the two output fuzzy sets, and z is the universe of discourse. For this example, the COG method requires calculation of the blue shaded area in the graph shown in [Figure 4.9](#). This area can be determined by adding the area A, B, C, D, and E. The detailed calculations involved in the COG method are provided on [page 148](#) based on [Figures 4.8](#) and [4.9](#).

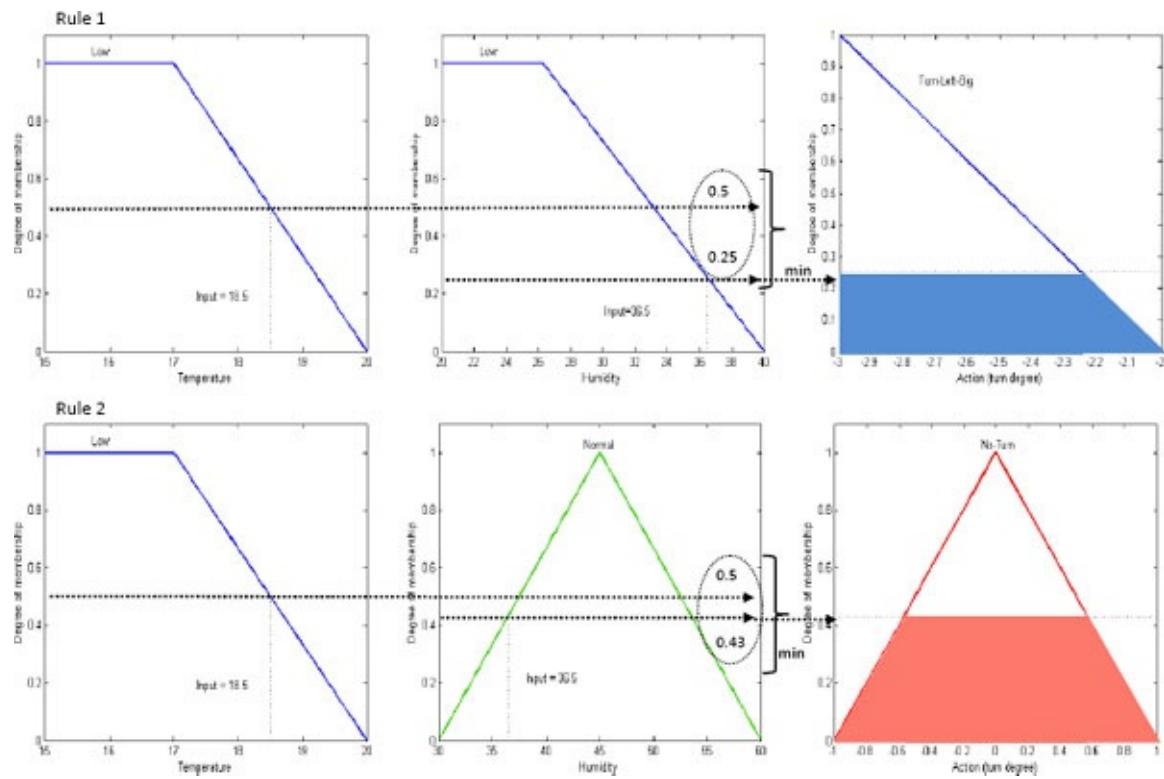


Figure 4.7: Membership degrees and inferred conclusions (Example 2.1).

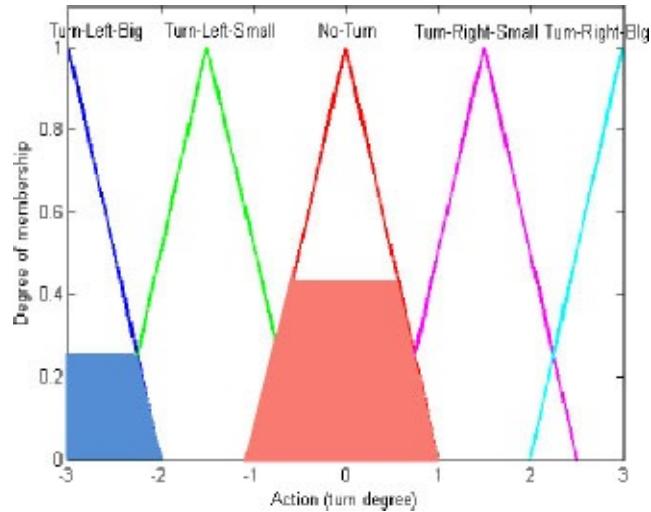


Figure 4.8: Rule consequents from Mamdani inference system (Example 2.2).

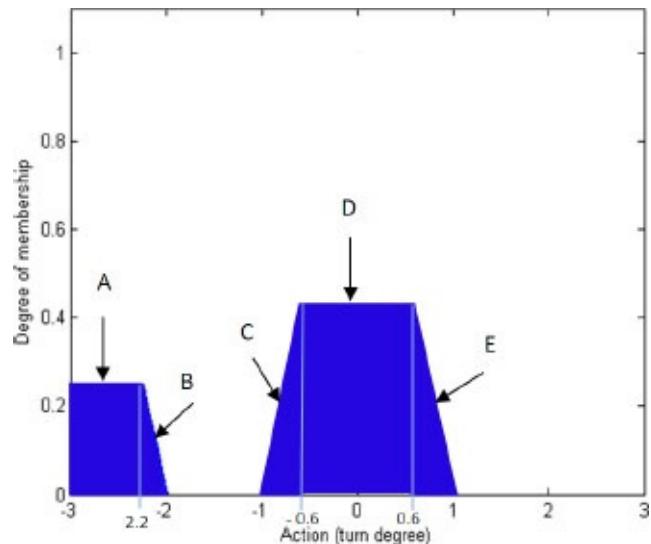


Figure 4.9: Result of aggregation (Example 2.2).

Considering Figure 4.9 and Example 2.2, the COG defuzzification can be given as:

$$\begin{aligned}
 z^* &= \frac{\int \mu_{\text{output}}(z) \cdot z \, dz}{\int \mu_{\text{output}}(z) \, dz} \\
 &= \frac{\int_{-3}^{-2.2} (0.25)z \, dz + \int_{-2.2}^{-2} (-1.25z - 2.5)z \, dz + \int_{-1}^{-0.6} (1.075z + 1.075)z \, dz \\
 &\quad + \int_{-0.6}^{0.6} (0.43)z \, dz + \int_{0.6}^1 (-1.075z + 1.075)z \, dz}{\int_{-3.0}^{-2.2} (0.25)dz + \int_{-2.2}^{-2.0} (-1.25z - 2.5)dz + \int_{-1}^{-0.6} (1.075z + 1.075)dz \\
 &\quad + \int_{-0.6}^{0.6} (0.43)dz + \int_{0.6}^1 (-1.075z + 1.075)dz} \\
 &= \frac{(-0.52) + (-0.053) + (-0.063) + 0.063}{0.2 + 0.025 + 0.086 + 0.516 + 0.086} = -0.627 \approx -0.6.
 \end{aligned}$$

The first three modeling steps are common to both Mamdani systems and Sugeno systems. The primary difference is that in Sugeno systems, the output consequence is not computed by clipping an output membership function at the rule strength. The reason is that in Sugeno systems there is no output membership function at all. Instead the output is a crisp number which is either a constant or computed by multiplying each input by a constant and then adding up the results. In the latter case, the output is a linear function of inputs.

Example 2.3. Consider the same system depicted in Figure 4.1 and the inputs as 18.5°C temperature and 36.5% humidity. As shown in Example 2.2, only the first three rules will be fired for the given inputs. Let us assume that the system is Sugeno type (zero-order) and the first three rules are given as:

Rule 1: *If T is low and H is low then -2.5*

Rule 2: *If T is low and H is normal then 0*

Rule 3: *If T is low and H is high then 1.5*

The firing strength of a rule is computed using product method as shown below:

$$\begin{aligned}
 \text{Rule 1: } &T(\mu_{\text{Low}}(18.5), \mu_{\text{Low}}(36.5)) \\
 &= \mu_{\text{Low}}(18.5) \cdot \mu_{\text{Low}}(36.5) = 0.5 \cdot 0.25 = 0.125, \\
 \text{Rule 2: } &T(\mu_{\text{Low}}(18.5), \mu_{\text{Normal}}(36.5)) \\
 &= \mu_{\text{Low}}(18.5) \cdot \mu_{\text{Normal}}(36.5) = 0.5 \cdot 0.43 = 0.215, \\
 \text{Rule 3: } &T(\mu_{\text{Low}}(18.5), \mu_{\text{High}}(36.5)) \\
 &= \mu_{\text{Low}}(18.5) \cdot \mu_{\text{High}}(36.5) = 0.5 \cdot 0 = 0.
 \end{aligned}$$

The final output of the system is the weighted average of all the rule outputs,

computed as

$$\hat{y} = \frac{\sum_{i=1}^N w_i y_i}{\sum_{i=1}^N w_i},$$

w_i is the weight and is same as the firing strength of rule i , y_i is the output of rule i , \hat{y} is the final output, and N is the number of rules.

$$\hat{y} = \frac{0.125 \times (-2.5) + 0.215 \times 0 + 0 \times 1.5}{0.125 + 0.215 + 0} = -0.92,$$

which indicates the knob is required to be turned 0.9° to the left.

Selection of defuzzification method is context or problem dependent. However, Hellendroon and Thomas (1993) have provided five criteria against which the defuzzification methods can be measured. The first criterion is *continuity*; a small variation in input should not lead to large output variation. The second criterion is *disambiguity* that requires the output from the defuzzification method to be always unique. The third criterion called *plausibility* requires the output to lie approximately in the middle of the support region and have high degree of membership. *Computational simplicity* is the fourth criterion. Finally, the fifth one is *weighting method* that weights the output method. This criterion is problem dependent as judging the weighting methods do not have any straightforward rules. One can compare the computational simplicity of the weighting methods but this is already taken into account by the fourth criterion. However, it is interesting to note that the defuzzification methods that are commonly used do not comply with all the given criteria. Although time consuming, another method is to use simulation tool like MATLAB to compare the results with various defuzzification methods and make the selection based on the obtained results.

4.3. Automated Modeling Approaches

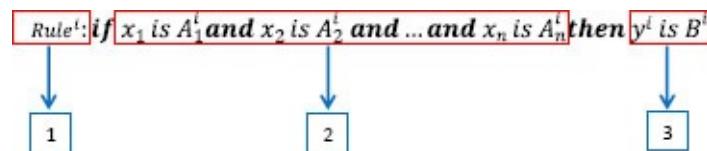
Let us consider the system depicted in [Figure 4.1](#) and assume that there are two persons in the room Alex and Baker. Alex turns the knob to the right or left depending on his comfort level, for example, he turns the knob to the right when he feels hot and humid. Baker is taking note of Alex's actions on particular temperature and humidity (readings from the sensors) in a log book. The recordings in Baker's book constitute the input–output data. The inputs are the temperature and humidity readings and the corresponding outputs are Alex's actions on those temperature and humidity readings. Such input–output data for a (manual or existing) system can be collected through various means. [Table 4.4](#) represents a small set of input–output data with 10 data samples for this cooling fan system. Automated methods make use of such input–output data for fuzzy system modeling, in this case the modeling of an automatic fuzzy cooling fan control.

[Table 4.4:](#) Example input–output data for cooling fan system.

x_1 (Temperature)	16	18	20	20	25	25	28	26	26	23
x_2 (Humidity)	25	25	20	40	50	60	65	55	50	55
y (Action)	-3	-3	-2	0	0	2	3	1	1	1

As discussed in the previous section, the modeling process mainly requires specification of two main components of a fuzzy system: inference mechanism and rule-base. The fuzzy inference needs specification of the implication, aggregation, and defuzzification mechanism, which can be resolved by the designer subjectively. In this section, we will discuss automated methods that can be used to construct the rule-base. However, it should be noted that these automated methods do not completely eliminate the involvement of a designer.

Rules constitutes a rule-base and if we look at a rule closely ([Figure 4.10](#)), then we need to address the following issues:



[Figure 4.10:](#) A fuzzy linguistic rule.

1. How many rules?
2. Antecedent part:
 - a. What are the antecedent fuzzy sets and corresponding membership functions (antecedent parameters)?
 - b. What is the type of membership function (Triangular, Trapezoidal, Gaussian etc.)?
 - c. How many membership functions?
 - d. Which operator (Fuzzy OR, AND or NOT) to use for connecting the rule

antecedents?

3. Consequent part:

- a. What are the consequent fuzzy sets and corresponding membership functions? If the rule is of TSK type then the issue is determining each consequent parameter.
- b. What is the type and number of each membership function?
- c. How many membership functions?

If sufficient input–output data from the system are available then issue 1 can be solved with automated methods. The remaining issues can be solved only partially through automated methods. The type of membership function (issue 2b, 3b) and the fuzzy connectives (issue 2d) still require designer involvement. As mentioned in the previous section, these issues can be resolved subjectively. Further, there are several comparative studies available in literature that can also guide the designer in selecting best possible fuzzy operator for a particular application (Beliakov and Warren, 2001; Cordon *et al.*, 1997).

There are two ways in which the automated methods can aid the fuzzy system modeling process. In some fuzzy systems, the rules and the membership functions and associated parameters, e.g., the center and spread of a Gaussian function, are defined by the designer (not necessarily taking into account the input–output data). The automated methods are then used to find better set of parameters using the input–output data. This process is commonly referred to as *tuning* the fuzzy system. On the other hand, the automated methods can be used to determine the rules and the membership functions along with parameters using the input–output data and without designer intervention. This process is referred to as *learning* the rule-base of the fuzzy system, and the input–output data used in the learning process is often referred to as *training data*.

4.3.1. *Template-Based Approach*

This approach combines the expert knowledge and input–output data. In this approach, the domains of the antecedent variable are simply partitioned into a specified number of membership functions. We explain the rule generation process considering the method developed by Wang and Mendel (1992).

The rules are generated by following steps:

- (i) *Partition the input and output spaces into fuzzy regions:* First, the domain intervals for the input and output variables, within which the values of the variable are expected to lie, are defined as:

$$X_1 = [x_1^-, x_1^+], \quad X_2 = [x_2^-, x_2^+], \quad Y = [y^-, y^+],$$

where x_i^- , x_i^+ , $i = 1, 2$ and y^- , y^+ are the upper and lower limits of the intervals for

input and output variables respectively. Each domain interval is divided into $2P + 1$ number of equal or unequal fuzzy partitions; P is an integer that can be different for different variables. Next, a type of membership function (triangular, trapezoidal, Gaussian) is selected and a fuzzy set is assigned to each partition.

- (ii) *Generate a preliminary rule set:* First, we determine the degree of each of the input and output data in all the partitions. A data is assigned to a partition and corresponding fuzzy set where it has maximum degree. Finally, one rule is formed for each input–output pair of data.
- (iii) *Assign degrees to each rule:* The step (ii) generates as many rules as the number of training data points. In such a scenario, the chance of getting conflicting rules is very high. The conflicting rules have same antecedent but different consequents. In this step such rules are removed by assigning degrees, and in the process the number of rules is also reduced. Suppose the following rule is extracted from the i th training example $[x_1^i, x_2^i, \dots, x_n^i, y^i]$.

Ruleⁱ: if x_1 is A_1 and x_2 is A_2 and ... and x_n is A_n then y is B ,

then the degree of this rule is given as,

$$\deg(Rule^i) = \mu_{A_1}(x_1^i) \cdot \mu_{A_2}(x_2^i) \cdots \mu_{A_n}(x_n^i) \cdot \mu_B(y^i).$$

Also, the degree of a rule can be represented as a product of the degree of its components and the degree of the training example that has generated this rule and can be given as,

$$\deg(Rule^i) = \mu_{A_1}(x_1^i) \cdot \mu_{A_2}(x_2^i) \cdots \mu_{A_n}(x_n^i) \cdot \mu_B(y^i) \cdot \mu^i,$$

where μ^i is the degree of the training example.

The degree of the training example is provided by the human expert based on its usefulness. For example, if the expert believes that a particular data sample is useful and crucial then a higher degree is assigned to it, and a lower degree is assigned for bad data sample that may be a measurement error.

- (iv) *Obtain the final set of rules from the preliminary set:* In this step the rules with high degrees are selected for each combination of antecedents.

Example 3.1. Let us consider the data in [Table 4.4](#) as the training data and the shape of the membership function to be triangular (other shapes for membership functions are also possible).

For the given training data and cooling fan system, the domain intervals are given as:

$$x_1 = [x_1^-, x_1^+] = [15, 30], \quad x_2 = [x_2^-, x_2^+] = [20, 70], \quad y = [y^-, y^+] = [-3, 3].$$

Next, the domain intervals are divided into equal size partitions as shown in [Figure](#)

4.11.

Also from Figure 4.11a, it can be observed that $x_1^1 = 16$ has maximum degree in the fuzzy set A1 and therefore it is assigned that fuzzy set. Similarly, $x_2^1 = 25$ is assigned to fuzzy set B2 (Figure 4.11b) and $y^1 = -3$ is assigned to C1 (Figure 4.12). Therefore, the rule corresponding to the first training data point [16, 25, -3] can be given as:

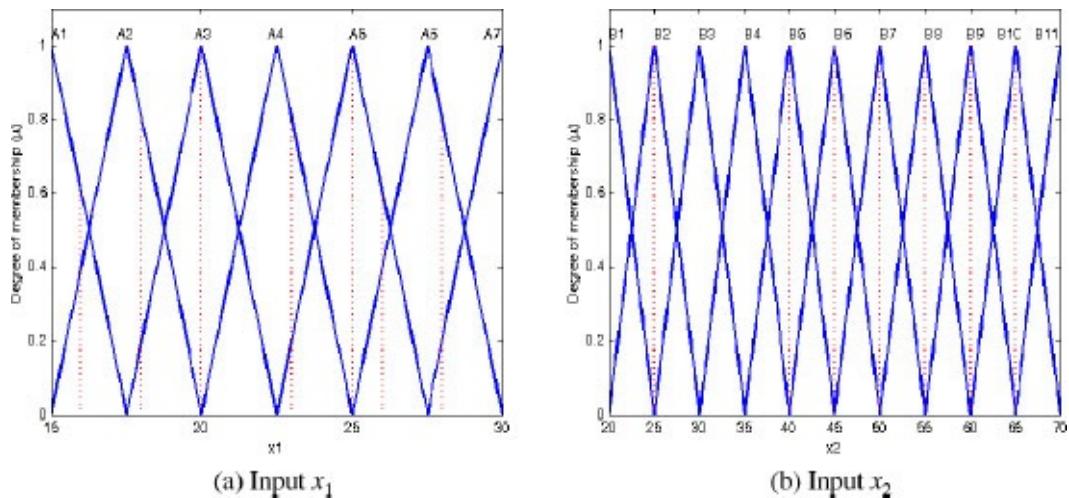


Figure 4.11: Partitions of input domain interval and corresponding fuzzy sets.

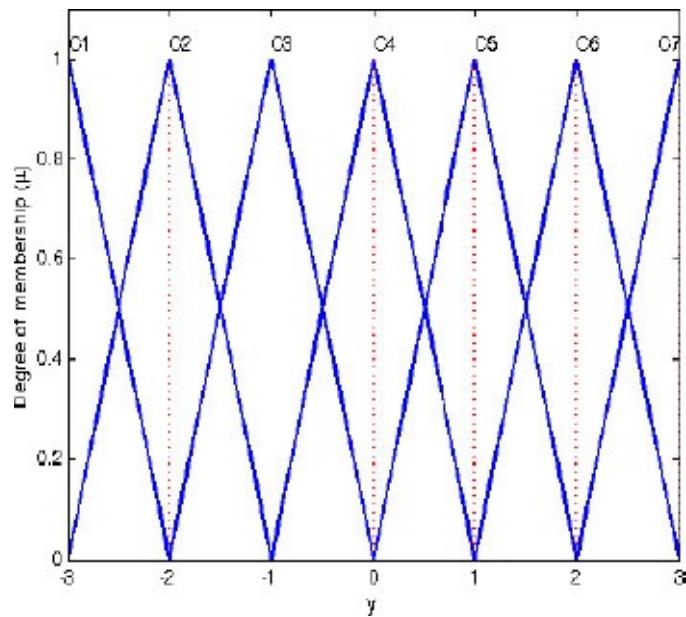


Figure 4.12: Output domain interval partitions and corresponding fuzzy sets.

if x_1 *is A1 and* x_2 *is B2 then* y *is C1.* (1)

The remaining rules are generated in the similar fashion and the preliminary set of rules is given as:

Rule 1: ***if*** x_1 ***is*** A1 ***and*** x_2 ***is*** B2 ***then*** y ***is*** C1.

Rule 2: ***if*** x_1 ***is A2 and*** x_2 ***is B2 then*** y ***is C1.***

Rule 3: ***if*** x_1 ***is*** A3 ***and*** x_2 ***is*** B1 ***then*** y ***is*** C2.

Rule 4: ***if*** x_1 ***is*** A3 ***and*** x_2 ***is*** B5 ***then*** y ***is*** C4.

Rule 5: **if** x_1 is A5 **and** x_2 is B7 **then** y is C4.

Rule 6: **if** x_1 is A5 **and** x_2 is B9 **then** y is C6.

Rule 7: **if** x_1 is A6 **and** x_2 is B10 **then** y is C7.

Rule 8: **if** x_1 is A5 **and** x_2 is B8 **then** y is C5.

Rule 9: **if** x_1 is A5 **and** x_2 is B7 **then** y is C5.

Rule 10: **if** x_1 is A4 **and** x_2 is B8 **then** y is C5.

For this problem, Rule 5 and Rule 9 are conflicting, therefore we determine the degree of Rule 5 and Rule 9. The degree of rule Rule 5 is 1 and that of Rule 9 is 0.6 [step (iii)]. Therefore, Rule 9 is removed and the final set of rules consists of Rule 1 to Rule 8, and Rule 10. Note that here the degree of all the data samples is considered to be 1, i.e., all are believed to be useful.

4.3.2. Neuro-Fuzzy Approach

A fuzzy system that employs neuro-fuzzy approach is commonly referred to as neuro-fuzzy system. Such a system is trained by a learning algorithm usually derived from neural network theory (Mitra and Hayashi, 2000). Neuro-fuzzy approaches are motivated by the fact that at the computational level, a fuzzy model can be seen as a layered structure (network), similar to artificial neural networks. A widely known neuro-fuzzy system is ANFIS (adaptive network-based fuzzy inference System) (Jang, 1993). ANFIS is an adaptive network consisting of nodes connected with directional links. It is called adaptive because some, or all, of the nodes have parameters which affect the output of the node and these parameters change to minimize the error. The ANFIS structure consists of TSK type of rules. However, the author suggests that it is possible to develop Mamdani system as well. ANFIS consists of nodes with variable parameters (square nodes) that represent membership functions of the antecedents, and membership functions of Mamdani-type consequent or the linear functions of the TSK-type consequent. The parameters of the nodes in the intermediate layers that connect the antecedents with the consequents are fixed (circular nodes). It is worth to note that in ANFIS the structure (nodes and layers) remains static, only the parameters of the nodes are adapted. This is the key distinction between ‘adaptive’ and ‘evolving’ fuzzy systems, in the latter both the structure and parameters are adapted (evolving fuzzy systems will be discussed in [Section 4.4](#)).

[Figure 4.13](#) shows the ANFIS structure for a rule-base consisting of following two TSK type rules.

Rule 1 : **if** x_1 is A₁ **and** x_2 is B₁ **then** $y_1 = a_{10} + a_{11}x_1 + a_{12}x_2$.

Rule 2 : **if** x_1 is A₂ **and** x_2 is B₂ **then** $y_2 = a_{20} + a_{21}x_1 + a_{23}x_2$.

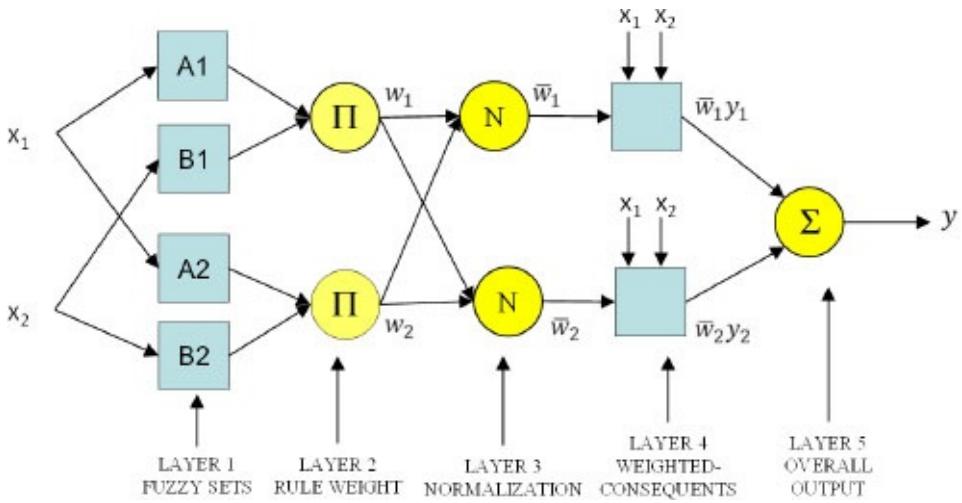


Figure 4.13: ANFIS structure.

In Layer 1, each node defines the fuzzy set in terms of bell-shaped membership function. The output from a node in this layer is the degree of membership of a given input given as

$$O_i^1 = \mu_{A_i}(x) = \exp \left\{ - \left[\left(\frac{x - c_i}{\sigma_i} \right)^2 \right]^{b_i} \right\}, \quad (5)$$

where $\{\sigma_i, b_i, c_i\}$ are antecedent parameter set. The bell shape function changes with the change in the values of these parameters.

Every node in Layer 2 determines the firing strength of a rule and the output of a node i can be given as

$$O_i^2 = w_i = \mu_{A_i}(x_1) \cdot \mu_{B_i}(x_2), \quad i = 1, 2. \quad (6)$$

In Layer 3, the nodes normalize the firing strength of a rule, and the output of a node i is given as

$$O_i^3 = \bar{w}_i = \frac{w_i}{w_1 + w_2}, \quad i = 1, 2. \quad (7)$$

In Layer 4, the nodes calculate the weighted output from each rule:

$$O_i^4 = \bar{w}_i y_i = \bar{w}_i (a_{i0} + a_{i1}x_1 + a_{i2}x_2), \quad (8)$$

where $\{a_{i0}, a_{i1}, a_{i2}\}$ is the consequent parameter set.

Finally, the single node in Layer 5 computes the overall output:

$$O_i^5 = \sum_i \bar{w}_i y_i. \quad (9)$$

For the given network in [Figure 4.13](#), the final output can be rewritten as

$$\begin{aligned}
y &= \frac{w_1}{w_1 + w_2} y_1 + \frac{w_2}{w_1 + w_2} y_2 \\
&= \bar{w}_1 y_1 + \bar{w}_2 y_2 \\
&= (\bar{w}_1) a_{10} + (\bar{w}_1 x_1) a_{11} + (\bar{w}_1 x_2) a_{12} + (\bar{w}_2) a_{20} \\
&\quad + (\bar{w}_2 x_1) a_{21} + (\bar{w}_2 x_2) a_{22}
\end{aligned} \tag{10}$$

which is a linear combination of consequent parameters.

Therefore, we have two sets of parameters, antecedent and consequent parameters. Initially, these parameters are set through partitioning of input data space. After the network parameters are set with their initial values, they are tuned using training data and hybrid learning algorithm. In the forward pass, the antecedent parameters are fixed, the functional signals go forward in the network till Layer 4, and finally the consequent parameters are determined using the method of least squares or recursive least squares. In the backward pass the consequent parameters are fixed, the error rates propagates back and the antecedent parameters are updated using gradient descent method. The combination of gradient descent and least squares method forms the hybrid learning algorithm.

Example 3.2. Consider the data in [Table 4.4](#) as training data. For this example, the input–output data given in [Table 4.4](#) are normalized as shown in [Table 4.5](#). Data normalization is performed as subtractive clustering (discussed later in [Section 4.3.4](#)) is used here to determine the initial set of antecedent parameters.

Let $[x_1^k, x_2^k, \dots, x_n^k, y^k]$ be the k th training example. Also, let r represent the number of rules and n represent the number of input variables.

Table 4.5: Training data (Example 3.2).

Input variable 1 (x_1)	Input variable 2 (x_2)	Output (y)
0	0.11	0
0.17	0.11	0
0.33	0	0.17
0.33	0.44	0.50
0.75	0.67	0.50
0.75	0.89	0.83
1.00	1.00	1.00
0.83	0.78	0.67
0.83	0.67	0.50
0.58	0.78	0.67

Table 4.6: Initial antecedent parameters.

		c	σ
Rule 1	A1	0.75	0.18
—	B1	0.67	0.18
Rule 2	A2	0.17	0.18
—	B2	0.11	0.18

Considering Gaussian membership functions, the initial set of antecedent parameters are given in [Table 4.6](#). These initial parameters can be obtained using any data partitioning method, for example, clustering. Let us assume that all the consequent parameters are initialized to 0.

To tune the antecedent and the consequent parameters we use the hybrid learning algorithm. We take the first data point $[x_1^1 = 0, x_2^1 = 0.11]$ as the input to the network shown in [Figure 4.13](#) and determine the output from Layer 1 to Layer 3.

Layer 1 output:

$$\begin{aligned} O_1^1 &= \mu_{A_1}(x_1^1) = e^{\frac{-(0-0.75)^2}{2*0.18^2}} = 0.0002, \\ O_2^1 &= \mu_{B_1}(x_2^1) = e^{\frac{-(0.11-0.67)^2}{2*0.18^2}} = 0.0079, \\ O_3^1 &= \mu_{A_2}(x_1^1) = e^{\frac{-(0-0.17)^2}{2*0.18^2}} = 0.6402, \\ O_4^1 &= \mu_{B_2}(x_2^1) = e^{\frac{-(0.11-0.11)^2}{2*0.18^2}} = 1. \end{aligned}$$

Layer 2 output:

$$\begin{aligned} O_1^2 &= w_1 = \mu_{A_1}(x_1^1) \cdot \mu_{B_1}(x_2^1) = 0.000004, \\ O_2^2 &= w_2 = \mu_{A_2}(x_1^1) \cdot \mu_{B_2}(x_2^1) = 0.00003. \end{aligned}$$

Layer 3 output:

$$\begin{aligned} O_1^3 &= \bar{w}_1 = \frac{w_1}{w_1 + w_2} = 0.000002, \\ O_2^3 &= \bar{w}_2 = \frac{w_2}{w_1 + w_2} = 1. \end{aligned}$$

Now, the consequent parameter can be determined using the Recursive Least Square (RLS) method. In the RLS method, the consequent parameters of r rules are represented in a matrix form $A = [a_{10}, a_{11}, \dots, a_{1n}, \dots, a_{r1}, \dots, a_{rn}]^T$ and the estimate of A at k^{th} iteration (based on k data points) is given as:

$$A_k = A_{k-1} + C_k X_{ek} (Y_k - X_{ek}^T A_{k-1}), \quad (11)$$

$$C_k = C_{k-1} - \frac{C_{k-1} X_{ek} X_{ek}^T C_{k-1}}{1 + X_{ek}^T C_{k-1} X_{ek}}, \quad (12)$$

where $X_{ek} = [\bar{w}_1, \bar{w}_1 x_1^k, \dots, \bar{w}_1 x_n^k, \bar{w}_2, \bar{w}_2 x_1^k, \dots, \bar{w}_2 x_n^k, \bar{w}_r, \bar{w}_r x_1^k, \dots, \bar{w}_r x_n^k]^T$, $Y_k = y^k$, C_k is $r(n +$

$1) \times r(n + 1)$ covariance matrix, and the initial conditions are $A_0 = 0$, $C_0 = \Omega I$, where Ω is a large positive integer.

For our example, with $k = 1$

$$X_{e1} = [\bar{w}_1, \bar{w}_1 * 0, \bar{w}_1 * 0.11, \bar{w}_2, \bar{w}_2 * 0, \bar{w}_2 * 0.11]^T \\ = [0.000002 \ 0 \ 0.00000022 \ 1 \ 0 \ 0.11]^T,$$

$$Y_k = 1.0$$

$$C_0 = \begin{bmatrix} 1000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1000 \end{bmatrix}.$$

$$C_1 = C_0 - \frac{C_0 X_{e1} X_{e1}^T C_0}{1 + X_{e1}^T C_0 X_{e1}} \\ = \begin{bmatrix} 1000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1000.0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 500.2 & 0 & -499.8 \\ 0 & 0 & 0 & 0 & 1000.0 & 0 \\ 0 & 0 & 0 & -499.8 & 0 & 500.2 \end{bmatrix},$$

$$A_0 = [000000]^T,$$

$$A_1 = A_0 + C_1 X_{e1} (Y_1 - X_{e1}^T A_0) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.4998 \\ 0 \\ 0.4998 \end{bmatrix}.$$

After $k = 1$ iteration of RLS method, we get the following consequent parameters:

$$a_{10} = 0, a_{11} = 0, a_{12} = 0, a_{20} = 0.4998, a_{21} = 0, a_{22} = 0.4998.$$

This process is continued for all the data points, i.e., for $k = 2, \dots, 10$. After getting the final set of consequent parameters (i.e., after $k = 10$), now the antecedent parameters are updated in the backward pass using gradient descent method. The aim of gradient descent is to minimize the error between the neural net's output and actual observed outputs.

The instantaneous error between the output y and the current reading y_k can be given as:

$$E_k = \frac{1}{2}(y - y_k)^2 = e, \quad k = 1, 2, \dots, 10. \quad (13)$$

By applying the chain rule, we obtain from Equation (13) the following equations for

updating the antecedent parameters:

$$\begin{aligned} c_{ij}(k) &= c_{ij}(k-1) - \alpha \frac{\partial E_k}{\partial c_{ij}}, \\ &= c_{ij}(k-1) - \alpha \bar{w}_i(a_{i0} + a_{i1}x_1 + \dots + a_{in}x_n - y)e \frac{x_j - c_{ij}(k-1)}{\sigma_{ij}^2(k-1)}, \end{aligned} \quad (14)$$

$$\begin{aligned} \sigma_{ij}(k) &= \sigma_{ij}(k-1) - \alpha \frac{\partial E_k}{\partial \sigma_{ij}} \\ &= \sigma_{ij}(k-1) - \alpha \bar{w}_i(a_{i0} + a_{i1}x_1 + \dots + a_{in}x_n - y)e \frac{(x_j - c_{ij}(k-1))^2}{\sigma_{ij}^3(k-1)}, \end{aligned} \quad (15)$$

where $i = 1, 2, \dots, r$ and $j = 1, 2, \dots, n$.

So, the antecedent parameters for every rule can be updated considering each of the data points at a time. One such cycle of update of antecedent and consequent parameters is referred to as epoch and many such epochs can be performed until the errors are within acceptable limits. Further, a separate set of input–output data (validation data or test data) can be used to validate the performance of the model by checking how closely it can predict the actual observed values.

4.3.3. Genetic-Fuzzy Approach

Fuzzy rule-based systems that involve GAs in the design process are commonly referred to as *genetic fuzzy rule-based systems* or *genetic fuzzy systems*, in general. When GA is used in the process of determining the membership functions with a fixed set of rules is often referred to as genetic tuning or optimizing parameters, and the process of determining rules is commonly known as genetic learning. During the genetic learning process, the rules can be determined at two levels. In the first level, the rules are determined with known membership functions, and in the second level both membership functions and fuzzy rules are determined using GA (Cordon *et al.*, 2004). In this section, first a brief overview of GAs is presented and then genetic tuning of membership functions and genetic learning of rules are described.

4.3.3.1. GAs

GAs are search algorithms that are inspired by the principles of natural evolution. A GA starts with a set of different possible solutions to the problem (population), then the performance of these solutions are evaluated based on a *fitness* or *evaluation function* (i.e., how good a solution is to the given problem). From these solutions only a fraction of good solutions is selected and the rest are eliminated (survival of the fittest). Finally, in the search for better solutions, the selected solutions undergo the process of reproduction, crossover, and mutation to create a new set of possible solutions (evolved population). This process of production of a new generation and its evaluation is repeated until

convergence is reached (Goldberg, 1989). This entire GA process is represented in [Figure 4.14](#).

There are primarily two representations of GAs, binary-coded GA (BCGA) and real-coded GA (RCGA). In BCGA, the possible solutions (or chromosomes or individuals in a population) are represented as strings of binary digits. In RCGAs, each chromosome is represented as a vector of floating-point numbers. Each gene represents a variable of the problem, and the size of the chromosome is kept the same as the length of the solution to the problem. Therefore, the RCGA can be considered to be directly operating on the optimization parameters whereas BCGA operates on an encoded (discretized) representation of these parameters.

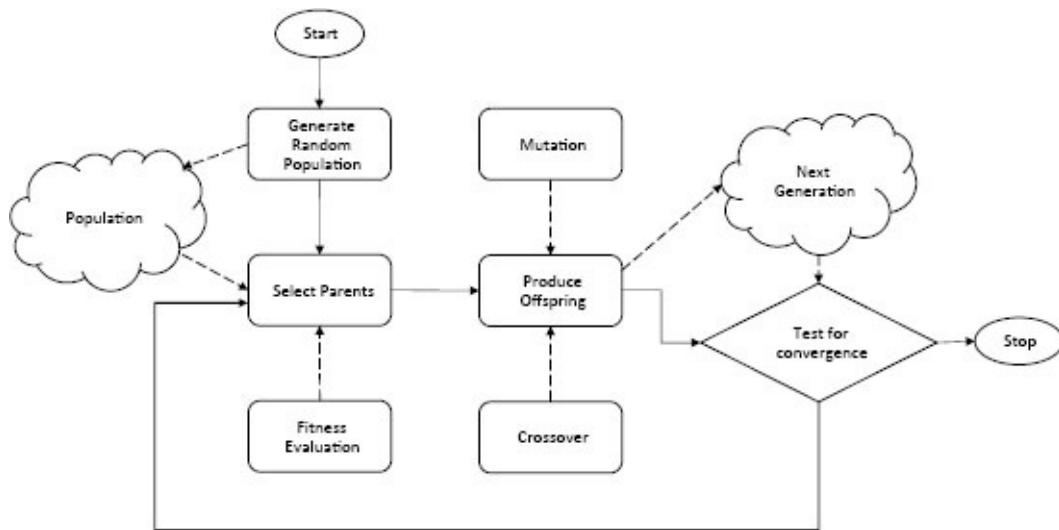


Figure 4.14: Diagrammatic representation of GA.

To apply GAs to solve a problem one needs to take into account the following issues:

- (i) Encoding scheme or genetic representation of solution to the problem: The choice of encoding scheme is one of the important design decisions. The bit string representation is the most commonly used encoding technique.
- (ii) Fitness evaluation function: Choosing and formulating an appropriate evaluation function is crucial to the efficient solution of any given genetic algorithm problem. One approach is to define a function that determines the error between the actual output (from training data) and the output returned by the model.
- (iii) Genetic operators: The genetic operators are used to create the next generation individuals by altering the genetic composition of an individual (from previous generation) during reproduction. The fundamental genetic operators are: selection, crossover, and mutation.
- (iv) Selection of input parameters: GAs requires some user-defined input parameters, for example, selection probability, mutation and crossover probability, and population size.

Selection: The aim of the selection operator is to allow better individuals (those that are

close to the solution) to pass on their genes to the next generation. In other words, they are the fittest individuals in the population and so are given the chance to become parents or to reproduce. The commonly known selection mechanisms are: proportionate selection method (e.g., Roulette wheel selection), ranking selection (e.g., linear ranking selection), and tournament selection (e.g., binary tournament selection).

Crossover: The purpose of the crossover operator is to produce new chromosomes that are distinctly different from their parents, yet retain some of their parent characteristics. It combines the features of two parent chromosomes to form two offspring with the possibility that the offspring generated through recombination are better adapted than their parents. A random choice is made, where the likelihood of crossover being applied depends on probability defined by a crossover rate, the crossover probability. Definitions for this operator are highly dependent on the particular representation chosen. Two widely known techniques for binary encoding are: one-point crossover and two-point crossover. In one-point crossover, two parent chromosomes are interchanged at a randomly selected point thus creating two children (Figure 4.15). In two-point crossover, two points are selected instead of just one crossover point. (Figure 4.16).

Mutation: The structure of some of the individuals in the new generation produced by selection and crossover is modified further by using the mutation operator. The most common form of mutation is to alter bits from a chromosome with some predetermined probability (mutation probability) (Figure 4.17). Generally, in BCGA the mutation probability is set to very low value.

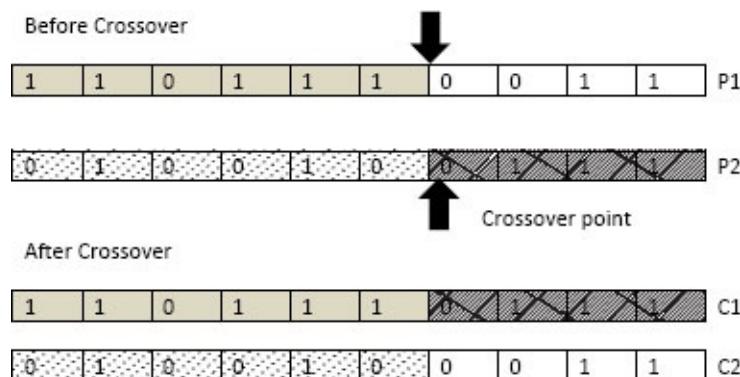


Figure 4.15: One-point crossover.

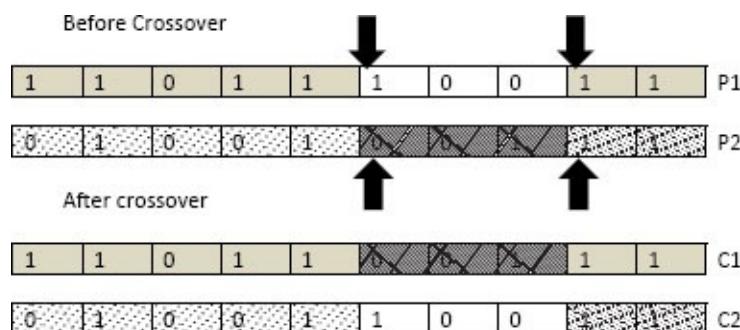


Figure 4.16: Two-point crossover.

Before Mutation										
1	1	0	1	1	1	0	0	1	1	
After Mutation										
1	1	0	1	1	1	1	0	1	1	

Figure 4.17: Mutation.

There is an additional policy, called elitist policy, that can be applied after crossover and mutation to retain some number of the best individuals at each generation. This is required because in the process of crossover and mutation the fittest individual may disappear.

4.3.3.2. Genetic tuning

In case of genetic tuning, the rules, the number of membership functions (fuzzy sets), and the type of each membership function are assumed to be available. The most common membership functions for which GAs are used to tune the parameters are triangular, trapezoidal, or Gaussian functions. Depending on the type of membership function the number of parameters per membership function ranges from one to four and each parameter is binary or real coded. In the following example, we demonstrate the genetic tuning of membership function parameter, given the fuzzy rules of the system.

Example 3.3. Consider a system with single input (x) and single output (y) with input–output values as shown in [Table 4.7](#) and the rules are represented in [Table 4.8](#) (For simplicity, we are not using the input–output data of the cooling fan system ([Table 4.4](#)), with two input and single output).

The range of x is $[0, 10]$ and the range of y is $[0, 100]$. The shape of both input and output membership functions is assumed to be triangular.

[Figure 4.18](#) shows the five parameters (P_1, P_2, \dots, P_5) that are required to be optimized by the GA.

Based on the range of input and output variables, and the required precision, first the length of the bit string required to encode each parameter is determined. The range of input is 10 and let us assume that the required precision is one digit after the decimal point, so each of the input parameter will require 7 bits ($\log_2 (10 \times 10)$). Similarly, 10 bits will be used to encode each of the output parameters. So an individual or a chromosome will have a total of 41 bits (3×7 bits for P_1, P_2, P_3 plus 2×10 bits for P_4 and P_5).

Table 4.7: Input–output data (Example 3.3).

Input (x)	1	2	3	4	5	6	7	8	9	10
Output (y)	2	5	10	17	26	37	50	65	80	100

Table 4.8: Rules (Example 3.3).

x	y
-----	-----

Low		Small
Medium		Big
High		Big

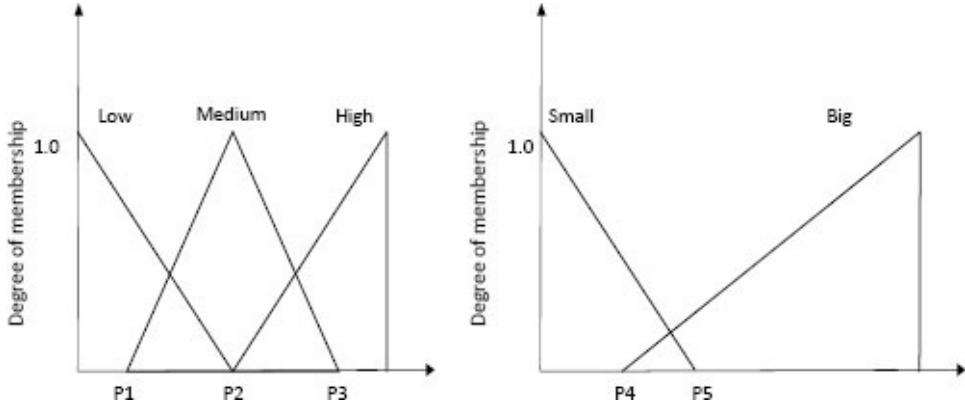


Figure 4.18: Membership functions and parameters for input and output variables.

Next, we need a mapping function to map the bit string to the value of a parameter. This can be done using the following mapping function (Goldberg, 1989):

$$d(UB, LB, L, bs) = LB + \frac{B}{2^L - 1}(UB - LB), \quad (16)$$

where bs is the bit string to be mapped to a parameter value, B is the number in decimal form that is being represented in binary form by the given bit string, UB is upper bound, LB is lower bound, L is the length of the bit string.

Finally, we need a fitness function to evaluate the chromosomes. The fitness function for this problem is given as:

$$f(I) = \frac{1}{1 + \text{RMSE}}, \quad (17)$$

where I is the individual for which the fitness is evaluated and RMSE is the root mean squared error.

[Table 4.9](#) shows the first iteration of genetic algorithm with an initial population size of four. The bit strings are first decoded into binary values (column 2) and then mapped to decimal values (column 3) using Equation (16). For parameters P_1, P_2, P_3 the value of LB is set to 0, UB is 10 and L is 7 and for parameters P_4, P_5 , the value of LB is 0, UB 100, and L is 10. [Figure 4.19](#) shows the membership functions of the system with values $P_1 = 4.0, P_2 = 7.3, P_3 = 8.7$, and $P_5 = 57.2$ (Individual 2 from [Table 4.9](#)).

After the decimal values of the string are determined, the estimated output is computed using the parameter values represented by each individual. To evaluate the fitness of each individual the RMSE is calculated as shown in [Table 4.10](#). Using the RMSE values from [Table 4.10](#) and Equation (17), the fitness of each individual is determined. From the initial population and based on the fitness values, some parents are selected for reproduction or to generate a new set of solutions. In this example, *roulette*

wheel method is used for parent selection. In roulette wheel method first the fitness values are normalized so that the range of fitness values becomes [0, 1]. Then the normalized values are sorted in descending order. A random number is generated between 0 and 1. The first individual is selected whose fitness added to the preceding individual is greater than or equal to the generated random number. As shown in [Table 4.11](#), the individual 2 is selected two times out of four times and individual 3 and 4 are selected once, and individual 1 did not get a chance to reproduce as it has the lowest fitness value. After the parents are selected the next generation of population is generated by applying crossover and mutation as shown in columns (1) and (2) of [Table 4.12](#). In this example, the crossover is applied to all the individuals (crossover probability is 1.0), and mutation is applied to two randomly selected individuals, where 1 out of 41 bit is flipped. The locations of crossover and mutation are indicated in column (1) and the individuals generated after these operations are shown in column (2) of [Table 4.12](#). These bit strings also undergo the same process of decoding and fitness evaluation as shown in [Table 4.12](#) (columns 3 and 4) and [Table 4.13](#). The final fitness values for the individuals in this new generation are indicated in [Table 4.14](#). It is clear from [Tables 4.11](#) and [4.14](#) that in the new generation none of the individuals have fitness value lower than 0.4, whereas the fitness value in the previous generation was 0.3. Also, if we compare the two best strings from the first generation and the second generation, then the former string has lower fitness value than the latter.

Table 4.9: First iteration of genetic algorithm.

String number (Individual)	(1) Bit string	(2) Decimal value					(3) Mapped value				
		P1	P2	P3	P4	P5	P1	P2	P3	P4	P5
1	0101100 1101010 1110000	44	106	112	9	657	3.5	8.3	8.8	0.9	64.2
	0000001001 1010010001										
2	0110011 1011101 1101111	51	93	111	451	585	4.0	7.3	8.7	44.1	57.2
	0111000011 1001001001										
3	1001011 1001101 1011100	75	77	92	540	819	5.9	6.1	7.2	52.8	80.1
	1000011100 1100110011										
4	0101000 1001110 1100101	40	78	101	453	910	3.1	6.1	8.0	44.3	89
	0111000101 1110001110										

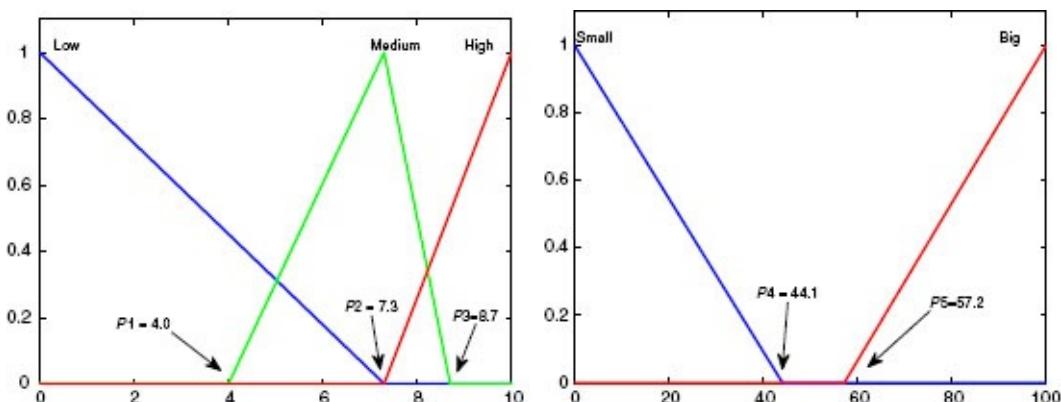


Figure 4.19: Input and output membership functions.

This process of generation and evaluation of the strings continues till convergence to the solution is arrived.

4.3.3.3. Genetic learning of rules

Many researchers have investigated the automatic generation of fuzzy rules using GAs. Their work in this direction can be broadly grouped into following categories:

- Genetic learning of rules with fixed membership functions.
- Learning both fuzzy rules and membership functions but serially, for example, first good membership functions are determined and then they are used to determine the set of rules.
- Simultaneous learning of both fuzzy membership functions and rules.

While learning rules, GAs can be applied to obtain a suitable rule-base using chromosomes that code single rules or a complete rule-base. On the basis of chromosome as a single or complete rule-base, there are three widely known approaches in which GAs have been applied, the Michigan (Holland and Reitman, 1978), the Pittsburgh (Smith, 1980), and the Iterative Rule Learning (IRL) (Venturini, 1993) approaches. In Michigan approach, the chromosome correspond to rules and a rule set is represented by entire population with genetic operators applied at the level of rules, whereas in the Pittsburgh approach, each chromosome encodes a complete set of rules. In the IRL approach, each chromosome represents only one rule, but contrary to the first approach, only the best individual is considered as the solution, discarding the remaining chromosomes in the population.

Table 4.10: RMSE values.

Actual output	Individual 1		Individual 2		Individual 3		Individual 4	
	Estimated output	Squared-error						
2	0	4	14.6	158.76	17.7	246.49	14.8	163.84
5	0	25	15.3	106.09	18.8	190.44	15.7	114.49
10	0	100	16.2	38.44	20.2	104.04	16.9	47.61
17	72.8	3113.64	17.3	0.09	21.9	24.01	32.8	249.64
26	81.6	3091.36	49.2	538.24	23.8	4.84	51.1	630.01
37	84.8	2284.84	66.2	852.64	85.8	2381.44	88.5	2652.25
50	86.9	1361.61	81	961	91.4	1713.96	96	2116
65	88.2	538.24	83.6	345.96	92.5	756.25	96	961
80	85.7	32.49	84.6	21.16	93.3	176.89	96.4	268.96
100	88.4	134.56	86.1	193.21	93.7	39.69	96.7	10.89
MSE	1068.57		321.56		568.8		721.47	
RMSE	32.69		17.93		23.84		26.86	

Table 4.11: Fitness values for initial population.

String number (Individual)	Bit string	Fitness (f)	Normalized fitness		Selection
			(f/f_{tot})	Sum = 1.00	
1	0101100 1101010 1110000 0000001001 1010010001	0.03	0.19	0	
2	0110011 1011101 1101111 0111000011 1001001001	0.05	0.31	2	
3	1001011 1001101 1011100 1000011100 1100110011	0.04	0.25	1	
4	0101000 1001110 1100101 0111000101 1110001110	0.04	0.25	1	
		Sum (f_{tot}) = 0.16		Sum = 1.00	

Table 4.12: Crossover and mutation operation.

String number (Old individual)	String number (New individual)	(1) Selected individuals (Bit string)	(2) New individuals (Bit string)	(3) Decimal value					(4) Mapped value				
				P1	P2	P3	P4	P5	P1	P2	P3	P4	P5
2	1	0110011 101 1101 1101111 0110011 1011101 1 <u>1</u> 11100 51 93 124 540 819 4.0 7.3 9.7 52.8 80.1	0111000011 1001001001 1000011100 1100110011										
3	2	1001011 100 1101 10 <u>1</u> 1100 1001011 1001101 1101111 75 77 111 451 585 5.9 6.1 8.7 44.1 57.2	1000011100 1100110011 0111000011 1001001001										
2	3	0110011 101 1101 1101111 0110011 1011110 11 <u>1</u> 0101 51 94 117 453 910 4.0 7.4 9.2 44.3 89	0111000011 1001001001 0111000101 1110001110										
4	4	0101000 100 1110 1100101 0101000 1001101 1101111 40 77 111 451 585 3.1 6.1 8.7 44.1 57.2	0111000101 1110001110 0111000011 1001001001										

Table 4.13: RMSE values for individuals of second generation.

Actual output	Individual 1		Individual 2		Individual 3		Individual 4	
	Estimated output	Squared-error						
2	17.6	243.36	14.5	156.25	14.7	161.29	14.7	161.29
5	18.4	179.56	15.3	106.09	15.3	106.09	15.6	112.36
10	19.4	88.36	16.5	42.25	16.2	38.44	16.8	46.24
17	20.8	14.44	17.9	0.81	17.3	0.09	47.8	948.64
26	41.1	228.01	19.5	42.25	33.3	53.29	66.5	1640.25
37	58.5	462.25	85.5	2352.25	49.4	153.76	84	2209
50	81.9	1017.61	96.3	2143.69	75	625	84.7	1204.09
65	93.2	795.24	96	961	96.3	979.69	83.5	342.25
80	93	169	96.4	268.96	96.2	262.44	85.3	28.09
100	93.7	39.69	96.7	10.89	96.7	10.89	86.1	193.21
MSE	323.75		608.44		239.09		688.54	
RMSE	17.99		24.66		15.46		26.24	

Table 4.14: Fitness values for individuals of second generation.

String number (Individual)	Bit string	Fitness (f)
1	0110011 1011101 1111100 1000011100 1100110011	0.05
2	1001011 1001101 1101111 0111000011 1001001001	0.04
3	0110011 1011110 1110101 0111000101 1110001110	0.06
4	0101000 1001101 1101111 0111000011 1001001001	0.04

Thrift (1991) described a method for genetic learning of rules with fixed membership functions based on encoding of complete set of rules (Pittsburg approach). Using genetic algorithm a two-input, one-output fuzzy controller was designed for centering a cart on a frictionless one-dimensional track. Considering triangular membership functions, for each input variable and output variable the fuzzy sets, Negative-Medium (NM), Negative-Small (NS), Zero (ZE), Positive-Small (PS) and Positive-Medium (PM), were defined. The

control logic was presented in the form of a 5×5 decision table with each entry encoding an output fuzzy set taken from {NM, NS, ZE, PS, PM, _} where the symbol “_” indicated absence of a fuzzy set. A chromosome is formed from the decision table by going row-wise and producing a string of numbers from the given code set {0, 1, 2, 3, 4, 5} corresponding to {NM, NS, ZE, PS, PM, _} respectively. For example, consider the following decision table ([Table 4.15](#)), the corresponding chromosome for the given table (rule set) is: ‘0321041425023413205110134’.

Thrift’s method of designing the control employed elitist selection scheme with standard two-point crossover operator. The GA mutation operator changes a code from the given code set either up or down a level, or to a blank code. After a simulation of 100 generations and using a population size of 31, Thrift’s system was able to evolve a good fuzzy control strategy.

Kinzel *et al.* (1994) described an evolutionary approach for learning both rules and membership functions in three stages. The design process involves the following three phases: (i) determine a good initial rule-base and fuzzy sets, (ii) apply GA to the rules by keeping the membership functions fixed, (iii) tune the fuzzy sets using GA to get optimal performance. In this approach also, the rule-base is represented in the form of a table and each chromosome encodes such a table. Firstly, the population is generated by applying the mutation operator on all genes of the initial rule-base. [Figure 4.20](#) shows mutation of rule-base considering cart-pole problem. During Mutation one fuzzy set (gene) is replaced by randomly chosen but similar fuzzy set. For example, the fuzzy set ‘ZE’ could be mutated to ‘NM’ or ‘PM’. After calculating the fitness of the population, the genetic operations selection, crossover and mutation are used to generate the next population. This process is continued until a good rule-base is found.

[Table 4.15](#): Example decision table considering five membership functions.

	NM	NS	ZE	PS	PM
NM	0	3	2	1	0
NS	4	1	4	2	5
ZE	0	2	3	4	1
PS	3	2	0	5	1
PM	1	0	1	3	4

(before Mutation)
(after Mutation)

NM	NB	ZE	NM	PM
NB	NM	NM	ZE	PM
NM	NM	ZE	PM	PM
NB	NM	ZE	PM	PB
PB	PM	PB	PB	PM

(ZE replace by PM)

NM	NB	PM	NM	PM
NB	NM	NM	ZE	PM
NM	NM	ZE	PM	PM
NB	NM	ZE	PM	PB
PB	PM	PB	PB	PM

[Figure 4.20](#): Mutation of rule-base.

After a rule-base is found, in the next stage the fuzzy sets are tuned. The authors argue against the use of bit string encoded genomes, due to the destructive action of crossover. The fuzzy sets are encoded by representing each domain by a string of genes. Each gene represents the membership values of the fuzzy sets of domain d at a certain x -value. Thus,

a fuzzy partition is described by discrete membership values. The standard two-point crossover operator is used and the mutation is done by randomly choosing a membership value $\mu_i(x)$ in a chromosome and changing it to a value in [0, 1]. For the cart-pole problem, Kinzel *et al.*'s (1994) method discovers good fuzzy rules after 33 generations using a population size of 200.

The learning method described by Liska and Melsheimer (1994) simultaneously learns the rules and membership function parameters. The design process tries to optimize, the number of rules, the structure of rules, and the membership function parameters simultaneously by using RCGA with one-at-time reproduction. In this type of GA, two offspring are produced by selecting and combining two parents. One of the offspring is randomly discarded, the other replaces the poorest performance string in the population. During each reproduction step only one operator is employed.

For simultaneous optimization, the chromosome is composed of three substrings: the first substring of real numbers encodes membership functions of input and output variables. Each membership function is represented by the two parameters (center and width). The second substring of integer numbers encodes the structure of each rule in the rule-base such that one integer number represents one membership in the space of an input variable. The membership functions are numbered in ascending order according to their centers. For example, a number “1” refers to the MF with the lowest value of the MF center in a particular input variable. The value “0” in the second substring indicates the “null” MF, i.e., the input variable is not involved in the rule. The third substring of integer numbers encodes MFs in rule consequents. A value “0” in the third substring means that the rule is deleted from the FLS rule-base. For example, in a system with n input variables and one output variable, p_i membership functions in i th input variable, q membership functions in the output variable, and N rules, the three substrings can be represented as shown in [Figure 4.21](#).

The inclusion of “0” in the second substring allows the number of input variables involved in each rule to change dynamically during the GA search. Similarly, ‘0’ in the third substring allows the number of rules to vary dynamically. The number of rules in FLS rule-base is constrained by the upper limit specified by a designer. The evolution process used a set of ordered genetic operators such that the relative order of MFs in each variable is preserved. The ordered operators are used only for the first substring, for the second and third substring ordinary genetic operators are used, viz., uniform crossover, mutation, and creep. Uniform crossover creates two offspring from two parents by deciding randomly which offspring receives the gene from which parent. Mutation replaces randomly selected gene in a parent by a random value between the minimum and maximum allowed values. Creep creates one offspring from one parent by altering randomly its gene within a specified range.

x_i	Width	Center	Width	Center	...	Width	Center
	0.15	0.20	0.19	0.50	...	0.22	0.75
	MF ₁		MF ₂			MF _{p_i}	
	Width	Center	Width	Center	...	Width	Center
$y(x)$	0.20	0.10	0.32	0.45	...	0.38	0.52
	MF ₁		MF ₂			MF _q	

(a) Membership function substring for all input and output variables.

$rule_k$	x_1	x_2	...	x_n
	1	3	...	0

(b) Rule-base substring

$y(x)$	$rule_1$	$rule_2$...	$rule_N$
	1	0	...	2

(c) Rule consequents substring

Figure 4.21: The three substrings of a chromosome.

An exponential ranking technique based on error function is used to evaluate the performance of each chromosome. This technique assigns the highest fitness to the string with the lowest value of the error function (e.g., $fitness(1) = 1000$). If $fitness(i)$ is the fitness value for the i th lowest value of the error then the fitness value of the next lowest value is set to $fitness(i + 1) = \alpha * fitness(i)$, $\alpha \in [0, 1]$, except that no string is given a fitness lesser than 1. Liska and Melsheimer (1994) obtained the best results with α value set to 0.96. In their GA implementation, no duplicates were allowed, i.e., a new offspring is allowed to be the member of the current population if it differs from every existing member at least in one gene.

Liska and Melsheimer (1994) applied their genetic learning approach to learn a dynamic model of a plant using input–output data. After the genetic learning process, they further applied another technique to fine tune the membership function parameters. The obtained results are comparable to those achieved using a three-layer feed-forward neural network.

4.3.4. Clustering-Based Approach

The aim of clustering is to partition a given dataset into different groups (clusters) so that the members in the same group are of similar nature, whereas members of different groups are dissimilar. While clustering, various similarity measures can be considered, one of the most commonly used measure is distance between data samples. Clustering can be either *hard* (or *crisp*) clustering technique, e.g., k -means (Hastie *et al.*, 2009), where a data sample is assigned only to one cluster or *fuzzy* clustering where a data sample can belong to all the clusters with certain degree of membership (de Oliveira and Pedrycz, 2007).

In the domain of fuzzy system design, a clustering algorithm is applied to structure identification by partitioning the input–output data in to clusters. Each cluster corresponds to a rule in the rule-base. The cluster centers can be considered as focal points for rules in the rule-base. Different clustering methods can be used for the purpose of data partitioning and rule generation, however, fuzzy clustering is being used extensively either independently or combined with other techniques. Methods based on fuzzy clustering are appealing as there is a close connection between fuzzy clusters and fuzzy rules (Klawonn,

1994; Kruse *et al.*, 1994). Some of the clustering algorithms that are commonly used for structure identification are, fuzzy c -means (Dunn, 1974; Bezdek, 1981), Gustafsson–Kessel algorithm (Gustafsson and Kessel, 1979), mountain clustering (Yager and Filev, 1993, 1994), and subtractive clustering (Chiu, 1997).

To get the fuzzy rules, clustering can be applied separately to input and/or output data or jointly to the input–output data. Sugeno and Yasukawa (1993) and Emami *et al.* (1998) used fuzzy clustering to cluster the output data and then these clusters are projected on to the input coordinate axes in order to generate linguistic fuzzy rules. For fuzzy system with TSK type rules, the common approach is to apply clustering in the input–output data and projecting the clusters on to the input variables coordinate to determine the premise part of the rule in terms of input fuzzy sets (Babuška and Verbruggen, 1996; Zhao *et al.*, 1994; Chiu, 1994) (Figure 4.22). Each of the clusters give rise to local regression models and the overall model is then structured into a set of *if-then* rules. The consequent parameters of such rules may be estimated separately by using methods like least squares method. Some authors have used clustering only on the input data and combined the results with a TSK-like consequent (Wang and Langari, 1994). While others have applied clustering separately to each input and output data for fuzzy modeling in terms of fuzzy relational equations (Pedrycz, 1984).

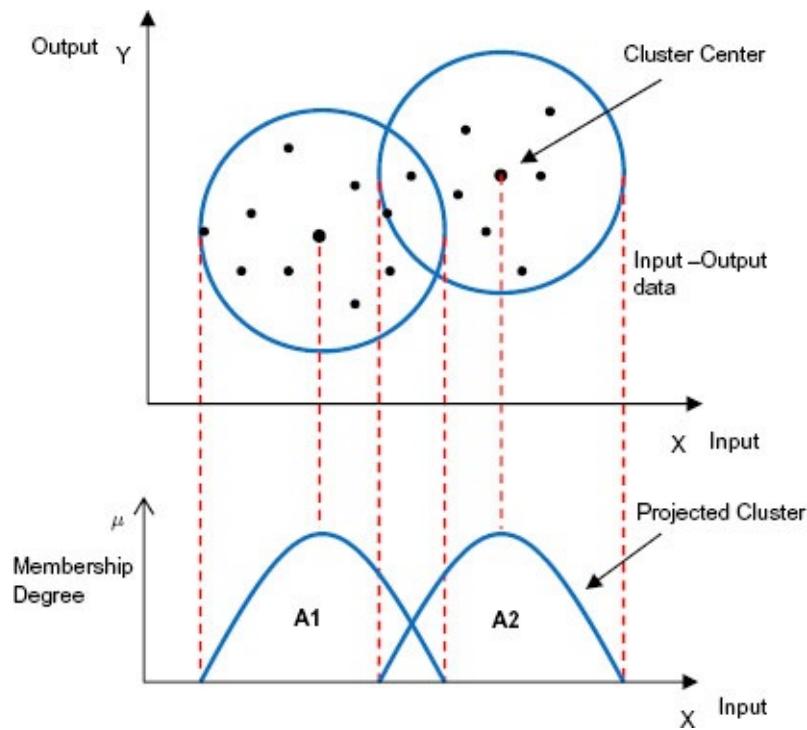


Figure 4.22: Fuzzy clusters and their interpretation as membership functions.

In the following example, we demonstrate the generation of rules using subtractive clustering method. Subtractive clustering is an improved version of mountain method for cluster estimation. One of the advantages of mountain method and subtractive clustering method over fuzzy c -means is that the former methods do not require specification of number of clusters (or number rules) by the user before the clustering process begins. In subtractive clustering data points are considered as potential clusters. The method assumes

normalized data points bounded by a hypercube. For every data point, a *potential* value is calculated as given in Equation (18), and the point with the highest potential value is selected as the first cluster center.

$$P(\mathbf{x}_k) = \sum_{i=1}^N e^{-\alpha \|\mathbf{x}_k - \mathbf{x}_i\|^2}, \quad (18)$$

where \mathbf{x}_k is the k^{th} data point, $k = 1, 2, 3, \dots, N$, and $\alpha = \frac{4}{r_a^2}$, r_a is a positive constant.

The potential value is dependent on the distance of the data point to all other data points, i.e., the larger the number of neighboring data points the higher is the potential. The constant r_a defines the neighborhood of a data point and the data points outside the neighborhood do not have significant influence on the potential value. After the first cluster center is identified, in the next step the potential of all data points is reduced by an amount that is dependent on their distance to the cluster center. The revised potential of each data point is given by Equation (19). So, the points closer to the cluster center have less chance to be selected as next cluster center. Now, the next cluster center is the point with the remaining maximum potential.

$$P(\mathbf{x}_k) = P(\mathbf{x}_k) - P_1^* e^{-\beta \|\mathbf{x}_k - \mathbf{x}_1^*\|^2}, \quad (19)$$

where \mathbf{x}_1^* is the first cluster center and P_1^* is its potential value, and $\beta = \frac{4}{r_b^2}$, r_b is a positive constant.

The constant r_b is the radius defining the neighborhood that will have measurable reductions in potential. To obtain cluster centers that are not too close to each other, r_b is set to a value greater than r_a .

The process of selection of cluster centers based on the potential values and subsequent reduction of potential values of each of the data points continues till the following conditions are satisfied:

if $P_k^* > u P_1^*$ **then** accept \mathbf{x}_k^* as a cluster center and continue,

else if $P_k^* < v P_1^*$ **then** reject \mathbf{x}_k^* and end the clustering process,

else let d_{\min} be the minimum distance between \mathbf{x}_k^* and all existing cluster centers.

if $\frac{d_{\min}}{r_a} + \frac{P_k^*}{P_1^*} \geq 1$ **then** accept \mathbf{x}_k^* as a cluster center and continue,

else reject \mathbf{x}_k^* and set the potential at \mathbf{x}_k^* to 0,

select the data point with the next highest potential as the new \mathbf{x}_k^*

and retest.

end if

end if

end if

Here, u is a threshold potential value above which the data point is definitely accepted as a cluster center, and a data point is rejected if the potential is lower than the threshold v . If the potential falls within u and v , then it is checked if the data point provides a good trade-off between having a sufficient potential and is not close to any existing cluster centers.

Example 4.1. Let us consider the input–output data from the cooling fan system provided in [Table 4.4](#). Subtractive clustering assumes normalized data for clustering, which is given in [Table 4.5](#). We assume Gaussian membership functions and TSK-type fuzzy rules.

The antecedent parameters of Gaussian membership function (c, σ) are identified by subtractive clustering and the consequent parameters are determined using RLS approach. First, subtractive clustering is applied to input–output data with the value of radius parameter set to $r_a = 0.5$. The rest of the user-defined input parameters are set to their default values, $r_b = 1.5r_a$, $u = 0.5$, and $v = 0.15$ (Chiu, 1997). The σ value is determined as, $\sigma^2 = 1/(2\alpha)$. The clustering resulted in six clusters with cluster centers and the range of influence as shown in [Table 4.16](#). [Figure 4.23](#) shows the cluster centers and the range of influence considering the two input variables. The number of clusters corresponds to number of fuzzy rules. As we have initialized the neighborhood parameter with same values for each of the data dimension so subtractive clustering returns same sigma values (radius) in each of the data dimensions. In [Table 4.16](#), the sigma value is represented with a scalar. Also note that it returns same sigma values for each of the clusters. While considering the antecedent parameters we neglect the cluster output dimension, i.e., only the values in the columns 1, 2, and 4 of [Table 4.16](#) are considered as c and σ respectively. The input fuzzy sets based on the cluster centers and respective range of influences are shown in [Figure 4.24](#).

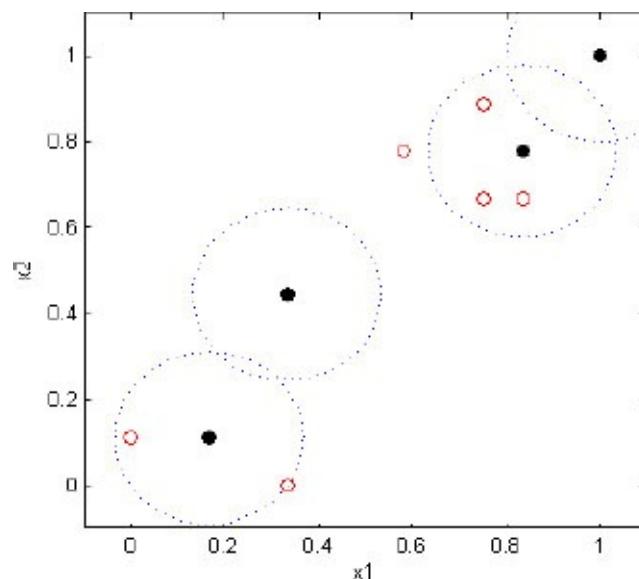


Figure 4.23: Cluster centers obtained by subtractive clustering (black dots indicate cluster centers).

Table 4.16: Cluster centers and corresponding radii.

<i>x</i> ₁	<i>x</i> ₂	<i>y</i>	<i>radius</i>
0.83	0.78	0.67	0.18
0.17	0.11	0	0.18
0.33	0.44	0.5	0.18
1.0	1.0	1.0	0.18

In the next step, the consequent parameters are determined using the least square estimate method. The resulting parameters are shown in [Table 4.17](#).

Therefore, the rules generated using the subtractive clustering method can be given as:

Rule 1 : if *x*₁ is A₁ and *x*₂ is B₁ then $y_1 = -3.39 + 5.37x_1 - 0.35x_2$.

Rule 2 : if *x*₁ is A₂ and *x*₂ is B₂ then $y_2 = 0.06 + 2.39x_1 + 1.93x_2$.

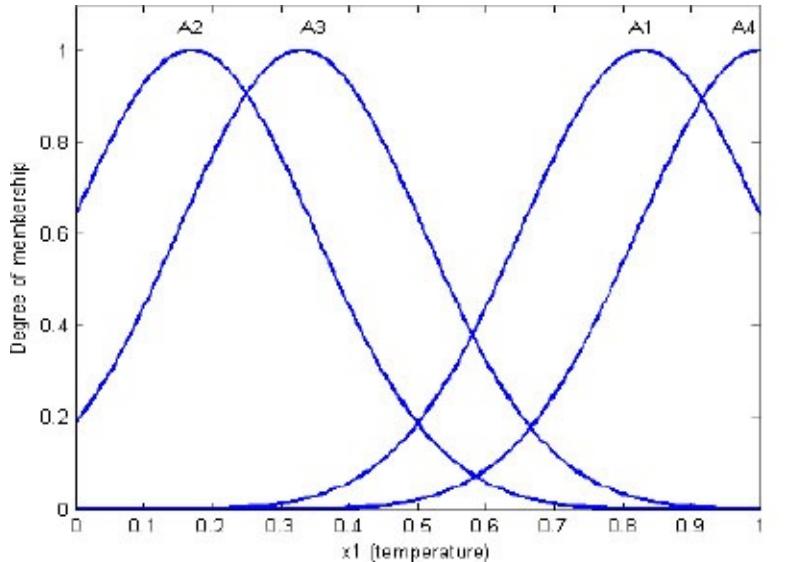
Rule 3 : if *x*₁ is A₃ and *x*₂ is B₃ then $y_3 = -8.45 + 19.85x_2$.

Rule 4 : if *x*₁ is A₄ and *x*₂ is B₄ then $y_4 = -9.25x_1 + 10.07x_2$.

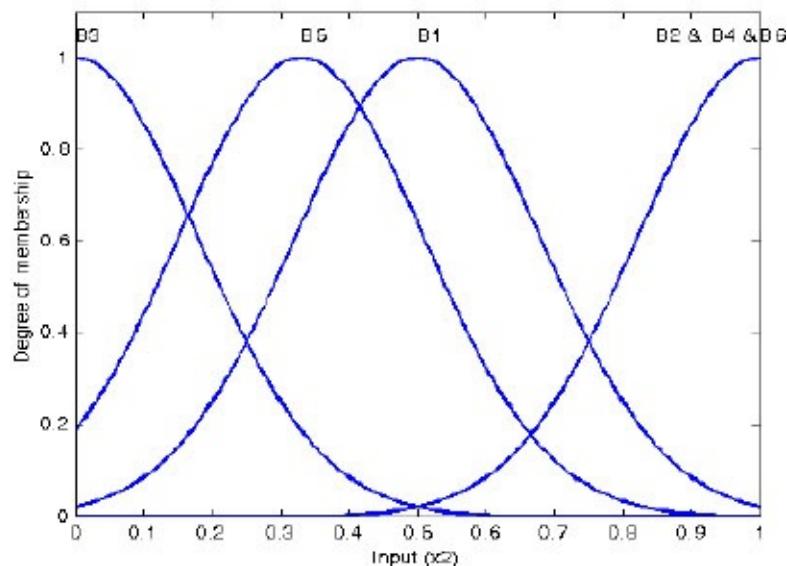
It should be noted that the antecedent parameter obtained using clustering can further be tuned using neural networks or GAs as discussed in previous sections.

4.4. Online Approaches

Online approaches can also be categorized under automated modeling approaches, however the FRB systems that are developed using these approaches have some interesting characteristics as discussed here. So far we have discussed the modeling of fuzzy systems that have fixed structure. Though we have mentioned the design of adaptive systems like ANFIS, such systems are adaptive in terms of parameters and not the structure. By fixed structure, we mean the number of rules in the rule-base and the number of fuzzy membership functions are fixed during the design process. The clustering methods like mountain clustering and subtractive clustering when applied to the design process do not require to specify the number of clusters (i.e., number of rules) beforehand. They assume that the entire data is present in the memory and iteratively delineates the clusters. The manner in which the rules are learnt is often referred to as *offline* or *batch* mode where the learning algorithm performs multiple iterations over the data to finally determine a fixed set of rules. However, due to the static nature of the rule-base, the resulting system cannot handle any deviations in the input data which may be due to changes in the operating environment over time. Such changes cannot be incorporated in the rule-base or existing model unless the entire design process is repeated with the new data and the whole system is re-modeled.



(a) Input fuzzy sets (x_1)



(b) Input fuzzy sets (x_2)

Figure 4.24: Input fuzzy sets formed from the cluster centers and radii obtained from subtractive clustering.

Table 4.17: Consequent parameters.

a_0	a_1	a_2
-3.39	5.37	-0.35
0.06	2.39	1.93
-8.45	0	19.85
0	-9.25	10.07

In the present scenario, the overabundance of data due to the technological advancement poses new challenges for the fuzzy system design. The application such as packet monitoring in the IP network, monitoring chemical process, real time surveillance systems and sensor networks, etc. generate data continuously at high speed that often evolve with time, commonly referred to as *data streams*, inhibit the application of conventional approaches to fuzzy system design. Learning rules from such type of data require the method to be fast and memory efficient. To attain real-time or online response,

the processing time per data sample should be a small constant amount of time to keep up with their speed of arrival, and the memory requirements should not increase appreciably with the progress of the data stream. Another requirement for a data stream learning approach is to be adaptive and robust to noise. The approach should be able to adapt the model structure and parameters in the presence of deviation so as to give an up-to-date model. In a streaming environment, it is difficult to distinguish noise from data shift. Noisy data can interfere with the learning process, for example, a greedy learning approach that adapts itself as soon as it sees a change in the data pattern may over-fit noise by mistakenly interpreting it as new data. On the other hand, if it is too conservative and slow to adapt, it may fail to incorporate important changes.

To meet such requirements, the area of *evolving fuzzy systems* (EFSs) emerged that focuses on *online* learning of fuzzy models that are capable of adapting autonomously to changes in the data pattern (Angelov, 1999, 2000, 2002; Angelov and Buswell, 2001, 2002; Kasabov, 1998a, 1998b). Typically, an EFS learns autonomously without much user intervention in an online mode by analyzing each incoming sample, and adjusting both model structure and parameters. The online working mode of EFS involves a sequence of ‘predict’ and ‘update’ phases. In the prediction phase, when an input sample is received, it is fuzzified using the membership function and the output is estimated using the existing fuzzy rules and inference mechanism. Finally, the output is determined and defuzzified. The update (or learning) phase occurs when the actual output for the given input is received. During the update phase, the rule-base is updated through the learning module using the actual output and the previously estimated output. In the update phase usually an online clustering is applied on a per sample basis (or sometimes on a chunk of data). Upon receiving the output for the current input data sample, the online clustering process determines if a new cluster is required to be formed (with the current data sample as the cluster center) or an existing cluster is required to be modified in terms of shift of the existing cluster center or change in the range of influence of the existing cluster center. If a new cluster is generated then it in turn generates a new rule corresponding to that or if an existing cluster is updated then fuzzy sets of the corresponding rule get updated. After the update of antecedent parameters, the consequent parameters are updated using recursive least square method. Some of the online structure identification techniques that have been successfully applied to EFS design are evolving Clustering (eClustering) (Angelov and Filev, 2004), evolving Vector Quantization (eVQ) (Lughofer, 2008), Evolving Clustering Method (ECM) (Kasabov and Song, 2002), online Gustafsson–Kessel algorithm (Georgieva and Filev, 2009), evolving Participatory Learning (ePL) (Lima *et al.*, 2006), and Dynamically Evolving Clustering (DEC) (Baruah and Angelov, 2014).

4.5. Summary

In this chapter, we presented various approaches to fuzzy system modeling. The early design approaches were solely based on expert knowledge. The knowledge-based approach is easy to implement for simple systems. The chapter has described the knowledge-based method for fuzzy modeling with illustrative examples and guidelines. Though the knowledge-based method is simple, it is not suitable for complex systems and has several limitations. For example, one expert may not have the complete knowledge or understanding of a particular system, in such a scenario multiple experts are consulted. Finally, integrating the views of all the experts into a single system can be difficult, particularly when the views are conflicting. Also, if the expert knowledge about the system is faulty then the resulting model will be incorrect leading to undesirable results. Due to such reasons and availability of input– output data, automated methods are more preferred over knowledge-based methods. However, automated methods do not completely eliminate expert's involvement. When sufficient input–output data are available, the automated methods can be applied in three levels: (i) only to tune the antecedent and consequent parameters with fixed rules, (ii) to learn the rules with predefined membership functions and fuzzy sets, (iii) to learn both rules and parameters. The chapter has described various automated methods that have been applied at all the three levels. First, it described the template-based methods that works at level (ii), then it presented the neuro-fuzzy approach and described its application at level (i). The chapter discussed application of GAs to fuzzy system design at all the three levels. Finally, clustering-based approach has been explained that is applied at level (iii). The chapter has also provided a brief discussion on online modeling approaches. Over the past decade this area has received enormous attention from the researchers due to its applicability to various application domains that include robotics, process control, image processing, speech processing, bioinformatics, and finance. Readers interested in this area are encouraged to refer (Angelov *et al.*, 2010; Angelov, 2012; Lughofe, 2011).

References

- Angelov, P. (1999). Evolving fuzzy rule-based models. In *Proc. Eighth Int. Fuzzy Syst. Assoc. World Congr.* Taipei, Taiwan, 1, pp. 19–23.
- Angelov, P. (2000). Evolving fuzzy rule-based models. *J. Chin. Inst. Ind. Eng.*, 17, pp. 459–468.
- Angelov, P. (2002). *Evolving Rule-Based Models: A Tool for Design of Flexible Adaptive Systems*. Heidelberg: Physica-Verlag.
- Angelov, P. (2012). *Autonomous Learning Systems: From Data Streams to Knowledge in Real-time*. Chichester, UK: John Wiley & Sons.
- Angelov, P. and Buswell, R. (2001). Evolving rule-based models: a tool for intelligent adaptation. In Smith, M. H., Gruver, W. A. and Hall, L. O. (eds.), *Proc. Ninth Int. Fuzzy Syst. Assoc. World Congr.* Vancouver, Canada, 1–5, pp. 1062–1067.
- Angelov, P. and Buswell, R. (2002). Identification of evolving fuzzy rule-based models. *IEEE Trans. Fuzzy Syst.*, 10(5), pp. 667–677.
- Angelov, P. and Filev, D. P. (2004). An approach to online identification of Takagi–Sugeno fuzzy models. *IEEE Trans. Syst., Man, Cybern., Part B-Cybern.*, 34(1), pp. 484–498.
- Angelov, P., Filev, D. and Kasabov, N. (eds.). (2010). *Evolving Intelligent Systems: Methodology and Applications*. IEEE Press Series on Computational Intelligence. Hoboken, NJ: John Wiley & Sons.
- Babuška, R. and Verbruggen, H. B. (1996). An overview of fuzzy modeling for control. *Control Eng. Pract.*, 4(11), 1593–1606.
- Baruah, R. D. and Angelov, P. (2014). DEC: Dynamically evolving clustering and its application to structure identification of evolving fuzzy models. *IEEE Trans. Cybern.*, 44(9), pp. 1619–1631.
- Beliakov, G. and Warren, J. (2001). Appropriate choice of aggregation operators in fuzzy decision support systems. *IEEE Trans. Fuzzy Syst.*, 9(6), pp. 773–784.
- Bezdek, J. (1981). *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum Press.
- Chiu, S. L. (1994). Fuzzy model identification based on cluster estimation. *J. Intell. Fuzzy Syst.*, 2, pp. 267–278.
- Chiu, S. L. (1997). Extracting fuzzy rules from data for function approximation and pattern classification. In Dubois, D., Prade, H. and Yager, R. (eds.), *Fuzzy Information Engineering: A Guided Tour of Applications*. Hoboken, NJ: John Wiley & Sons.
- Cordon, O., Herrera, F. and Peregrin, A. (1997). Applicability of the fuzzy operators in the design of fuzzy logic controllers. *Fuzzy Sets Syst.*, 86(1), pp. 15–41.
- Cordón, O., Gomide, F., Herrera, F., Hoffmann, F. and Magdalena, L. (2004). Ten years of genetic fuzzy systems: current framework and new trends. *Fuzzy Sets Syst.*, 141(1), pp. 5–31.
- de Oliveira, J. V. and Pedrycz, W. (eds.). (2007). *Advances in Fuzzy Clustering and its Applications*. Hoboken, NJ: Wiley.
- Dunn, J. C. (1974). A fuzzy relative of the ISODATA process and its use in detecting compact, well-separated clusters. *J. Cybern.*, 3, pp. 32–57.
- Emami, M. R., Turksen, I. B. and Goldenberg, A. A. (1998). Development of a systematic methodology of fuzzy logic modeling. *IEEE Trans. Fuzzy Syst.*, 6, pp. 346–361.
- Georgieva, O. and Filev, D. (2009). Gustafsson–Kessel algorithm for evolving data stream clustering. In *Proc. Int. Conf. Comput. Syst. Technol.*, 3B, pp. 14–16.
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.
- Gustafsson, D. and Kessel, W. (1979). Fuzzy clustering with a fuzzy covariance matrix. In *Proc. IEEE CDC*, San Diego, California. Piscataway, New Jersey: IEEE Press, pp. 761–766.
- Hastie, T., Tibshirani, R. and Friedman, J. (2009). *The Elements of Statistical Learning Data Mining, Inference, and Prediction, Second Edition*. Heidelberg: Springer.

- Hellendoorn, H. and Thomas, C. (1993). Defuzzification in fuzzy controllers. *J. Intell. Fuzzy Syst.*, 1, pp. 109–123.
- Holland, J. H. and Reitman, J. S. (1978). Cognitive systems based on adaptive algorithms. In Waterman, D.A. and Roth, F.H.(eds.), *Pattern-Directed Inference Systems*. Waltham, MA: Academic Press.
- Jang, J.-S. R. (1993). ANFIS: adaptive-network-based fuzzy inference system. *IEEE Trans. Syst., Man, Cybern.*, 23(3), pp. 665–685.
- Kasabov, N. (1998a). ECOS: A framework for evolving connectionist systems and the ECO learning paradigm. In Usui, S. and Omori, T. (eds.), *Proc. Fifth Int. Conf. Neural Inf. Process.*, Kitakyushu, Japan: IOS Press, 1–3, pp. 1232–1235.
- Kasabov, N. (1998b). The ECOS framework and the ECO learning method for evolving connectionist system. *J. Adv. Comput. Intell.*, 2(6), pp. 195–202.
- Kasabov, N. and Song, Q. (2002). DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. *IEEE Trans. Fuzzy Syst.*, 10(2), pp. 144–154.
- Kinzel, J., Klawonn, F. and Kruse, R. (1994). Modifications of genetic algorithms for designing and optimizing fuzzy controllers. In *Proc. First IEEE Conf., IEEE World Congr. Comput. Intell., Evol. Comput.*, 1, pp. 28–33.
- Klawonn, F. (1994). Fuzzy sets and vague environments. *Fuzzy Sets Syst.*, 66, pp. 207–221.
- Kruse, R., Gebhardt, J. and Klawonn, F. (1994). *Foundations of Fuzzy Systems*. Chichester: Wiley.
- Lima, E., Gomide, F. and Ballini, R. (2006). Participatory evolving fuzzy modeling. In Angelov, P., Filev, D., Kasabov, N. and Cordon, O. (eds.), *Proc. Int. Symp. Evolving Fuzzy Syst.* Ambleside, Lake District, U.K.: IEEE Press, pp. 36–41.
- Liska, J. and Melsheimer, S. S. (1994). Complete design of fuzzy logic systems using genetic algorithms. In *Proc. Third IEEE Conf., IEEE World Congr. Comput. Intell., Fuzzy Syst.*, 2, pp. 1377–1382.
- Lughofer, E. D. (2008). Extensions of vector quantization for incremental clustering. *Pattern Recognit.*, 41(3), pp. 995–1011.
- Lughofer, E. (2011). *Evolving Fuzzy Systems: Methodologies, Advanced Concepts and Applications*. Berlin Heidelberg: Springer Verlag.
- Mamdani, E. H. (1997). Application of fuzzy logic to approximate reasoning using linguistic systems. *Fuzzy Sets Syst.*, 26, pp. 1182–1191.
- Mitra, S. and Hayashi, Y. (2000). Neuro-fuzzy rule generation: survey in soft computing framework. *IEEE Trans. Neural Netw.*, 11(3), pp. 748, 768.
- Pedrycz, W. (1984). An identification algorithm in fuzzy relational systems. *Fuzzy Sets Syst.*, 13(2), pp. 153–167.
- Ross, T. (2010). *Fuzzy Logic with Engineering Applications*. New York: McGraw-Hill.
- Smith, S. F. (1980). *A Learning System Based Genetic Adaptive Algorithms*. Doctoral dissertation. University of Pittsburgh, PA, USA: Department of Computer Science.
- Sugeno, M. and Yasukawa, T. (1993). A fuzzy-logic based approach to qualitative modeling. *IEEE Trans. Fuzzy Syst.*, 1, pp. 7–31.
- Takagi, T. and Sugeno, M. (1985). Fuzzy identification of systems and its application to modeling and control. *IEEE Trans. Syst., Man, Cybern.*, 15(1), pp. 116–132.
- Thrift, P. (1991). Fuzzy logic synthesis with genetic algorithms. In Rechard, K. B. and Lashon, B. B. (eds.), *Proc. Fourth Int. Conf. Genet. Algorithms*. San Diego, USA: Morgan Kaufman, pp. 509–513.
- Venturini, G. (1993). SIA: A supervised inductive algorithm with genetic search for learning attributes based concepts. In Pavel, B. B. (ed.), *Proc. Eur. Conf. Mach. Learn.* London, UK: Springer, pp. 280–296.
- Wang, L.-X. and Mendel, J. M. (1992). Generating fuzzy rules by learning from examples. *IEEE Trans. Syst. Man Cybern.*, 22(6), pp. 1414–1427.
- Wang, L. and Langari, R. (1994). Complex systems modeling via fuzzy logic. In *Proc. 33rd IEEE Conf. Decis. Control*, 4, Florida, USA, pp. 4136–4141.
- Yager, R. R. and Filev, D. P. (1993). Learning of fuzzy rules by mountain clustering. In Bruno, B. and James C. B. (eds.),

Proc. SPIE Conf. Appl. Fuzzy Log. Technol. Boston, MA, pp. 246–254.

Yager, R. R. and Filev, D. P. (1994). Approximate cluster via the mountain method. *IEEE Trans. Syst., Man, Cybern.*, 24(8), pp. 1279–1284.

Zhao, J., Wertz, V. and Gerez, R. (1994). A fuzzy clustering method for the identification of fuzzy models for dynamical systems. In *Proc. Ninth IEEE Int. Symp. Intell. Control*, Columbus, Ohio, USA: IEEE, pp. 172–177.

Chapter 5

Fuzzy Classifiers

Abdelhamid Bouchachia

Fuzzy classifiers as a class of classification systems have witnessed a lot of developments over many years now by various communities. Their main strengths stem from their transparency to human users and their capabilities to handle uncertainty often present in real-world data. Like with other classification systems, the construction of fuzzy classifiers follows the same development lifecycle. During training, in particular fuzzy classification rules are developed and undergo an optimization process before the classifier is tested and deployed. The first part of the present chapter overviews this process in detail and highlights the various optimization techniques dedicated to fuzzy rule-based systems. The second part of the chapter discusses a particular facet of research, that is, online learning of fuzzy classification systems. Throughout the chapter, a good review of the related literature is covered to highlight the various research directions of fuzzy classifiers.

5.1. Introduction

Over the recent years, fuzzy rule-based classification systems have emerged as a new class of attractive classifiers due to their transparency and interpretability characteristics. Motivated by such characteristics, fuzzy classifiers have been used in various applications such as smart homes (Bouchachia, 2011; Bouchachia and Vanaret, 2013), image classification (Thiruvenkadam *et al.*, 2006), medical applications (Tan *et al.*, 2007), pattern recognition (Toscano and Lyonnet, 2003), etc.

Classification rules are simple consisting of two parts: the premises (conditions) and consequents that correspond to class labels as shown in the following:

Rule 1 := If x_1 is small then Class 1

Rule 2 := If x_1 is large then Class 2

Rule 3 := If x_1 is medium and x_2 is very small then Class 1

Rule 4 := If x_2 is very large then Class 2

Rules may be associated with degrees of confidence that explains how good the rule covers a particular input region:

Rule 5 := If x_1 is small then Class 1 with confidence 0.8

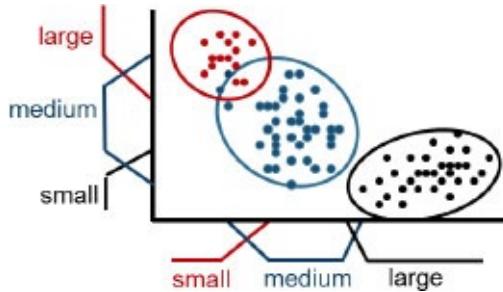


Figure 5.1: Two-dimensional illustrative example of specifying the antecedent part of a fuzzy if-then rule.

Graphically, the rules partition the space into regions. Ideally, each region is covered by one rule as shown in [Figure 5.1](#). Basically, there are two main approaches for designing fuzzy rule-based classifiers:

- Human expertise: The rules are explicitly proposed by the human expert. Usually, no tuning is required and the rules are used for predicting the classes of the input using certain inference steps (see [Section 5.3](#)).
- Machine generated: The standard way of building rule-based classifiers is to apply an automatic process which consists of certain steps: partitioning of the input space, finding the fuzzy sets of the rules' premisses and associating class labels as consequents. To predict the label of an input, an inference process is applied. Usually additional steps are involved, especially for optimizing the rule base (Fazzolari *et al.*, 2013).

Fuzzy classifiers come in the form of explicit if-then classification rules as illustrated

earlier. However, rules can also be encoded in neural networks resulting in neuro-fuzzy architectures (Kasabov, 1996). Moreover, different computational intelligence techniques have been used to develop fuzzy classifiers: evolutionary algorithms (Fazzolari *et al.*, 2013), rough sets (Shen and Chouchoulas, 2002), ant colony systems (Ganji and Abadeh, 2011), immune systems (Alatas and Akin, 2005), particle swarm optimization (Rani and Deepa, 2010), petri nets (Chen *et al.*, 2002), etc. These computational approaches are used for both design and optimizations of the classifier's rules.

5.2. Pattern Classification

The problem of pattern classification can be formally defined as follows. Let $T = \{(\mathbf{x}_i, y_i)_{i=1,\dots,N}\}$ be a set of training patterns. Each $\mathbf{x}_i = (x_{i1}, \dots, x_{id}) \in X$ is a d-dimensional vector and $y_i \in Y = \{y_1, \dots, y_C\}$, where Y denotes a discrete set of classes (i.e., labels). A classifier is inherently characterized by a decision function $f: X \rightarrow Y$ that is used to predict the class label y_i for each pattern \mathbf{x}_i . The decision function is learned from the training data patterns T drawn i.i.d. at random according to an unknown distribution from the space $X \times Y$. An accurate classifier has very small generalization error (i.e., it generalizes well to unseen patterns). In general, a loss (cost) is associated with errors, meaning that some misclassification errors are “worst” than others. Such a loss is expressed formally as a function $\ell(\mathbf{x}_i, y_i, f(\mathbf{x}_i))$ which can take different forms (e.g., 0–1, hinge, squared hinge, etc.) (Duda *et al.*, 2001). Ideally the classifier (i.e., decision function) should minimize the expected error $E(f)$ (known as the empirical risk) = $(1/N) \sum_{i=1}^N \ell(\mathbf{x}_i, y_i, f(\mathbf{x}_i))$. Hence, the aim is to learn a function f among a set of all functions that minimizes the error $E(f)$. The empirical risk may be regularized by putting additional constraints (terms) to restrain the function space, penalize the number of parameters and model complexity and avoid overfitting (which occurs when the classifier does not generalize well on unseen data). The regularized risk is then: $R(f) = E(f) + \lambda \Omega(f)$.

Classifiers can be either linear or nonlinear. A linear classifier (i.e., the decision function is linear and the classes are linearly separable) has the form: $y_i = g(\mathbf{w} \cdot \mathbf{x}_i) = g(\sum_j w_j x_{ij})$ where \mathbf{w} is a weight vector computed by learning from the training patterns. The function g outputs discrete values $\{1, -1\}$ or $\{0, 1\}$ which indicate the classes. Clearly the argument of decision function, $f = \sum_j w_j x_{ij}$, is a linear combination of the input and the weight vectors.

Linear classifiers are of types: discriminative or generative. The discriminative linear classifiers try to minimize R (or its simplest version E) without necessarily caring about the way the (training) patterns are generated. Examples of discriminative linear classifiers are: perceptron, support vector machines (SVM) and logistic regression and linear discriminant analysis (LDA).

Generative classifiers aim at learning a model of the joint probability $p(\mathbf{x}, y)$ over the data X and the labels Y . They aim at inferring class-conditional densities $p(\mathbf{x}|y)$ and priors $p(y)$. The prediction decision is made by exploiting the Bayes’ rule to compute $p(y|\mathbf{x})$ and then by selecting the label with the higher probability. The discriminative classifiers on the other hand aim at computing $p(y|\mathbf{x})$ directly from the pairs $(\mathbf{x}_i, y_i)_{i=1\dots N}$. Examples of generative linear classifiers are: probabilistic linear discriminant analysis (LDA) and the naive Bayes classifier.

Nonlinear classifiers (i.e., the decision function is not linear and can be quadratic, exponential, etc.). The straightforward formulation of nonlinear classifiers is the generalized

linear classifier which tries to nonlinearly map the input space onto another space ($X \subset \mathbb{R}^d \rightarrow Z \subset \mathbb{R}^K$), where the patterns are separable. Thus, $\mathbf{x} \in \mathbb{R}^d \mapsto \mathbf{x} \in \mathbb{R}^K$, $\mathbf{z} = [f_1(\mathbf{x}), \dots, f_K(\mathbf{x})]^T$ where f_k is a nonlinear function (e.g., log-sigmoid, tan-sigmoid, etc.). The decision function is then of the form: $y_i = g(\sum_k w_k f_k(x_i))$.

There are many nonlinear classification algorithms of different types such as generalized linear classifiers, multilayer perceptron, polynomial classifiers, radial basis function networks, etc., nonlinear SVM (kernels), k -nearest neighbor, rule-based classifiers (e.g., decision trees, fuzzy rule-based classifiers, different hybrid neuro-genetic-fuzzy classifiers), etc. (Duda *et al.*, 2001).

We can also distinguish between two classes of classifiers: symbolic and sub-symbolic classifiers. The symbolic classifiers are those that learn the decision function in the form of rules:

$$\text{If } x_{i1} \text{ is } A \text{ and } x_{i2} \text{ is } B \text{ and } \dots x_{id} \text{ is } Z \text{ then class is } C. \quad (1)$$

The advantage of this class of classifiers is their transparency and interpretability of their behavior. The sub-symbolic classifiers are those that do not operate on symbols and could be seen as black box.

5.3. Fuzzy Rule-Based Classification Systems

Fuzzy rule-based systems represent knowledge in the form of fuzzy “IF-THEN” rules. These rules can be either encoded by human experts or extracted from raw data by an inductive learning process. Generically, a fuzzy rule has the form:

$$R_r \equiv \text{If } x_1 \text{ is } A_{r,1} \wedge \cdots \wedge x_d \text{ is } A_{r,d} \text{ then } y_r \quad (2)$$

where $\{x_i\}_{i=1 \dots d}$ are fuzzy linguistic input variables, $A_{r,i}$ are antecedent linguistic terms in the form of fuzzy sets that characterize x_i and y_r is the output. This form can take different variations. However, the most known ones are: Mamdani-type, Takagi– Sugeno type and classification type. The former two ones have been introduced in the context of fuzzy control. Specifically, a rule of *Mamdani-type* has the following structure:

$$\text{If } x_1 \text{ is } A_{r,1} \wedge \cdots \wedge x_d \text{ is } A_{r,d} \text{ then } y_1 \text{ is } B_{r,1} \wedge \cdots \wedge y_m \text{ is } B_{r,m}, \quad (3)$$

where x_i are fuzzy linguistic input variables, y_j are fuzzy linguistic output variables, and $A_{r,i}$ and $B_{r,j}$ are linguistic terms in the form of fuzzy sets that characterize x_i and y_j . The Mamdani’s model is known for its transparency since all of the rule’s terms are linguistic terms.

The *Takagi–Sugeno type* (T–S) differs from the Mamdani-type model at the rule consequent level. In a T–S rule type, the consequent is a combination of the inputs as shown in Equation (4):

$$\text{If } x_1 \text{ is } A_{r,1} \wedge \cdots \wedge x_d \text{ is } A_{r,d} \text{ then } \begin{cases} y_{r,1} = f_{r,1}(x) \\ \vdots \\ y_{r,m} = f_{r,m}(x) \end{cases}. \quad (4)$$

A rule consists of d input fuzzy variables (x_1, x_2, \dots, x_d) and m output variables (y_1, y_2, \dots, y_m) such that $y_{r,j} = f_{r,j}(x)$. The most popular form of f is the polynomial form: $y_{r,j} = f_{r,j}(x) = b_{r,j}^1 x_1 + \cdots + b_{r,j}^d x_d + b_{r,j}^0$, where $b_{r,j}^i$ ’s denote the system’s $m + 1$ output parameters. These parameters are usually determined using iterative optimization methods. T–S-type FRS are used to approximate nonlinear systems by a set of linear systems.

In this chapter, we focus on classification systems where a rule looks as follows:

$$\text{If } x_1 \text{ is } A_{r,1} \wedge \cdots \wedge x_d \text{ is } A_{r,d} \text{ then } y_r \text{ is } C_r[\tau_r], \quad (5)$$

where C_r , τ_r indicate respectively the class label and the certainty factor associated with the rule r . A certainty factor represents the confidence degree of assigning the input to a class C_r , i.e., how good the rule covers the space of a class C_r . For a rule r , it is expressed as follows (Ishibuchi and Yamamoto, 2005):

$$\tau_r = \frac{\sum_{x_i \in C_r} \mu_{A_r}(x_i)}{\sum_{i=1}^N \mu_{A_r}(x_i)}, \quad (6)$$

where $\mu_{A_r}(x_i)$ is obtained as follows:

$$\mu_{A_r}(x_i) = \prod_{p=1}^d \mu_{A_{r,p}}(x_{ip}). \quad (7)$$

Equation (7) uses the product, but can also use any t-norm: $\mu_{A_r}(x_i) = T(\mu_{A_{r,1}}(x_{i1}), \dots, \mu_{A_{r,d}}(x_{id}))$. Moreover, the rule in Equation (5) could be generalized to multi-classes consequent to account for class distribution and in particular overlap

$$\text{If } x_1 \text{ is } A_{r,1} \wedge \dots \wedge x_d \text{ is } A_{r,d} \text{ then } y_r \text{ is } C_1[\tau_1], \dots, C_k[\tau_k]. \quad (8)$$

This formulation was adopted in particular in (Bouchachia and Mittermeir, 2007), where τ_j 's correspond to the proportion of patterns of each class in the region covered by the rule j .

Note also that the rule's form in Equation (5) could be seen as a special case of the Sugeno-type fuzzy rule model, in particular the zero-order model, where the consequent is a singleton.

Furthermore, there exists another type of fuzzy classification systems based on multidimensional fuzzy sets, where rules are expressed in the form (Bouchachia, 2011):

$$R_r \equiv \text{If } x \text{ is } K_i \text{ then } y_r \text{ is } C_j, \quad (9)$$

where K_i is a cluster. The rule means that if the sample x is CLOSE to K_j then the label of x should be that of class C_j .

Classes in the consequent part of the rules can be determined following two options:

- The generation of the fuzzy sets (partitions) is done using supervised clustering. That is, each partition will be labeled and will emanate from one class. In this case, certainty factors associated with the rules will be needless (Bouchachia and Mittermeir, 2007; Bouchachia, 2011; Bouchachia and Vanaret, 2013).
- The generation of the fuzzy sets is done using clustering, so that partitions can contain patterns from different classes at the same time. In this case, a special process is required to determine the rule consequents. Following the procedure described in (Ishibuchi and Nojima, 2007), the consequent is found as follows

$$y_r = \arg \max_{j=1 \dots C} \{v_j\}, \quad (10)$$

where v_j is given as:

$$v_j = \sum_{x_k \in C_r} \mu_r(x_k) = \sum_{x_k \in C_r} \left(\prod_{p=1}^d \mu_{A_{r,p}}(x_{kp}) \right). \quad (11)$$

Given a test pattern, x_k , entering the system, the output of the fuzzy classifier with respect to this pattern is the winner class, w , referred in the consequent of the rule with the highest activation degree. The winning class is computed as follows:

$$y(x_k) = \arg \max_{j=1 \dots C} \{w_j(x_k)\}, \quad (12)$$

where ω_j is given as:

$$w_j(x_k) = \max \{ \mu_r(x_k) * \tau_r | y_r = j \}. \quad (13)$$

If there is more than one winner (taking the same maximum value ω_j), then the pattern is considered unclassifiable.

A general fuzzy rule-based system consists mainly of four components: the knowledge base, the inference engine, the fuzzifier and the defuzzifier as shown in [Figure 5.2](#) and briefly described below. Note that generally fuzzy classifiers do not include the defuzzification step, since a fuzzy output is already indicative.

1. The knowledge base consists of a rule-base that holds a collection of fuzzy IF-THEN rules and a database that includes the membership functions defining the fuzzy sets used by the fuzzy rule system.
2. The inference engine maps the input to the rules and computes an aggregated fuzzy output of the system according to an inference method. The common inference method is the Max–Min composition, where rules are first combined with the input before aggregating using some operator (Mamdani, Gödel, etc.) to produce a final output or rules are combined first before acting on the input vector. Typical inference in classification systems is rather simple as explained earlier. It consists of sequentially computing the following degrees: activation [i.e., the product/t-norm of the antecedent part, see [Equation \(7\)](#)], association [i.e., combine the confidence factors with the activation degree to determine association degree of the input with the classes, see [Equation \(13\)](#)], and classification [i.e., choose the class with the highest association degree, see [Equation \(12\)](#)].

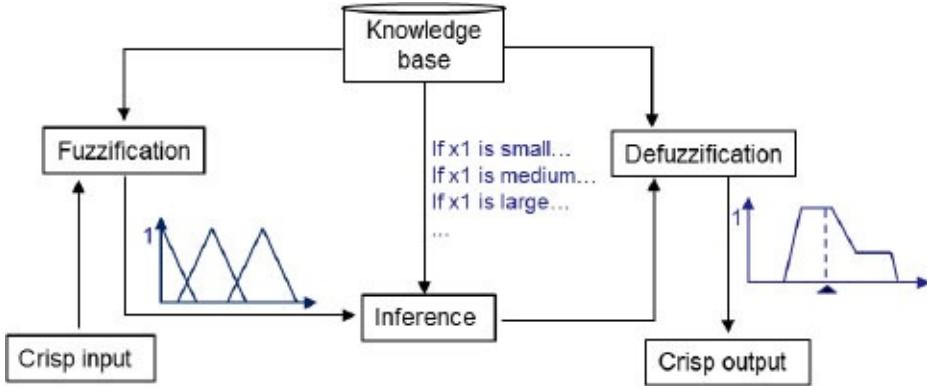


Figure 5.2: Structure of the adaptive fuzzy classifier.

3. Fuzzification transforms the crisp input into fuzzy input, by matching the input to the linguistic values in the rules' antecedents. Such values are represented as membership functions (e.g., triangular, trapezoidal and Gaussian) and stored in the database. These functions specify a partitioning of the input and eventually the output space. The number of fuzzy partitions determines that of the rules. There are three types of partitioning: grid, tree, and scatter. The latter is the most common one, since it finds the regions covering the data using clustering. Different clustering algorithms have been used (Vernieuwe *et al.*, 2006). Known algorithms are Gustafson–Kessel and Gath–Geva (Abonyi *et al.*, 2002), mountain (Yager and Filev, 1994), subtractive (Angelov, 2004), fuzzy hierarchical clustering (Tsekouras *et al.*, 2005), and FCM (Bouchachia, 2004).
4. The defuzzifier transforms the aggregated fuzzy output generated by the inference engine into a crisp output because very often a crisp output of the system is required for facilitating decision-making. The known defuzzification methods are: center of area, height-center of area, max criterion, first of maxima, and middle of maxima. However, the most popular method is center of area (Saade and Diab, 2000). Recall that fuzzy classification systems do not necessarily involve defuzzification.

5.4. Quality Issues of Fuzzy Rule-Based Systems

In general, knowledge maintenance entails the activities of knowledge validation, optimization, and evolution. The validation process aims at checking the consistency of knowledge by repairing/removing contradictions. Optimization seeks the compactness of knowledge by alleviating all types of redundancy and rendering the knowledge base transparent. Finally, evolution of knowledge is about incremental learning of knowledge without jeopardizing the old one allowing a continuous and incremental update of the knowledge.

This section aims at discussing the optimization issues of rule-based systems. Optimization in this context deals primarily with the transparency of the rule-base. Given the symbolic representation of rules, the goal is to describe the classifier with a small number of concise rules relying on transparent and interpretable fuzzy sets intelligible by human experts. Using data to automatically build the classifier is a process that does not always result in a transparent rule-base (Kuncheva, 2000). Hence, there is a need to optimize the number and the structure of the rules while keeping the classifier accuracy as high as possible (more on this tradeoff will follow in section). Note that beyond classification, the Mamdani model is geared towards more interpretability, hence the name of linguistic fuzzy modeling. Takagi–Sugeno model is geared towards accuracy, hence the name precise fuzzy modeling (Gacto *et al.*, 2011).

Overall, the quality of the rule-base (knowledge) can be evaluated using various criteria: performance, comprehensibility, and completeness which are explained in the following sections.

5.4.1. Performance

A classifier, independent of its computational model, is judged on its performance. That is, how well does the classifier perform on unseen novel data? Measuring the performance is about assessing the quality of decision-making.

Accuracy is one of the mostly used measures. It quantifies to which extent the system meets the human being decision. Thus, accuracy measures the ratio of correct decisions. Expressively, $Accuracy = \frac{Correct}{Test}$, where *correct* and *test* indicate respectively the number of patterns well classified (corresponding to true positives + true negatives decisions), while *Test* is the total number of patterns presented to the classifier). But accuracy as a sole measure for evaluating the performance of the classifier might be misleading in some situations like in the case of imbalanced classes. Obviously, different measures can be used, like precision (positive predictive value), false positive rate (false alarm rate), true positive rate (sensitivity), ROC curves (Fawcett, 2006). It is therefore recommended to use multiple performance measures to objectively evaluate the classifier.

5.4.2. Completeness

Knowledge is complete if for making a decision all needed knowledge elements are available. Following the architecture presented in [Figure 5.2](#), the system should have all ingredients needed to classify a pattern. The other aspect of completeness is the coverage of the discourse representation space. That is, all input variables (dimensions) to be considered should be fully covered by the fuzzy sets using *the frame of cognition* (FOC) (Pedrycz and Gomide, 1998), which stipulates that a fuzzy set (patch) along a dimension must satisfy: normality, typicality, full membership, convexity, and overlap.

5.4.3. Consistency

Consistency is a key issue in knowledge engineering and is considered as one important aspect of the rule-base comprehensibility assessment. In the absence of consistency, the knowledge is without value and its use leads to contradictory decisions. Inconsistency results from conflicting knowledge elements. For instance, two rules are in conflict if they have identical antecedents but different consequents. In the case of fuzzy rule-based classification, there is no risk to have inconsistency, since each rule corresponds to a given region of the data space. Moreover, even if an overlap between antecedents of various rules exists, the output for a given data point is unambiguously computed using the confidence factors related to each rule in the knowledge base.

5.4.4. Compactness

Compactness is about the conciseness and the ease of understanding and reasoning about the knowledge. Systems built on a symbolic ground, like rule-based systems, are easy to understand and to track how and why they reach a particular decision. To reinforce this characteristic, the goal of system design is to reduce, as much as possible, the number of rules to make the system's behavior more transparent. Thus, small number of rules and small number of conditions in the antecedent of rules ensures high compactness of the system's rule-base.

To reduce the complexity of the rule-base and consequently to get rid of redundancy and to strengthen compactness, the optimization procedure can consist of a certain number of steps (Bouchachia and Mittermeir, 2007). All these steps are based on similarity measures.

There are a number of measures based on set-theoretic, proximity, interval, logic, linguistic approximation and fuzzy-valued (Wu and Mendel, 2008). In the following, the optimization steps are described.

- Redundant partitions: They are discovered by computing the similarity of the fuzzy sets describing these partitions to the universe. A fuzzy set $A_{i,j}^r$ is removed if:

$$\text{Sim}(A_{i,j}^r, U) > \epsilon, \quad (14)$$

where $\epsilon \in (0, 1)$ and indicates a threshold (a required level of similarity), U indicates the universe that is defined as follows:

$$\forall x_k, \mu_U(x_{ki}) = 1.$$

Any pattern has a full membership in the universe of discourse.

- Merge of fuzzy partitions: Two fuzzy partition are merged if their similarity exceeds a certain threshold:

$$Sim(A_{ij}^r, A_{ik}^r) > \tau, \quad (15)$$

where A_{ij}^r, A_{ik}^r are the j th and k th partitions of the feature i in the rule r .

- Removal of weakly firing rules: This consists of identifying rules whose output is always close to 0.

$$\mu_i^r < \beta. \quad (16)$$

β is a threshold close to 0.

- Removal of redundant rules: There is redundancy if the similarity (e.g., overlap) between the antecedents of the rules is high exceeding some threshold δ . The similarity of the antecedents of two rules r and p is given as:

$$sim(A^r, A^p) = \min_{i,i=1,d} \{Sim(A_i^r, A_i^p)\}, \quad (17)$$

where the antecedents A^r and A^p are given by the set of fuzzy partitions representing the d features $A^r = \langle A_1^r, \dots, A_d^r \rangle$ and $A^p = \langle A_1^p, \dots, A_d^p \rangle$. That is, similar rules (i.e., having similar antecedents and same consequent) are merged. In doing so, if some rules have the same consequent, the antecedents of those rules can be connected. However, this may result in a conflict if the antecedents are not the same. One can however rely on the following rules of thumb:

- If, for some set of rules with the same consequent, a variable takes all forms (it belongs to all forms of fuzzy set, e.g., small, medium, large), then such a variable can be removed from the antecedent of that set of rules. In other words, this set of rules is independent of the variable that takes all possible values. This variable corresponds then to “*don’t care*”. For instance, if we consider [Figure 5.3](#), the rules 1, 3, and 4 are about class C_1 and the input variable x_2 takes all possible linguistic variable. The optimization will replace these rules with a generic rule as shown in [Figure 5.4](#).
- If, for some set of rules with the same consequent, a variable takes a subset of all possible forms (e.g., small and medium), then the antecedents of such rules can be combined by or(ing) the values corresponding to that variable. For instance, if we consider [Figure 5.3](#), the rules 2 and 5 are about class C_2 such that the variable x_3 takes the linguistic values: medium and large, while the other variables take the same values.

The two rules can be replaced with one generic rule as shown in [Figure 5.5](#).

	x_1	x_2	x_3	x_4	Class
1	S	S	S	S	1
2	M	S	M	M	2
3	S	M	S	S	1
4	S	L	S	S	1
5	M	S	L	M	2

Figure 5.3: Rules in the system's rule-base.

	x_1	x_2	x_3	x_4	Class
1	S	S	S	S	1
3	S	M	S	S	1
4	S	L	S	S	1
1'	S	—	S	S	1

Figure 5.4: Combining rules 1, 3 and 4.

	x_1	x_2	x_3	x_4	Class
2	M	S	M	M	2
5	M	S	L	M	2
2'	M	S	(M \vee L)	M	2

Figure 5.5: Combining rules 2 and 5.

5.4.4.1. Feature selection

To enhance the transparency and the compactness of the rules, it is important that the if-part of the rules does not involve many features. Low classification performance generally results from non-informative features. The very conventional way to get rid of these features is to apply feature selection methods. Basically, there exist three classes of feature selection methods (Guyon and Elisseeff, 2003):

1. Filters: The idea is to filter out features that have small potential to predict the outputs. The selection is done as a pre-processing step ahead of the classification task. Filters are preprocessing techniques and refer to statistical selection methods such as principal components analysis, LDA, and single value decomposition. For instance, Tikk *et al.* (2003) described a feature ranking algorithm for fuzzy modeling aiming at higher transparency of the rule-based. Relying on interclass separability as in LDA and using the backward selection method, the features are sequentially selected. Vanhoucke and Silipo (2003) used a number of measures that rely on mutual information to design a highly transparent classifier and particularly to select the features deemed to be the most informative. Similar approach has been taken in (Sanchez *et al.*, 2008) using a mutual information-based feature selection for optimizing a fuzzy rule-based system. Lee *et al.* (2001) applied fuzzy entropy (FE) in the context of fuzzy classification to achieve low complexity and high classification efficiency. First, FE was used to partition the pattern space into non-overlapping regions. Then, it was applied to select relevant features using the standard forward selection or backward elimination.
2. Wrappers: Select features that optimize the accuracy of a chosen classifier. Wrappers

largely depend on the classifier to judge how well feature subsets are at classifying the training samples. For instance, in Cintra and Camargo (2010), the authors use a fuzzy wrapper and a fuzzy C4.5 decision tree to identify discriminative features. Wrappers are not very popular in the area of fuzzy rule-based systems. But many references claimed their methods to be wrappers, while actually they are embedded.

3. Embedded: Embedded methods perform variable selection in the process of training and are usually specific to given learning machines. In del Jesus *et al.* (2003), to design a fuzzy rule-based classifier while selecting the relevant features, multi-objective genetic algorithms are used. The aim is to optimize the precision of the classifier. Similar approach is suggested in Chen *et al.* (2012), where the T–S model is considered. In the same vein, in Schmitt *et al.* (2008), the selection of features is done while a fuzzy rule classifier is being optimized using the Choquet Integral. The embedded methods dominate the other two categories due to the natural process of optimization rule-based system to obtain high interpretable rules.

5.5. Unified View of Rule-Based Optimization

Different taxonomies related to interpretability of fuzzy systems in general have been suggested:

- Corrado-Mencar and Fanelli (2008) proposed a taxonomy in terms of interpretability constraints to the following levels: the fuzzy sets, the universe of discourse, the fuzzy granules, the rules, the fuzzy models the learning algorithms.
- Zhou and Gan (2008) proposed a taxonomy in terms of low-level interpretability and high-level interpretability. Low-level interpretability is associated with the fuzzy set level to capture the semantics, while high-level interpretability is associated with the rule level.
- Gacto *et al.* (2011) proposed a taxonomy inspired from the second one and distinguished between complexity-based interpretability equivalent to the high-level interpretability and semantics-based interpretability equivalent to low-level interpretability associated with the previous taxonomy.

A straight way to deal with interpretability issues in a unified way, by considering both transparency and performance at the same time, is to use optimization methods. We can use either meta-heuristics (evolutionary methods) or special-purpose designed methods. In the following, some studies are briefly reported on.

Ishibuchi *et al.* (1997) proposed a genetic algorithm for rule selection in classification problems, considering the following two objectives: to maximize the number of correctly classified training patterns and minimize the number of selected rules. This improves the complexity of the model, thanks to the reduction in the number of rules and the use of dont care conditions in the antecedent part of the rule. Ishibuchi and Yamamoto (2004); Ishibuchi and Nojima (2007) present a multi-objective evolutionary algorithm (MOEA) for classification problems with three objectives: Maximizing the number of correctly classified patterns, minimizing the number of rules and minimizing the number of antecedent conditions. Narukawa *et al.* (2005) rely on NSGA-II to optimize the rule-base by eliminating redundant rules using a multi-objective optimization that aims at increasing the accuracy while minimizing the number of rules and the premises.

Additional studies using evolutionary algorithms, in particular multi-objective evolutionary algorithms, can be found in a recent survey by Fazzolari *et al.* (2013).

Different from the previously mentioned studies, others have used special purpose optimization methods. For instance, Mikut *et al.* (2005) used decision trees to generate rules before an optimization process is applied. This latter consists of feature selection using an entropy-based method and applying iteratively a search-based formula that combines accuracy and transparency to find the best configuration of the rule-base.

Nauck (2003) introduced a formula that combines by product three components:

complexity (expressed as the proportion of the number of classes to the total number of variables used in the rules), coverage (the average extent to which the domain of each variable is covered by the actual partitions of the variable), and partition complexity (quantified for each variable as inversely proportional to the number of partitions associated with that variable).

Guillaume and Charnomordic (2003) devised a distance-based formula to decide whether two partitions can be merged. The formula relies on the intra-distance (called internal distance) of a fuzzy set for a given variable and the inter-distance (called external distance) of fuzzy sets for a variable, similar to that used for computing clusters. Any pairs of fuzzy sets that minimize the combination of these two measures over the set of data points will be merged.

de Oliveira (1999a, 1999b) used backpropagation to optimize a performance index that consists of three constraining terms: accuracy, coverage and distinguishability of fuzzy sets.

5.6. Incremental and Online Fuzzy Rule-Based Classification Systems

Traditional fuzzy classification systems are designed in batch (offline) mode, that is, by using the complete training data at once. Thus, offline development of fuzzy classification systems assumes that the process of rule induction is done in a one-shot experiment, such that the learning phase and the deployment phase are two sequential and independent stages. For stationary processes this is sufficient, but if, for instance, the rule system's performance deteriorates due to a change of the data distribution or a change of the operating conditions, the system needs to be re-designed from scratch. Many offline approaches do simply perform "adaptive tuning", that is, they permanently re-estimate the parameters of the computed model. However, it is quite often necessary to adapt the structure of the rule-base. In general, for time-dependent and complex non-stationary processes, efficient techniques for updating the induced models are needed. Such techniques must be able to adapt the current model using only the new data. They have to be equipped with mechanisms to react to gradual changes or abrupt ones. The adaptation of the model (i.e., rules) should accommodate any information brought in by the new data and reconcile this with the existing rules.

Online development of fuzzy classification systems (Bouchachia and Vanaret, 2013), on the other hand, enables both learning and deployment to happen concurrently. In this context, rule learning takes place over long periods of time, and is inherently open-ended (Bouchachia and Mittermeir, 2007). The aim is to ensure that the system remains amenable to refinement as long as data continues to arrive. Moreover, online systems can also deal with both applications starving of data (e.g., experiments that are expensive and slow to produce data as in some chemical and biological applications) as well as applications that are data intensive (Arandjelovic and Cipolla, 2005; Bouchachia, 2011).

Generally, online systems face the challenge of accurately estimating the statistical characteristics of data in the future. In non-stationary environments, the challenge becomes even more important, since the FRS's behavior may need to change drastically over time due to concept drift (Bouchachia, 2011; Gama *et al.*, 2013). The aim of online learning is to ensure continuous adaptation. Ideally, only the learning model (e.g., only rules) and uses that model as basis in the future learning steps. As new data arrive, new rules may be created and existing ones may be modified allowing the system to evolve over time.

Online and incremental fuzzy rule systems have been recently introduced in a number of studies involving control (Angelov, 2004), diagnostic (Lughofer, 2011), and pattern classification (Angelov and Zhou, 2008; Bouchachia and Mittermeir, 2007; Lughofer, 2011). Type-1 fuzzy systems are currently quite established, since they do not only operate online, but also consider related advanced concepts such as concept drift and online feature selection.

For instance in Bouchachia and Mittermeir (2007), an integrated approach called

FRCS was proposed. To accommodate incremental rule learning, appropriate mechanisms are applied at all steps: (1) *Incremental supervised clustering* to generate the rule antecedents in a progressive manner, (2) *Online and systematic update* of fuzzy partitions, (3) *Incremental feature selection* using an *incremental version* of the Fisher's interclass separability criterion to dynamically select features in an online manner.

In Bouchachia (2011), a fuzzy rule-based system for online classification is proposed. Relying on fuzzy minmax neural networks, the paper explains how fuzzy rules can be continuously online generated to meet the requirements of non-stationary dynamic environments, where data arrives over long periods of time. The classifier is sensitive to drift. It is able to detect data drift (Gama *et al.*, 2013) using different measures and react to it. An outline of the algorithm proposed is described in Algorithm 1. Actually, IFCS consists of three steps:

Algorithm 1: Steps of the incremental fuzzy classification system (IFCS)

```

1: if Initial=true then
2:  $\mathcal{M} \leftarrow \text{Train\_Classifier}(<\text{TrainingData}, \text{Labels}>)$ 
3:  $\mathcal{E} \leftarrow \text{Test\_Classifier}(<\text{TestingData}, \text{Labels}>, \mathcal{M})$ 
   // Just for the sake of observation
4: end if
5:  $i \leftarrow 0$ 
6: while true do
7:  $i \leftarrow i + 1$ 
8: Read <Input, Label>
9: if IsLabeled(Label)=true then
10: if Saturation_Training(i)=false then
11:    $\mathcal{M} \leftarrow \text{Train\_Classifier}(<\text{Input}, \text{Label}>, \mathcal{M})$ 
12:   If Input falls in a hyperbox with Flabel, then Flabel  $\leftarrow$  Label
13: else
14:   Err  $\leftarrow \text{Test\_Classifier}(<\text{Input}, \text{Label}>, \mathcal{M})$ 
15:   Cumulated_Err  $\leftarrow \text{Cumulated\_Err} + \text{Err}$ 
16:   if Detect_Drift(Cumulated_Err)=true then
17:      $\mathcal{M} \leftarrow \text{Reset}(\text{Cumulated\_Err}, \mathcal{M})$ 
18:   else

```

```

19:    $\mathcal{M} \leftarrow \text{Update\_Classifier}(<\text{Input}, \text{Label}>, \mathcal{M})$ 
20:   If Input falls in a hyperbox with Flabel, then Flabel  $\leftarrow$  Label
21:   end if
22: end if
23: else
24:   Flabel  $\leftarrow \text{Predict\_Label}(\text{Input}, \mathcal{M})$ 
25:    $\mathcal{M} \leftarrow \text{Update\_Classifier}(<\text{Input}, \text{Flabel}>, \mathcal{M})$ 
26: end if
27: end while

```

- (a) Initial one-shot experiment training: Available data is used to obtain an initial model of the IFCS.
- (b) Training over time before saturation: Given a saturation training level, incoming data is used to further adjust the model.
- (c) Correction after training saturation: Beyond the saturation level, incoming data is used to observe the evolution of classification performance which allow to correct the classifier if necessary.

In Bouchachia and Vanaret (2013), a growing type-2 fuzzy classifier (GT2FC) for online fuzzy rule learning from real-time data streams is presented. To accommodate dynamic change, GT2FC relies on a new semi-supervised online learning algorithm called 2G2M (Growing Gaussian Mixture Model). In particular, 2G2M is used to generate the type-2 fuzzy membership functions to build the type-2 fuzzy rules. GT2FC is designed to accommodate data online and to reconcile labeled and unlabeled data using self-learning. Moreover, GT2FC maintains low complexity of the rule-base using online optimization and feature selection mechanisms. Type-2 fuzzy classification is very suitable for dealing with applications where input is prone to faults and noise. Thus, GT2FC offers the advantage of dealing with uncertainty in addition to self-adaptation in an online manner.

Note that at the operational level, T2 FRS differ from T1 FRS in the type of fuzzy sets and the operations applied on these sets. T2 fuzzy sets are equipped mainly with two newly introduced operators called the *meet*, \sqcap and *join*, \sqcup which correspond to the fuzzy intersection and fuzzy union. As shown in [Figure 5.7](#), T2 FRS at the structural level is similar to T1 FRS but contains an additional module, the *type-reducer*. In a classification type-2 fuzzy rule system, the fuzzy rules for a C-class pattern classification problem with n -input variable can be formulated as:

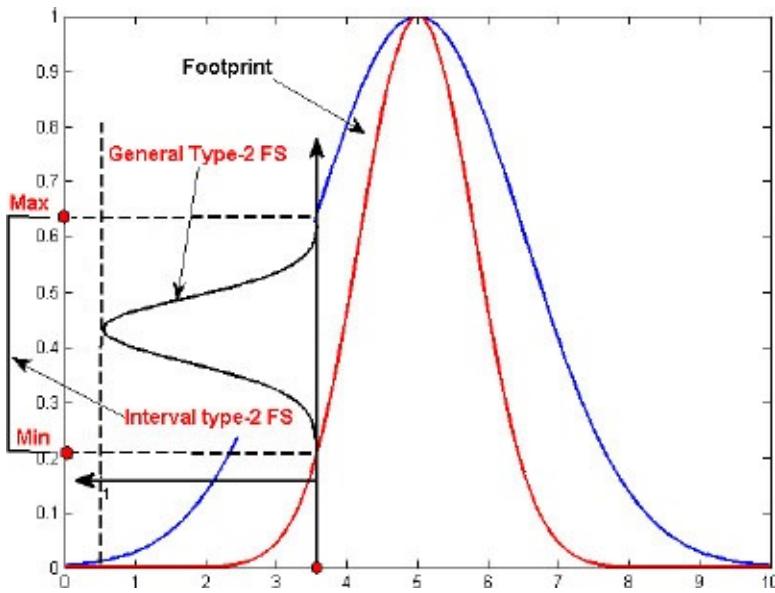


Figure 5.6: Type-2 fuzzy sets.

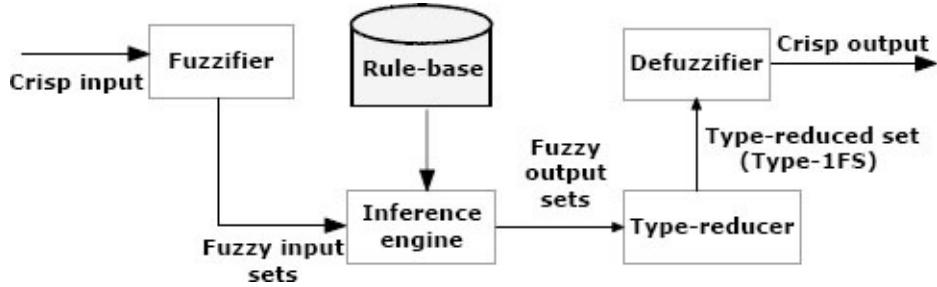


Figure 5.7: Type-2 fuzzy logic system.

$$R_j \equiv \text{if } x_1 \text{ is } \tilde{A}_{j,1} \wedge \dots \wedge x_n \text{ is } \tilde{A}_{j,n} \text{ then } C_i, \quad (18)$$

where $\mathbf{x} = [x_1, \dots, x_n]^t$ such that each x_i is a an input variable and $\tilde{A}_{r,i}$ the corresponding fuzzy terms in the form of type-2 fuzzy sets. We may associate these fuzzy sets with linguistic labels to enhance interpretability. C_i is a consequent class, and $j = 1, \dots, N$ is the number of fuzzy rules. The inference engine computes the output of type-2 fuzzy sets by combining the rules. Specifically, the meet operator is used to connect the type-2 fuzzy propositions in the antecedent. The degree of activation of the j th rule using the n input variables is computed as:

$$\tilde{\beta}_j(x) = \sqcap_{q=1}^n \mu_{\tilde{A}_{j,q}}(x_q), \quad j = 1, \dots, N. \quad (19)$$

The meet operation that replaces the fuzzy ‘and’ in T1 FRS is given as follows:

$$\tilde{A} \sqcap \tilde{B} = \sum_{u \in J_x} \sum_{w \in J_x} f_x(u) * g_x(w) / (u \wedge w), \quad x \in X. \quad (20)$$

If we use the interval Singleton T2 FRS, the meet is given for input $x = x'$ by the firing set, i.e.,

$$\begin{aligned} \tilde{\beta}_j(x) &= [\underline{\beta}_j(x), \bar{\beta}_j(x)], \\ &= [\underline{\mu}_{A_{j,1}}(x_1) * \dots * \underline{\mu}_{A_{j,n}}(x_n), \bar{\mu}_{A_{j,1}}(x_1) * \dots * \bar{\mu}_{A_{j,n}}(x_n)]. \end{aligned} \quad (21)$$

In (Bouchachia and Vanaret, 2013), the Gaussian membership function is adopted and it is given as follows:

$$\mu_A(x) = \exp \left\{ -\frac{(x - m)^2}{2\sigma^2} \right\} = \mathcal{N}(m, \sigma; x), \quad (22)$$

where m and σ are the mean and the standard deviation of the function. To generate the lower and upper membership functions, the authors used concentration and dilation hedges to generate the footprint of Gaussians with uncertain deviation as shown in [Figure 5.8](#). These are given as follows:

$$\bar{\mu}_{\tilde{A}}(x) = \mu_{DIL(A)}(x) = [\mu_A(x)]^{1/2}. \quad (23)$$

$$\underline{\mu}_{\tilde{A}}(x) = \mu_{CON(A)}(x) = [\mu_A(x)]^2. \quad (24)$$

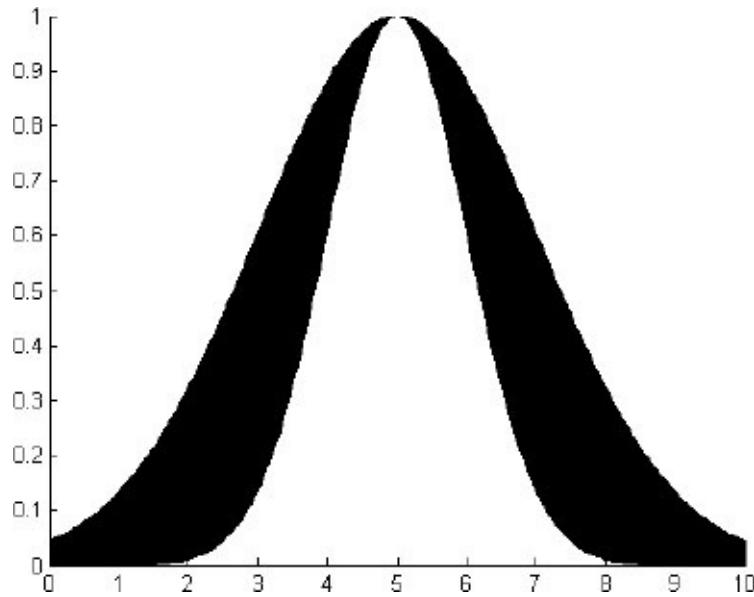


Figure 5.8: Uncertain deviation.

According to Equation (22), we obtain the following expressions:

$$\bar{\mu}_{\tilde{A}}(x) = \mathcal{N}(m, \sqrt{2}\sigma x). \quad (25)$$

$$\underline{\mu}_{\tilde{A}}(x) = \mathcal{N}\left(m, \frac{\sigma}{\sqrt{2}}; x\right). \quad (26)$$

Type-reducer is responsible for converting T2 FSs obtained as output of the inference engine into T1 FSs. Type-reduction for FRS (Mamdani and Sugeno models) was proposed by Karnik and Mendel (2001); Mendel (2013). This will not be adopted in our case, since the rules' consequent represents the label of a class. Traditionally in Type-1 fuzzy classification systems, the output of the classifier is determined by the rule that has the highest degree of activation:

$$\text{Winner} = C_{j*}, j* = \arg \max_{1 \leq j \leq N} \beta_j, \quad (27)$$

where β_j is the firing degree of the j rule. In type-2 fuzzy classification systems, we have an interval $\tilde{\beta}_j = [\underline{\beta}_j, \bar{\beta}_j]$ as defined in Equation (21). Therefore, we compute the winning class by considering the center of the interval $[\underline{\beta}_j(x), \bar{\beta}_j(x)]$, that is:

$$\text{Winner} = C_{j*}, \quad (28)$$

where

$$j* = \arg \max_{1 \leq j \leq N} \frac{1}{2}(\underline{\beta}_j(x) + \bar{\beta}_j(x)). \quad (29)$$

For the sake of illustration, an excerpt of rules generated is shown in [Table 5.1](#).

Table 5.1: Fuzzy rules for D2.

Rule	Antecedent	C
1	$x_1 \text{ IN } N(-17.60, [0.968, 1.93]) \wedge x_2 \text{ IN } N(-2.31, [1.42, 2.85]) \vee (x_2 \text{ IN } N(-9.45, [2.04, 4.08]))$	2
2	$x_1 \text{ IN } N(-5.27, [2.62, 5.24]) \vee x_1 \text{ IN } N(-13.10, [1.44, 2.89]) \vee x_2 \text{ IN } N(1.61, [1.01, 2.02])$	2
3	$x_1 \text{ IN } N(-13.10, [1.44, 2.89]) \wedge x_2 \text{ IN } N(1.61, [1.01, 2.02]) \vee x_2 \text{ IN } N(-15.12, [0.98, 1.96])$	2
4	$x_1 \text{ IN } N(5.47, [1.28, 2.56]) \wedge x_2 \text{ IN } N(7.78, [1.19, 2.39]) \vee x_2 \text{ IN } N(-6.10, [0.97, 1.94])$	1
5	$x_1 \text{ IN } N(-5.27, [2.62, 5.24]) \vee x_1 \text{ IN } N(5.470, [1.28, 2.56]) \vee x_1 \text{ IN } N(9.79, [1.89, 3.78]) \wedge x_2 \text{ IN } N(-5.30, [1.91, 3.83])$	1
6	$x_1 \text{ IN } N(2.69, [0.93, 1.87]) \wedge x_2 \text{ IN } N(-9.45, [2.04, 4.08])$	1
7	$x_1 \text{ IN } N(0.28, [0.97, 1.93]) \vee x_1 \text{ IN } N(-17.60, [0.96, 1.93]) \wedge x_2 \text{ IN } N(-2.31, [1.42, 2.85])$	2

5.7. Conclusion

This chapter briefly presents fuzzy rule-based classifiers. The working cycle for both type-1 and type-2 fuzzy classification systems has been described. Because the primary requirement of such systems is interpretability, quality issues have been lengthily discussed including various approaches with some illustrative studies. Towards the end, the chapter also introduces incremental and online learning of fuzzy classifiers.

While type-1 fuzzy classifiers have been extensively studied over the past in different contexts, type-2 fuzzy classifiers are still emerging and are not as popular as their predecessors. It is expected that this category of fuzzy classifiers will continue to be the focus of future studies, especially with regard to transparency, interpretability and online generation and deployment.

References

- Abonyi, J., Babuska, R. and Szeifert, F. (2002). Modified Gath–Geva fuzzy clustering for identification of Takagi–Sugeno fuzzy models. *IEEE Trans. Syst. Man Cybern. Part B*, 32(5), pp. 612–621.
- Alatas, B. and Akin, E. (2005). Mining fuzzy classification rules using an artificial immune system with boosting. In *ADBIS*, pp. 283–293.
- Angelov, P. (2004). An approach for fuzzy rule-base adaptation using on-line clustering. *Int. J. Approx. Reason.*, 35(3), pp. 275–289.
- Angelov, P. and Zhou, X. (2008). Evolving fuzzy rule-based classifiers from data streams. *IEEE Trans. Fuzzy Systems*, 16(6), pp. 1462–1475.
- Arandjelovic, O. and Cipolla, R. (2005). Incremental learning of temporally coherent Gaussian mixture models. In *Proc. 16th Br. Mach. Vis. Conf.*, pp. 759–768.
- Bouchachia, A. (2004). Incremental rule learning using incremental clustering. In *Proc. 10th Conf. Inf. Process. Manag. Uncertain. Knowl.-Based Syst.*, 3, pp. 2085–2092.
- Bouchachia, A. and Mittermeir, R. (2007). Towards incremental fuzzy classifiers. *Soft Comput.*, 11(2), pp. 193–207.
- Bouchachia, A. (2011). Fuzzy classification in dynamic environments. *Soft Comput.*, 15(5), pp. 1009–1022.
- Bouchachia, A. and Vanaret, C. (2013). Gt2fc: An online growing interval type-2 self-learning fuzzy classifier. *IEEE Trans. Fuzzy Syst.*, In press.
- Chen, X., Jin, D. and Li, Z. (2002). Fuzzy petri nets for rule-based pattern classification. In *Commun., Circuits Syst. West Sino Expositions, IEEE 2002 Int. Conf.*, June, 2, pp. 1218–1222.
- Chen, Y.-C., Pal, N. R. and Chung, I.-F. (2012). An integrated mechanism for feature selection and fuzzy rule extraction for classification. *IEEE Trans. Fuzzy Syst.*, 20(4), pp. 683–698.
- Cintra, M. and Camargo, H. (2010). Feature subset selection for fuzzy classification methods. In *Inf. Process. Manag. Uncertain Knowl.-Based Syst. Theory and Methods. Commun. Comput. Inf. Sci.*, 80, pp. 318–327.
- Corrado-Mencar, C. and Fanelli, A. (2008). Interpretability constraints for fuzzy information granulation. *Inf. Sci.*, 178(24), pp. 4585–4618.
- del Jesus, M., Herrera, F., Magdalena, L., Cordn, O. and Villar, P. (2003). *Interpretability Issues in Fuzzy Modeling*. A multiobjective genetic learning process for joint feature selection and granularity and context learning in fuzzy rule-based classification systems. Springer-Verlag, pp. 79–99.
- de Oliveira, J. (1999a). Semantic constraints for membership function optimization. *IEEE Trans. Syst. Man Cybern. Part A: Syst. Humans*, 29(1), pp. 128–138.
- de Oliveira, J. (1999b). Towards neuro-linguistic modeling: constraints for optimization of membership functions. *Fuzzy Sets Syst.*, 106, pp. 357–380.
- Duda, P., Hart, E. and Stork, D. (2001). *Pattern Classification*. New York: Willey.
- Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8), pp. 861–874.
- Fazzolari, M., Alcalá, R., Nojima, Y., Ishibuchi, H. and Herrera, F. (2013). A review of the application of multiobjective evolutionary fuzzy systems: Current status and further directions. *IEEE Trans. Fuzzy Syst.*, 21(1), pp. 45–65.
- Gacto, M., Alcalá, R. and Herrera, F. (2011). Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures. *Inf. Sci.*, 181(20), pp. 4340–4360.
- Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M. and Bouchachia, A. (2013). A survey on concept drift adaptation. *IEEE Trans. Fuzzy Syst.*, In press.
- Ganji, M. and Abadeh, M. (2011). A fuzzy classification system based on ant colony optimization for diabetes disease diagnosis. *Expert Syst. Appl.*, 38(12), pp. 14650–14659.
- Guillaume, S. and Charnomordic, B. (2003). *Interpretability Issues in Fuzzy Modeling*. A new method for inducing a set of interpretable fuzzy partitions and fuzzy inference systems from data. Springer-Verlag, p. 148175.
- Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3, pp. 1157–1182.

- Ishibuchi, H., Murata, T. and Türk, sen, I. (1997). Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems. *Fuzzy Sets Syst.*, 89(2), pp. 135–150.
- Ishibuchi, H. and Yamamoto, T. (2004). Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining. *Fuzzy Sets Syst.*, 141(1), pp. 59–88.
- Ishibuchi, H. and Nojima, Y. (2007). Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning. *Int. J. Approx. Reason.*, 44(1), pp. 4–31.
- Ishibuchi, H. and Yamamoto, T. (2005). Rule weight specification in fuzzy rule-based classification systems. *IEEE Trans. Fuzzy Syst.*, 13(4), pp. 428–435.
- Karnik, N. and Mendel, J. (2001). Operations on type-2 fuzzy sets. *Fuzzy Sets Syst.*, 122(2), pp. 327–348.
- Kasabov, N. (1996). *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering*. MIT Press.
- Kuncheva, L. (2000). How good are fuzzy if-then classifiers? *IEEE Trans. Syst. Man Cybern. Part B*, 30(4), pp. 501–509.
- Lee, H.-M., Chen, C.-M., Chen, J.-M. and Jou, Y.-L. (2001). An efficient fuzzy classifier with feature selection based on fuzzy entropy. *IEEE Trans. Syst. Man Cybern.*, 31, pp. 426–432.
- Lughofer, E. (2011). *Evolving Fuzzy Systems—Methodologies, Advanced Concepts and Applications. Studies in Fuzziness and Soft Computing*. Springer.
- Mendel, J. M. (2013). On km algorithms for solving type-2 fuzzy set problems. *IEEE Trans. Fuzzy Syst.*, 21(3), pp. 426–446.
- Mikut, R., Jäkel, J. and Gröll, L. (2005). Interpretability issues in data-based learning of fuzzy systems. *Fuzzy Sets Syst.*, 150(2), pp. 179–197.
- Narukawa, K., Nojima, Y. and Ishibuchi, H. (2005). Modification of evolutionary multi-objective optimization algorithms for multiobjective design of fuzzy rule-based classification systems. In *14th IEEE Int. Conf. Fuzzy Syst.*, pp. 809–814.
- Nauck, D. (2003). Measuring interpretability in rule-based classification systems. In *12th IEEE Int. Conf. Fuzzy Syst.*, 1, pp. 196–201.
- Pedrycz, W. and Gomide, F. (1998) *Introduction to Fuzzy Sets: Analysis and Design*. MIT Press.
- Rani, C. and Deepa, S. N. (2010). Design of optimal fuzzy classifier system using particle swarm optimization. In *Innov. Comput. Technol. (ICICT), 2010 Int. Conf.*, pp. 1–6.
- Saade, J. and Diab, H. (2000). Defuzzification techniques for fuzzy controllers. *IEEE Trans. Syst. Man Cybern. Part B: Cybern.*, 30(1), pp. 223–229.
- Schmitt, E., Bombardier, V. and Wendling, L. (2008). Improving fuzzy rule classifier by extracting suitable features from capacities with respect to the choquet integral. *IEEE Trans. Syst. Man Cybern. Part B: Cybern.*, 38(5), pp. 1195–1206.
- Shen, Q. and Chouchoulas, A. (2002). A rough-fuzzy approach for generating classification rules. *Pattern Recognit.*, 35(11), pp. 2425–2438.
- Sanchez, L., Surez, M. R., Villar, J. R. and Couso, I. (2008). Mutual information-based feature selection and partition design in fuzzy rule-based classifiers from vague data. *Int. J. Approx. Reason.*, 49(3), pp. 607–622.
- Tan, W., Foo, C. and Chua, T. (2007). Type-2 fuzzy system for ecg arrhythmic classification. In *FUZZ-IEEE*, pp. 1–6.
- Thiruvenkadam, S. R., Arcot, S. and Chen, Y. (2006). A pde based method for fuzzy classification of medical images. In *2006 IEEE Int. Conf. Image Process.*, pp. 1805–1808.
- Tikk, D., Gedeon, T. and Wong, K. (2003). *Interpretability Issues in Fuzzy Modeling*. A feature ranking algorithm for fuzzy modelling problems. Springer-Verlag, p. 176192.
- Toscano, R. and Lyonnet, P. (2003). Diagnosis of the industrial systems by fuzzy classification. *ISA Trans.*, 42(2), pp. 327–335.
- Tsekouras, G., Sarimveis, H., Kavakli, E. and Bafas, G. (2005). A hierarchical fuzzy-clustering approach to fuzzy modeling. *Fuzzy Sets Syst.*, 150(2), pp. 245–266.

- Vanhoucke, V. and Silipo, R. (2003). *Interpretability Issues in Fuzzy Modeling*. Interpretability in multidimensional classification. Springer-Verlag, p. 193217.
- Vernieuwe, H., De Baets, B. and Verhoest, N. (2006). Comparison of clustering algorithms in the identification of Takagi–Sugeno models: A hydrological case study. *Fuzzy Sets Syst.*, 157(21), pp. 2876–2896.
- Wu, D. and Mendel, J. (2008). A vector similarity measure for linguistic approximation: Interval type-2 and type-1 fuzzy sets. *Inf. Sci.*, 178(2), pp. 381–402.
- Yager, R. R. and Filev, D. P. (1994). Approximate clustering via the mountain method. *IEEE Trans. Syst. Man Cybern.*, 24(8), pp. 1279–1284.
- Zhou, S. and Gan, J. (2008). Low-level interpretability and high-level interpretability: A unified view of data-driven interpretable fuzzy system modelling. *Fuzzy Sets Syst.*, 159(23), pp. 3091–3131.

Chapter 6

Fuzzy Model-Based Control—Predictive and Adaptive Approaches

Igor Škrjanc and Sašo Blažič

This chapter deals with fuzzy model-based control and focuses on approaches that meet the following criteria: (1) The possibility of controlling nonlinear plants, (2) Simplicity in terms of implementation and computational complexity, (3) Stability and robust stability are ensured under some *a priori* known limitations. Two approaches are presented and discussed, namely, a predictive and an adaptive one. Both are based on Takagi–Sugeno model form, which possesses the property of universal approximation of an arbitrary smooth nonlinear function and can be therefore used as a proper model to predict the future behavior of the plant. In spite of many successful implementations of Mamdani fuzzy model-based approaches, it soon became clear that this approach lack the systematic ways to analyze the control systems stability, performance, robustness and the systematic way of tuning the controller parameters to adjust the performance. On the other hand, Takagi–Sugeno fuzzy models enable more compact description of nonlinear system, rigorous treatment of stability and robustness. But the most important feature of Takagi–Sugeno models is the ability that they can be easily adapted to the different linear control algorithms to cope with the demanding nonlinear control problems.

6.1. Introduction

In this chapter, we face the problem of controlling a nonlinear plant. Classical linear approaches try to treat a plant as linear and a linear controller is designed to meet the control objectives. Unfortunately, such an approach results in an acceptable control performance only if the nonlinearity is not too strong. The previous statement is far from being rigorous in the definitions of “acceptable” and “strong”. But the fact is that by increasing the control performance requirements, the problems with the nonlinearity also become much more apparent. So, there is a clear need to cope with the control of nonlinear plants.

The problem of control of nonlinear plants has received a great deal of attention in the past. A natural solution is to use a nonlinear controller that tries to “cancel” the nonlinearity in some sense or at least it increases the performance of the controlled system over a wide operating range with respect to the performance of an “optimal” linear controller. Since a controller is just a nonlinear dynamic system that maps controller inputs to controller actions, we need to somehow describe this nonlinear mapping and implement it into the controlled system. In the case of a finite-dimensional system, one possibility is to represent a controller in the state-space form where four nonlinear mappings are needed: state-to-state mapping, input-to-state mapping, state-to-output mapping, and direct input-to-output mapping. The Stone–Weierstrass theorem guarantees that all these mappings can be approximated by basis functions arbitrary well. Immense approximators of nonlinear functions have been proposed in the literature to solve the problem of nonlinear-system control. Some of the most popular ones are: piecewise linear functions, fuzzy models, artificial neural networks, splines, wavelets, etc.

In this chapter, we put our focus on fuzzy controllers. Several excellent books and papers exist that cover various aspects of fuzzy control (Babuska, 1998; Passino and Yurkovich, 1998; Pedrycz, 1993; Precup and Hellendoorn, 2011; Tanaka and Wang, 2002). Following a seminal paper of Zadeh (1973) that introduced fuzzy set theory, the fuzzy logic approach was soon introduced into controllers (Mamdani, 1974). In these early ages of fuzzy control, the controller was usually designed using Zadeh’s notion of linguistic variables and fuzzy algorithms. It was often claimed that the approach with linguistic variables, linguistic values and linguistic rules is “parameterless” and therefore the controllers are extremely easy to tune based on the expert knowledge that is easily transformable into the rule database. Many successful applications of fuzzy controllers have shown their ability to control nonlinear plants. But it soon became clear that the Mamdani fuzzy model control approach lacks systematic ways to analyze control system stability, control system performance, and control system robustness.

Takagi–Sugeno fuzzy model (Takagi and Sugeno, 1985) enables more compact description of the fuzzy system. Moreover, it enables rigorous treating of stability and robustness in the form of linear matrix inequalities (Tanaka and Wang, 2002). Several

control algorithms originally developed for linear systems can be adapted in a way that it is possible to combine them with the Takagi–Sugeno fuzzy models. Thus, the number of control approaches based on a Takagi–Sugeno model proposed in the literature in the past three decades is huge. In this chapter, we will show how to design predictive and adaptive controllers for certain classes of nonlinear systems where the plant model is given in the form of a Takagi–Sugeno fuzzy model. The proposed algorithms are not developed in an *ad hoc* manner, but having the stability of the over-all system in mind. This is why the stability analysis of the algorithms complements the algorithms themselves.

6.2. Takagi–Sugeno Fuzzy Model

Typical fuzzy model (Takagi and Sugeno, 1985) is given in the form of rules

$$\begin{aligned} R_j: \text{if } x_{p1} \text{ is } A_{1,k_1} \text{ and } x_{p2} \text{ is } A_{2,k_2} \text{ and } \dots \text{ and } x_{pq} \text{ is } A_{q,k_q} \text{ then } y = \phi_j(\mathbf{x}) \\ j = 1, \dots, m \\ k_1 = 1, \dots, f_1, \quad k_2 = 1, \dots, f_2 \quad \dots \quad k_q = 1, \dots, f_q. \end{aligned} \quad (1)$$

The q -element vector $\mathbf{x}_p^T = [x_{p1}, \dots, x_{pq}]$ denotes the input or variables in premise, and variable y is the output of the model. With each variable in premise x_{pi} ($i = 1, \dots, q$), f_i fuzzy sets $(A_{i,1}, \dots, A_{i,f_i})$ are connected, and each fuzzy set A_{i,k_i} ($k_i = 1, \dots, f_i$) is associated with a real-valued function $\mu_{A_{i,k_i}}(x_{pi}) : \mathbb{R} \rightarrow [0, 1]$, that produces membership grade of the variable x_{pi} with respect to the fuzzy set A_{i,k_i} . To make the list of fuzzy rules complete, all possible variations of fuzzy sets are given in Equation (1), yielding the number of fuzzy rules $m = f_1 \times f_2 \times \dots \times f_q$. The variables x_{pi} are not the only inputs of the fuzzy system. Implicitly, the n -element vector $\mathbf{x}^T = [x_1, \dots, x_n]$ also represents the input to the system. It is usually referred to as the consequence vector. The functions $\phi_j(\cdot)$ can be arbitrary smooth functions in general, although linear or affine functions are usually used.

The system in Equation (1) can be described in closed form if the intersection of fuzzy sets and the defuzzification method are previously defined. The generalized form of the intersection is the so-called *triangular norm* (T-norm). In our case, the latter was chosen as algebraic product while weighted average method was employed as a method of defuzzification yielding the following output of the fuzzy system

$$\hat{y} = \frac{\sum_{k_1=1}^{f_1} \sum_{k_2=1}^{f_2} \dots \sum_{k_q=1}^{f_q} \mu_{A_{1,k_1}}(x_{p1}) \mu_{A_{2,k_2}}(x_{p2}) \dots \mu_{A_{q,k_q}}(x_{pq}) \phi_j(\mathbf{x})}{\sum_{k_1=1}^{f_1} \sum_{k_2=1}^{f_2} \dots \sum_{k_q=1}^{f_q} \mu_{A_{1,k_1}}(x_{p1}) \mu_{A_{2,k_2}}(x_{p2}) \dots \mu_{A_{q,k_q}}(x_{pq})}. \quad (2)$$

It has to be noted that a slight abuse of notation is used in Equation (2) since j is not explicitly defined as running index. From Equation (1) is evident that each j corresponds to the specific variation of indexes k_i , $i = 1, \dots, q$.

To simplify Equation (2), a partition of unity is considered where functions $\beta_j(\mathbf{x}_p)$ defined by

$$\beta_j(\mathbf{x}_p) = \frac{\mu_{A_{1,k_1}}(x_{p1}) \mu_{A_{2,k_2}}(x_{p2}) \dots \mu_{A_{q,k_q}}(x_{pq})}{\sum_{k_1=1}^{f_1} \sum_{k_2=1}^{f_2} \dots \sum_{k_q=1}^{f_q} \mu_{A_{1,k_1}}(x_{p1}) \mu_{A_{2,k_2}}(x_{p2}) \dots \mu_{A_{q,k_q}}(x_{pq})}, \quad j = 1, \dots, m \quad (3)$$

give information about the fulfilment of the respective fuzzy rule in the normalized form. It is obvious that $\sum_{j=1}^m \beta_j(\mathbf{x}_p) = 1$ irrespective of \mathbf{x}_p as long as the denominator of $\beta_j(\mathbf{x}_p)$ is not equal to zero (that can be easily prevented by stretching the membership functions over the whole potential area of \mathbf{x}_p). Combining Equations (2) and (3) and changing

summation over k_i by summation over j , we arrive to the following equation:

$$\hat{y} = \sum_{j=1}^m \beta_j(\mathbf{x}_p) \phi_j(\mathbf{x}). \quad (4)$$

The class of fuzzy models has the form of linear models, this refers to $\{\beta^j\}$ as a set of basis functions. The use of membership functions in input space with overlapping receptive fields provides interpolation and extrapolation.

Very often, the output value is defined as a linear combination of signals in a consequence vector

$$\phi_j(\mathbf{x}) = \boldsymbol{\theta}_j^T \mathbf{x}, \quad j = 1, \dots, m, \quad \boldsymbol{\theta}_j^T = [\theta_{j1}, \dots, \theta_{jn}]. \quad (5)$$

If Takagi–Sugeno model of the zeroth order is chosen, $\phi_j(\mathbf{x}) = \theta_{j0}$, and in the case of the first-order model, the consequent is $\phi_j(\mathbf{x}) = \theta_{j0} + \boldsymbol{\theta}_j^T \mathbf{x}$. Both cases can be treated by the model Equation (5) by adding 1 to the vector \mathbf{x} and augmenting vector $\boldsymbol{\theta}$ with θ_{j0} . To simplify the notation, only the model in Equation (5) will be treated in the rest of the chapter. If the matrix of the coefficients for the whole set of rules is written as $\boldsymbol{\Theta}^T = [\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_m]$ and the vector of membership values as $\boldsymbol{\beta}^T(\mathbf{x}_p) = [\beta^1(\mathbf{x}_p), \dots, \beta^m(\mathbf{x}_p)]$, then Equation (4) can be rewritten in the matrix form

$$\hat{y} = \boldsymbol{\beta}^T(\mathbf{x}_p) \boldsymbol{\Theta} \mathbf{x}. \quad (6)$$

The fuzzy model in the form given in Equation (6) is referred to as affine Takagi– Sugeno model and can be used to approximate any arbitrary function that maps the compact set $\mathbf{C} \subset \mathbb{R}^d$ (d is the dimension of the input space) to \mathbb{R} with any desired degree of accuracy (Kosko, 1994; Wang and Mendel, 1992; Ying, 1997). The generality can be proven by Stone–Weierstrass theorem (Goldberg, 1976) which indicates that any continuous function can be approximated by fuzzy basis function expansion (Lin, 1997).

When identifying the fuzzy model, there are several parameters that can be tuned. One possibility is to only identify the parameters in the rule consequents and let the antecedent-part parameters untouched. If the position of the membership functions is good (the input space of interest is completely covered and the density of membership functions is higher where nonlinearity is stronger), then a good model can be obtained by only identifying the consequents. The price for this is to introduce any existing prior knowledge in the design of the membership functions. If, however, we do not know anything about the controlled system, we can use some evolving system techniques where the process of identification changes not only the consequent parameters but also the antecedent parameters (Angelov *et al.*, 2001; Angelov and Filev, 2004; Angelov *et al.*, 2011; Cara *et al.*, 2010; Johanyák and Papp, 2012; Sadeghi–Tehran *et al.*, 2012; Vaščák, 2012).

6.3. Fuzzy Model-Based Predictive Control

The fundamental methods which are essentially based on the principle of predictive control are Generalized Predictive Control (Clarke *et al.*, 1987), Model Algorithmic Control (Richalet *et al.*, 1978) and Predictive Functional Control (Richalet, 1993), Dynamic Matrix Control (Cutler and Ramaker, 1980), Extended Prediction Self-Adaptive Control (De Keyser *et al.*, 1988) and Extended Horizon Adaptive Control (Ydstie, 1984). All these methods are developed for linear process models. The principle is based on the process model output prediction and calculation of control signal which brings the output of the process to the reference trajectory in a way to minimize the difference between the reference and the output signal in a certain interval, between two prediction horizons, or to minimize the difference in a certain horizon, called coincidence horizon. The control signal can be found by means of optimization or it can be calculated using the explicit control law formula (Bequette, 1991; Henson, 1998).

The nature of processes is inherently nonlinear and this implies the use of nonlinear approaches in predictive control schemes. Here, we can distinguish between two main group of approaches: the first group is based on the nonlinear mathematical models of the process in any form and convex optimization (Figueroa, 2001), while the second group relies on approximation of nonlinear process dynamics with nonlinear approximators such as neural networks (Wang *et al.*, 1996; Wang and Mendel, 1992), piecewise-linear models (Padin and Figueroa, 2000), Volterra and Wiener models (Doyle *et al.*, 1995), multi-models and multi-variables (Li *et al.*, 2004; Roubos *et al.*, 1999), and fuzzy models (Abonyi *et al.*, 2001; Škrjanc and Matko, 2000). The advantage of the latter approaches is the possibility of stating the control law in the explicit analytical form.

In some highly nonlinear cases the use of nonlinear model-based predictive control can be easily justified. By introducing the nonlinear model into predictive control problem, the complexity increases significantly. In Bequette (1991); Henson (1998), an overview of different nonlinear predictive control approaches is given.

When applying the model-based predictive control with Takagi–Sugeno fuzzy model, it is always important how to choose fuzzy sets and corresponding membership functions. Many existing clustering techniques can be used in the identification phase to make this task easier. There exist many fuzzy model, based predictive algorithms (Andone and Hossu, 2004; Kim and Huh, 1998; Sun *et al.*, 2004) that put significant stress on the algorithm that properly arranges membership functions. An alternative approach is to introduce uncertainty to membership functions which results in the so-called type-2 fuzzy sets. Obviously, control algorithms exist based on type-2 fuzzy logic (Cervantes *et al.*, 2011).

The basic idea of model-based predictive control is to predict the future behavior of the process over a certain horizon using the dynamic model and obtaining the control actions to minimize a certain criterion. Traditionally, the control problem is formally

stated as an optimization problem where the goal is to obtain control actions on a relatively short control horizon by optimising the behavior of the controlled system in a larger prediction horizon. One of the main properties of most predictive controllers is that they utilize a so-called receding horizon approach. This means that even though the control signal is obtained on a larger interval in the current point of time, only the first sample is used while the whole optimization routine is repeated in each sampling instant.

The model-based predictive control relies heavily on the prediction model quality. When the plant is nonlinear, the model is also nonlinear. As it is very well-known, fuzzy model possesses the property of universal approximation of an arbitrary smooth nonlinear function and can be used as a prediction model of a nonlinear plant. Many approaches originally developed for the control of linear systems can be adapted to include a fuzzy model-based predictor. In this chapter fuzzy model is incorporated into the Predictive Functional Control (PFC) approach. This combination provides the means of controlling nonlinear systems. Since no explicit optimization is used, the approach is very suitable for the implementation on an industrial hardware. The fuzzy model-based predictive control (FMBPC) algorithm is presented in the state-space form (Blažič and Škrjanc, 2007). The approach is an extension of predictive functional algorithm (Richalet, 1993) to the nonlinear systems. The proposed algorithm easily copes with phase non-minimal and time-delayed dynamics. The approach can be combined by a clustering technique to obtain the FMBPC algorithm where membership functions are not fixed *a priori* but this is not intention of this work.

6.3.1. The Development of the Control Algorithm

In the case of FMBPC, the prediction of the plant output is given by its fuzzy model in the state-space domain. This is why the approach in the proposed form is limited to the open-loop stable plants. By introducing some modifications, the algorithm can be made applicable also for the unstable plants.

The problem of delays in the plant is circumvented by constructing an auxiliary variable that serves as the output of the plant if there were no delay present. The so-called “undelayed” model of the plant will be introduced for that purpose. It is obtained by “removing” delays from the original (“delayed”) model and converting it to the state space description:

$$\begin{aligned} \mathbf{x}_m^0(k+1) &= \bar{\mathbf{A}}_m \mathbf{x}_m^0(k) + \bar{\mathbf{B}}_m \mathbf{u}(k) + \bar{\mathbf{R}}_m, \\ \mathbf{y}_m^0(k) &= \bar{\mathbf{C}}_m \mathbf{x}_m^0(k), \end{aligned} \tag{7}$$

where $\mathbf{y}_m^0(k)$ models the “undelayed” output of the plant.

The behavior of the closed-loop system is defined by the reference trajectory which is given in the form of the reference model. The control goal is to determine the future control action so that the predicted output value coincides with the reference trajectory.

The time difference between the coincidence point and the current time is called a coincidence horizon. It is denoted by H . The prediction is calculated under the assumption of constant future manipulated variables ($u(k) = u(k + 1) = \dots = u(k + H - 1)$), i.e., the mean level control assumption is used. The H -step ahead prediction of the “undelayed” plant output is then obtained from Equation (7):

$$y_m^0(k + H) = \bar{\mathbf{C}}_m \left(\bar{\mathbf{A}}_m^H \mathbf{x}_m^0(k) + (\bar{\mathbf{A}}_m^H - \mathbf{I}) (\bar{\mathbf{A}}_m - \mathbf{I})^{-1} (\bar{\mathbf{B}}_m u(k) + \bar{\mathbf{R}}_m) \right). \quad (8)$$

The reference model is given by the first-order difference equation

$$y_r(k + 1) = a_r y_r(k) + b_r w(k), \quad (9)$$

where w stands for the reference signal. The reference model parameters should be chosen so that the reference model gain is unity. This is accomplished by fulfilling the following equation

$$(1 - a_r)^{-1} b_r = 1. \quad (10)$$

The main goal of the proposed algorithm is to find the control law which enables the reference trajectory tracking of the “undelayed” plant output $y_p^0(k)$. In each time instant, the control signal is calculated so that the output is forced to reach the reference trajectory after H time samples ($y_p^0(k + H) = y_r(k + H)$). The idea of FMBPC is introduced through the equivalence of the objective increment Δ_p and the model output increment Δ_m . The former is defined as a difference between the predicted reference signal $y_r(k + H)$ and the actual output of the “undelayed” plant $y_p^0(k)$

$$\Delta_p = y_r(k + H) - y_p^0(k). \quad (11)$$

Since the variable $y_p^0(k)$ cannot be measured directly, it will be estimated by using the available signals:

$$y_p^0(k) = y_p(k) - y_m(k) + y_m^0(k). \quad (12)$$

It can be seen that the delay in the plant is compensated by the difference between the outputs of the “undelayed” and the “delayed” model. When the perfect model of the plant is available, the first two terms on the right-hand side of Equation (12) cancel and the result is actually the output of the “undelayed” plant model. If this is not the case, only the approximation is obtained.

The model output increment Δ_m is defined by the following formula:

$$\Delta_m = y_m^0(k + H) - y_m^0(k). \quad (13)$$

Prescribing the control goal

$$\Delta_p = \Delta_m \quad (14)$$

and taking into account Equations (11), (13), and (8), we obtain the FMBPC control law

$$u(k) = g_0^{-1} \left((y_r(k+H) - y_p^0(k) + y_m^0(k)) - \bar{\mathbf{C}}_m \bar{\mathbf{A}}_m^H \mathbf{x}_m^0(k) \right. \\ \left. - \bar{\mathbf{C}}_m (\bar{\mathbf{A}}_m^H - \mathbf{I}) (\bar{\mathbf{A}}_m - \mathbf{I})^{-1} \bar{\mathbf{R}}_m \right), \quad (15)$$

where g_0 stands for:

$$g_0 = \bar{\mathbf{C}}_m (\bar{\mathbf{A}}_m^H - \mathbf{I}) (\bar{\mathbf{A}}_m - \mathbf{I})^{-1} \bar{\mathbf{B}}_m. \quad (16)$$

The control law of FMBPC in analytical form is finally obtained by introducing Equation (12) into Equation (15):

$$u(k) = g_0^{-1} \left((y_r(k+H) - y_p(k) + y_m(k)) - \bar{\mathbf{C}}_m \bar{\mathbf{A}}_m^H \mathbf{x}_m^0(k) \right. \\ \left. - \bar{\mathbf{C}}_m (\bar{\mathbf{A}}_m^H - \mathbf{I}) (\bar{\mathbf{A}}_m - \mathbf{I})^{-1} \bar{\mathbf{R}}_m \right). \quad (17)$$

In the following, it will be shown that the realizability of the control law, Equation (17) relies heavily on the relation between the coincidence horizon H and the relative degree of the plant ρ . In the case of discrete-time systems, the relative degree is directly related to the pure time-delay of the system transfer function. If the system is described in the state-space form, any form can be used in general, but the analysis is much simplified in the case of certain canonical descriptions. If the system is described in controllable canonical form in each fuzzy domain, then also the matrices $\bar{\mathbf{A}}_m$, $\bar{\mathbf{B}}_m$, and $\bar{\mathbf{C}}_m$ of the fuzzy model, Equation (36), take the controllable canonical form:

$$\bar{\mathbf{A}}_m = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -\bar{a}_n & -\bar{a}_{n-1} & -\bar{a}_{n-2} & \dots & -\bar{a}_1 \end{bmatrix}, \quad \bar{\mathbf{B}}_m = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \\ \bar{\mathbf{C}}_m = [\bar{b}_n \ \dots \ \bar{b}_\rho \ 0 \ \dots \ 0], \quad \bar{\mathbf{R}}_m = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \bar{r} \end{bmatrix}, \quad (18)$$

where $\bar{a}_j = \sum_{i=1}^m \beta_i a_{ji}$, $j = 1, \dots, n$, $\bar{b}_j = \sum_{i=1}^m \beta_i b_{ji}$, $j = \rho, \dots, n$, and $\bar{r} = \sum_{i=1}^m \beta_i (r_i/b_i)$, $j = 1, \dots, n$, and where the parameters a_{ji} , b_{ji} and r_i are state-space model parameters defined as in Equation (35). Note that the state-space system with matrices from Equation (19) has relative degree ρ what is reflected in the form of matrix $\bar{\mathbf{C}}_m$ – last $(\rho - 1)$ elements are equal to 0 while $\bar{b}_\rho \neq 0$.

Proposition 1.1. If the coincidence horizon H is lower than the plant relative degree ρ ($H < \rho$), then the control law, Equation (17) is not applicable.

Proof. By taking into account the form of matrices in Equation (19), it can easily be shown that

$$(\bar{\mathbf{A}}_m^H - \mathbf{I}) (\bar{\mathbf{A}}_m - \mathbf{I})^{-1} \bar{\mathbf{B}}_m = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ \cdot \\ \vdots \\ \cdot \end{bmatrix}, \quad (19)$$

i.e., the first $(n - H)$ elements of the vector are zeros, then there is the element 1, followed by $(H - 1)$ arbitrary elements. It then follows from Equations (16) and (19) that $g_0 = 0$ if $\rho > H$, and consequently the control law cannot be implemented.

The closed-loop system analysis makes sense only in the case of non-singular control law. Consequently, the choice of H is confined to the interval $[\rho, \infty)$.

6.3.2. Stability Analysis

The stability analysis of the proposed predictive control can be performed using an approach of linear matrix inequalities (LMI) proposed in Wang *et al.* (1996) and Tanaka *et al.* (1996) or it can be done assuming the frozen-time theory (Leith and Leithead, 1998, 1999) which discusses the relation between the nonlinear dynamical system and the associated linear time-varying system. There also exist alternative approaches for stability analysis (Baranyi *et al.*, 2003; Blažič *et al.*, 2002; Perng, 2012; Precup *et al.*, 2007).

In our stability study, we have assumed that the frozen-time system given in Equation (36) is a perfect model of the plant, i.e., $y_p(k) = y_m(k)$ for each k . Next, it is assumed that there is no external input to the closed-loop system ($w = 0$)—an assumption often made when analysing stability of the closed-loop system. Even if there is external signal, it is important that it is bounded. This is assured by selecting stable reference model, i.e., $|a_r| < 1$. The results of the stability analysis are also qualitatively the same if the system operates in the presence of bounded disturbances and noise.

Note that the last term in the parentheses of the control law Equation (17) is equal to $g_0 \bar{r}$. This is obtained using Equations (16) and (19). Taking this into account and considering the above assumptions the control law, Equation (17) simplifies to:

$$u(k) = g_0^{-1} (-\bar{\mathbf{C}}_m \bar{\mathbf{A}}_m^H \mathbf{x}_m^0(k)) - \bar{r}. \quad (20)$$

Inserting the simplified control law (20) into the model of the “undelayed” plant. Equation

(7), we obtain:

$$\mathbf{x}_m(k+1) = (\bar{\mathbf{A}}_m - \bar{\mathbf{B}}_m g_0^{-1} \bar{\mathbf{C}}_m \mathbf{A}_m^H) \mathbf{x}_m^0(k). \quad (21)$$

The closed-loop state transition matrix is defined as:

$$\mathbf{A}_c = \bar{\mathbf{A}}_m - \bar{\mathbf{B}}_m g_0^{-1} \bar{\mathbf{C}}_m \bar{\mathbf{A}}_m^H. \quad (22)$$

If the system is in the controllable canonical form, the second term on the right-hand side of Equation (22) has non-zero elements only in the last row of the matrix, and consequently \mathbf{A}_c is also in the Frobenius form. The interesting form of the matrix is obtained in the case $H = \rho$. If $H = \rho$, it can easily be shown that $g_0 = \bar{b}_\rho$ and that \mathbf{A}_c takes the following form:

$$\mathbf{A}_c = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \\ 0 & \cdots & 0 & -\frac{\bar{b}_n}{\bar{b}_\rho} & \cdots & -\frac{\bar{b}_{\rho+1}}{\bar{b}_\rho} \end{bmatrix}. \quad (23)$$

The corresponding characteristic equation of the system is:

$$z^\rho \left(z^{n-\rho} + \frac{\bar{b}_{\rho+1}}{\bar{b}_\rho} z^{n-\rho-1} + \frac{\bar{b}_{\rho+2}}{\bar{b}_\rho} z^{n-\rho-2} + \cdots + \frac{\bar{b}_{n-1}}{\bar{b}_\rho} z + \frac{\bar{b}_n}{\bar{b}_\rho} \right) = 0. \quad (24)$$

The solutions of this equation are closed-loop system poles: ρ poles lie in the origin of the z -plane while the other $(n - \rho)$ poles lie in the roots of the polynomial $\bar{b}_\rho z^{n-\rho} + \bar{b}_{\rho+1} z^{n-\rho-1} + \cdots + \bar{b}_{n-1} z + \bar{b}_n$. These results can be summarized in the following proposition:

Proposition 1.2. *When the coincidence horizon is equal to the relative degree of the model ($H = \rho$), then $(n - \rho)$ closed-loop poles tend to open-loop plant zeros, while the rest (ρ) of the poles go to the origin of z -plane.*

The proposition states that the closed-loop system is stable for $H = \rho$ if the plant is minimum phase. When this is not the case, the closed-loop system would become unstable if H is chosen equal to ρ . In such case the coincidence horizon should be larger.

The next proposition deals with the choice of a very large coincidence horizon.

Proposition 1.3. *When the coincidence horizon tends to infinity ($H \rightarrow \infty$) and the open-loop plant is stable, the closed-loop system poles tend to open-loop plant poles.*

Proof. The proposition can be proven easily. In the case of stable plants, \mathbf{A}_m is a Hurwitz

matrix that always satisfies:

$$\lim_{H \rightarrow \infty} \mathbf{A}_m^H = 0. \quad (25)$$

Combining Equations (22) and (25), we arrive at the final result

$$\lim_{H \rightarrow \infty} \mathbf{A}_c = \mathbf{A}_m. \quad (26)$$

The three propositions give some design guidelines on choosing the coincidence horizon H . If $H < \rho$, the control law is singular and thus not applicable. If $H = \rho$, the closed-loop poles go to open-loop zeros, i.e., high-gain controller is being used. If H is very large, the closed-loop poles go to open-loop poles, i.e., low-gain controller is being used and the system is almost open-loop. If the plant is stable, the closed-loop system can be made stable by choosing coincidence horizon large enough.

6.3.3. Practical Example—Continuous Stirred-Tank Reactor

The simulated continuous stirred-tank reactor (CSTR) process consists of an irreversible, exothermic reaction, $A \rightarrow B$, in a constant volume reactor cooled by a single coolant stream, which can be modelled by the following equation (Morningred *et al.*, 1992):

$$\dot{C}_A^0 = \frac{q}{V} [C_{A0} - C_A^0] - k_0 C_A^0 \exp\left(\frac{-E}{RT}\right). \quad (27)$$

$$\begin{aligned} \dot{T} = & \frac{q}{V} (T_0 - T) - \frac{k_0 \Delta H}{\rho C_p} C_A^0 \exp\left(\frac{-E}{RT}\right) + \frac{\rho_c C_{pc}}{\rho C_p V} q_c \\ & \times \left[1 - \exp\left(-\frac{h_A}{q_c \rho_c C_{pc}}\right) \right] (T_{c0} - T). \end{aligned} \quad (28)$$

The actual concentration C_A^0 is measured with a time delay $t_d = 0.5$ min:

$$C_A(t) = C_A^0(t - t_d). \quad (29)$$

The objective is to control the concentration of $A(C_A)$ by manipulating the coolant flow rate q_c . This model is a modified version of the first tank of a two-tank CSTR example from Henson and Seborg (1990). In the original model, the time delay was zero.

The symbol q_c represents the coolant flow rate (manipulated variable) and the other symbols represent constant parameters whose values are defined in [Table 6.1](#). The process dynamics are nonlinear due to the Arrhenius rate expression which describes the dependence of the reaction rate constant on the temperature (T). This is why the CSTR exhibits some operational and control problems. The reactor presents multiplicity behavior with respect to the coolant flow rate q_c , i.e., if the coolant flow rate $q_c \in (11.1 \text{ l/min}, 119.7 \text{ l/min})$ there are three equilibrium concentrations C_A . Stable equilibrium points are obtained in the following cases:

- $q_c > 11.1 \text{ l/min} \Rightarrow$ stable equilibrium point $0.92 \text{ mol/l} < C_A < 1 \text{ mol/l}$.
- $q_c < 111.8 \text{ l/min} \Rightarrow$ stable equilibrium point $C_A < 0.14 \text{ mol/l}$ (the point where $q_c \approx 111.8 \text{ l/min}$ is a Hopf Bifurcation point).

If $q_c \in (11.1 \text{ l/min}, 119.7 \text{ l/min})$, there is also at least one unstable point for the measured product concentration C_A . From the above facts, one can see that the CSTR exhibits quite complex dynamics. In our application, we are interested in the operation in the stable operating point given by $q_c = 103.41 \text{ l min}^{-1}$ and $C_A = 0.1 \text{ mol/l}$.

Table 6.1: Nominal CSTR parameter values.

Measured product concentration	C_A	0.1 mol/l
Reactor temperature	T	438.54 K
Coolant flow rate	q_c	$103.41 \text{ l min}^{-1}$
Process flow rate	q	100 l min^{-1}
Feed concentration	C_{A0}	1 mol/l
Feed temperature	T_0	350 K
Inlet coolant temperature	T_{c0}	350 K
CSTR volume	V	100 l
Heat transfer term	hA	$7 \times 10^5 \text{ cal min}^{-1} \text{ K}^{-1}$
Reaction rate constant	k_0	$7.2 \times 10^{10} \text{ min}^{-1}$
Activation energy term	E/R	$1 \times 10^4 \text{ K}$
Heat of reaction	ΔH	$-2 \times 10^5 \text{ cal/mol}$
Liquid densities	ρ, ρ_c	$1 \times 10^3 \text{ g/l}$
Specific heats	C_p, C_{pc}	$1 \text{ cal g}^{-1}\text{K}^{-1}$

6.3.3.1. Fuzzy identification of the continuous stirred-tank reactor

From the description of the plant, it can be seen that there are two variables available for measurement—measured product concentration C_A and reactor temperature T . For the purpose of control, it is certainly beneficial to make use of both although it is not necessary to feed back reactor temperature if one wants to control product concentration. In our case, the simple discrete compensator was added to the measured reactor temperature output:

$$\Delta q_{ff} = K_{ff} [T(k) - T(k-1)], \quad (30)$$

where K_{ff} was chosen to be 3, while the sampling time $T_s = 0.1 \text{ min}$. The above compensator is a sort of the D-controller that does not affect the equilibrium points of the system (the static curve remains the same), but it does to some extent affect their stability.

In our case the Hopf bifurcation point moved from $(q_c, C_A) = (111.8 \text{ l/min}, 0.14 \text{ mol/l})$ to $(q_c, C_A) = (116.2 \text{ l/min}, 0.179 \text{ mol/l})$. This means that the stability interval for the product concentration C_A expanded from $(0, 0.14 \text{ mol/l})$ to $(0, 0.179 \text{ mol/l})$. The proposed FMBPC will be tested on the compensated plant, so we need fuzzy model of the compensated plant.

The plant was identified in a form of discrete second-order model with the premise defined as $\mathbf{x}_p^T = [C_A(k)]$ and the consequence vector as $\mathbf{x}^T = [C_A(k), C_A(k - 1), q_c(k - T_{D_m}), 1]$. The functions $\phi_j(\cdot)$ can be arbitrary smooth functions in general, although linear or affine functions are usually used. Due to strong nonlinearity the structure with six rules and equidistantly shaped gaussian membership functions was chosen. The normalized membership functions are shown in Figure 6.1.

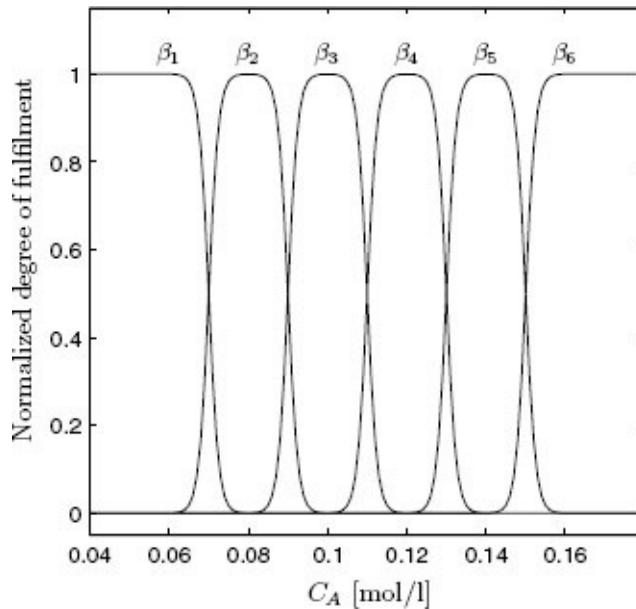


Figure 6.1: The membership functions.

The structure of the fuzzy model is the following:

$$\begin{aligned} R_j: & \text{if } C_A(k) \text{ is } A_j \text{ then } C_A(k + 1) \\ &= -a_{1j}C_A(k) - a_{2j}C_A(k - 1) + b_{1j}q_c(k - T_{D_m}) + r_j \quad j = 1, \dots, 6. \end{aligned} \tag{31}$$

The parameters of the fuzzy form in Equation (31) have been estimated using least square algorithm where the data have been preprocessed using QR factorization (Moon and Stirling, 1999). The estimated parameters can be written as vectors $\mathbf{a}_1^T = [a_{11}, \dots, a_{16}]$, $\mathbf{a}_2^T = [a_{21}, \dots, a_{26}]$, $\mathbf{b}_1^T = [b_{11}, \dots, b_{16}]$ and $\mathbf{r}_1^T = [r_{11}, \dots, r_{16}]$. The estimated parameters in the case of CSTR are as follows:

$$\begin{aligned} \mathbf{a}_1^T &= [-1.3462, -1.4506, -1.5681, -1.7114, -1.8111, -1.9157] \\ \mathbf{a}_2^T &= [0.4298, 0.5262, 0.6437, 0.7689, 0.8592, 0.9485] \\ \mathbf{b}_1^T &= [1.8124, 2.1795, 2.7762, 2.6703, 2.8716, 2.8500] \cdot 10^{-4} \\ \mathbf{r}_1^T &= [-1.1089, -1.5115, -2.1101, -2.1917, -2.5270, -2.7301] \cdot 10^{-2} \end{aligned} \tag{32}$$

and $T_{D_m} = 5$.

After estimation of parameters, the TS fuzzy model Equation (31) was transformed into the state space form to simplify the procedure of obtaining the control law:

$$\mathbf{x}_m(k+1) = \sum_i \beta_i(\mathbf{x}_p(k)) (\mathbf{A}_{mi} \mathbf{x}_m(k) + \mathbf{B}_{mi} u(k - T_{D_m}) + \mathbf{R}_{mi}), \quad (33)$$

$$y_m(k) = \sum_i \beta_i(\mathbf{x}_p(k)) \mathbf{C}_m \mathbf{x}_m(k), \quad (34)$$

$$\mathbf{A}_{mi} = \begin{bmatrix} 0 & 1 \\ -a_{2i} & -a_{1i} \end{bmatrix} \quad \mathbf{B}_{mi} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \mathbf{C}_m = [b_{1i} \ 0] \quad \mathbf{R}_{mi} = \begin{bmatrix} 0 \\ r_i/b_i \end{bmatrix}, \quad (35)$$

where the process measured output concentration C_A is denoted by y_m and the input flow q_c by u .

The frozen-time theory (Leith and Leithead, 1998, 1999) enables the relation between the nonlinear dynamical system and the associated linear time-varying system. The theory establishes the following fuzzy model

$$\begin{aligned} \mathbf{x}_m(k+1) &= \tilde{\mathbf{A}}_m \mathbf{x}_m(k) + \tilde{\mathbf{B}}_m u(k - T_{D_m}) + \tilde{\mathbf{R}}_m, \\ y_m(k) &= \tilde{\mathbf{C}}_m \mathbf{x}_m(k), \end{aligned} \quad (36)$$

where $\tilde{\mathbf{A}}_m = \sum_i \beta_i(\mathbf{x}_p(k)) \mathbf{A}_{mi}$, $\tilde{\mathbf{B}}_m = \sum_i \beta_i(\mathbf{x}_p(k)) \mathbf{B}_{mi}$, $\tilde{\mathbf{C}}_m = \sum_i \beta_i(\mathbf{x}_p(k)) \mathbf{C}_{mi}$ and $\tilde{\mathbf{R}}_m = \sum_i \beta_i(\mathbf{x}_p(k)) \mathbf{R}_{mi}$.

6.3.3.2. Simulation results

Reference tracking ability and disturbance rejection capability of the FMBPC control algorithm were tested experimentally on a simulated CSTR plant. The FMBPC was compared to the conventional PI controller.

In the first experiment, the control system was tested for tracking the reference signal that changed the operating point from nominal ($C_A = 0.1$ mol/l) to larger concentration values and back, and then to smaller concentration values and back. The proposed FMBPC used the following design parameters: $H = 9$ and $a_r = 0.96$. The parameters of the PI controller were obtained by minimising the following criterium:

$$C_{PI} = \sum_{k=0}^{400} (y_r(k) - y_{PI}(k))^2, \quad (37)$$

where $y_r(k)$ is the reference model output depicted in Figure 6.2, and $y_{PI}(k)$ is the controlled output in the case of PI control. This means that the parameters of the PI controller were minimized to obtain the best tracking of the reference model output for the case treated in the first experiment. The optimal parameters were $K_p = 64.6454$ l²mol

min^{-1} and $T_i = 0.6721$ min. Figure 6.2 also shows manipulated and controlled variables for the two approaches. In the lower part of the figure, the set-point is depicted with the dashed line, the reference model output with the dotted line, the FMBPC response with the thick solid line and the PI response with the thin solid line. The upper part of the figure represents the two control signals. The performance criteria obtained in the experiment are the following:

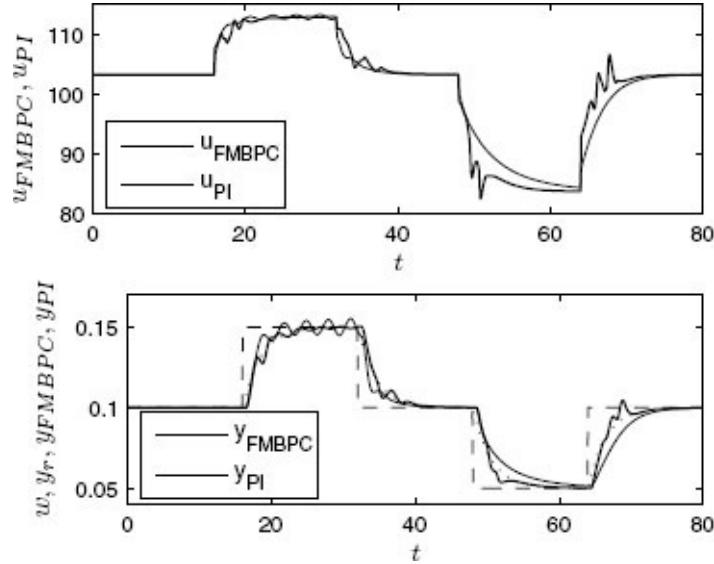


Figure 6.2: The performance of the FMBPC and the PI control in the case of reference trajectory tracking.

$$C_{PI} = \sum_{k=0}^{400} (y_r(k) - y_{PI}(k))^2 = 0.0165, \quad (38)$$

$$C_{FMBPC} = \sum_{k=0}^{400} (y_r(k) - y_{FMBPC}(k))^2 = 0.0061. \quad (39)$$

The disturbance rejection performance was tested with the same controllers that were set to the same design parameters as in the first experiment. In the simulation experiment, the step-like positive input disturbance of 3 l/min appeared and disappeared later. After some time, the step-like negative input disturbance of -3 l/min appeared and disappeared later. The results of the experiment are shown in Figure 6.3 where the signals are depicted by the same line types as in Figure 6.2. Similar performance criteria can be calculated as in the case of reference tracking:

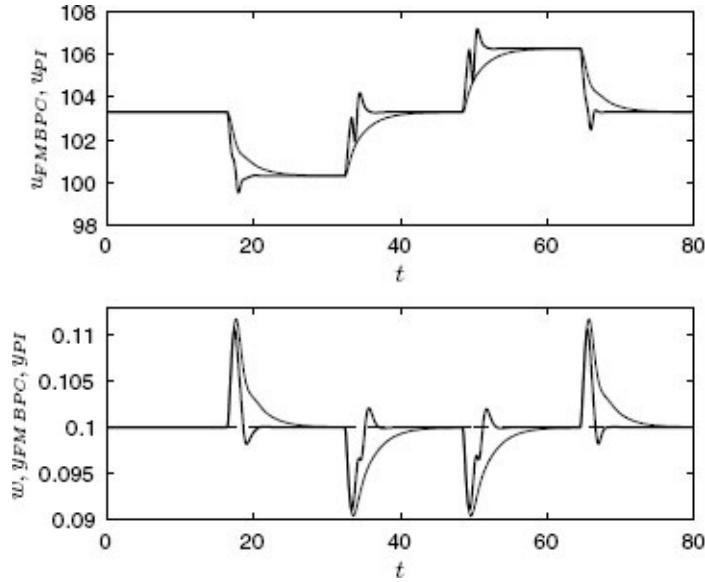


Figure 6.3: The control performance of the FMBPC and the PI control in the case of disturbance rejection.

$$C_{PI} = \sum_{k=0}^{400} (y_r(k) - y_{PI}(k))^2 = 0.0076, \quad (40)$$

$$C_{FMBPC} = \sum_{k=0}^{400} (y_r(k) - y_{FMBPC}(k))^2 = 0.0036. \quad (41)$$

The obtained simulation results have shown that better performance criteria are obtained in the case of the FMBPC control in both control modes: The trajectory tracking mode and the disturbance rejection mode. This is obvious because the PI controller assumes linear process dynamics, while the FMBPC controller takes into account the plant nonlinearity through the fuzzy model of the plant. The proposed approach is very easy to implement and gives a high control performance.

6.4. Direct Fuzzy Model Reference Adaptive Control

We have already established that the fuzzy controllers are capable of controlling nonlinear plants. If the model of the plant is not only nonlinear but also unknown or poorly known, the solution becomes considerably more difficult. Nevertheless, several approaches exist to solve the problem. One possibility is to apply adaptive control. Adaptive control schemes for linear systems do not produce good results, although adaptive parameters try to track the “true” local linear parameters of the current operating point which is done with some lag after each operating-point change. To overcome this problem, adaptive control was extended in the 1980s and 1990s to time-varying and nonlinear plants (Krstić *et al.*, 1995).

It is also possible to introduce some sort of adaptation into the fuzzy controller. The first attempts at constructing a fuzzy adaptive controller can be traced back to Procyk and Mamdani (1979), where the so-called linguistic self-organizing controllers were introduced. Many approaches were later presented where a fuzzy model of the plant was constructed online, followed by control parameters adjustment (Layne and Passino, 1993). The main drawback of these schemes was that their stability was not treated rigorously. The universal approximation theorem (Wang and Mendel, 1992) provided a theoretical background for new fuzzy controllers (Pomares *et al.*, 2002; Precup and Preitl, 2006; Tang *et al.*, 1999; Wang and Mendel, 1992) whose stability was treated rigorously.

Robust adaptive control was proposed to overcome the problem of disturbances and unmodeled dynamics (Ioannou and Sun, 1996). Similar solutions have also been used in adaptive fuzzy and neural controllers, i.e., projection (Tong *et al.*, 2000), dead zone (Koo, 2001), leakage (Ge and Wang, 2002), adaptive fuzzy backstepping control (Tong and Li, 2012), etc. have been included in the adaptive law to prevent instability due to reconstruction error.

The control of a practically very important class of plants is treated in this section that, in our opinion, occurs quite often in process industries. The class of plants consists of nonlinear systems of arbitrary order but where the control law is based on the first-order nonlinear approximation. The dynamics not included in the first-order approximation are referred to as parasitic dynamics. The parasitic dynamics are treated explicitly in the development of the adaptive law to prevent the modeling error to grow unbounded. The class of plant also includes bounded disturbances.

The choice of simple nominal model results in very simple control and adaptive laws. The control law is similar to the one proposed by Blažič *et al.* (2003, 2012), but an extra term is added in this work where an adaptive law with leakage is presented (Blažič *et al.*, 2013). It will be shown that the proposed adaptive law is a natural way to cope with parasitic dynamics. The boundedness of estimated parameters, the tracking error and all the signals in the system will be proven if the leakage parameter σ' satisfies certain condition. This means that the proposed adaptive law ensures the global stability of the

system. A very important property of the proposed approach is that it can be used in the consequent part of Takagi–Sugeno-based control. The approach enables easy implementation in the control systems with evolving antecedent part (Angelov *et al.*, 2001; Angelov and Filev, 2004; Angelov *et al.*, 2011; Cara *et al.*, 2010; Sadeghi–Tehran *et al.*, 2012). This combination results in a high-performance and robust control of nonlinear and slowly varying systems.

6.4.1. The Class of Nonlinear Plants

Our goal is to design control for a class of plants that include nonlinear time-invariant systems where the model behaves similarly to a first-order system at low frequencies (the frequency response is not defined for nonlinear systems so frequencies are meant here in a broader sense). If the plant were the first-order system (without parasitic dynamics), it could be described by a fuzzy model in the form of if-then rules:

$$\begin{aligned} &\text{if } z_1 \text{ is } A_{i_a} \text{ and } z_2 \text{ is } B_{i_b} \text{ then } \dot{y}_p = -a_i y_p + b_i u + c_i \\ &i_a = 1, \dots, n_a \quad i_b = 1, \dots, n_b \quad i = 1, \dots, k, \end{aligned} \quad (42)$$

where u and y_p are the input and the output of the plant respectively, A_{i_a} and B_{i_b} are fuzzy membership functions, and a_i , b_i , and c_i are the plant parameters in the i th domain. Note the c_i term in the consequent. Such an additive term is obtained if a nonlinear system is linearized in an operating point. This additive term changes by changing the operating point. The term c_i is new comparing to the model used in Blažič *et al.* (2003, 2012). The antecedent variables that define the domain in which the system is currently situated are denoted by z_1 and z_2 (actually there can be only one such variable or there can also be more of them, but this does not affect the approach described here). There are n_a and n_b membership functions for the first and the second antecedent variables, respectively. The product $k = n_a \times n_b$ defines the number of fuzzy rules. The membership functions have to cover the whole operating area of the system. The output of the Takagi–Sugeno model is then given by the following equation

$$\dot{y}_p = \frac{\sum_{i=1}^k [\beta_i^0(\mathbf{x}_p)(-a_i y_p + b_i u + c_i)]}{\sum_{i=1}^k \beta_i^0(\mathbf{x}_p)}, \quad (43)$$

where \mathbf{x}_p represents the vector of antecedent variables z_i (in the case of fuzzy model given by Equation (42), $\mathbf{x}_p = [z_1 \ z_2]^T$). The degree of fulfilment $\beta_i^0(\mathbf{x}_p)$ is obtained using the T-norm, which in this case is a simple algebraic product of membership functions

$$\beta_i^0(\mathbf{x}_p) = T(\mu_{A_{i_a}}(z_1), \mu_{B_{i_b}}(z_2)) = \mu_{A_{i_a}}(z_1) \cdot \mu_{B_{i_b}}(z_2), \quad (44)$$

where $\mu_{A_{i_a}}(z_1)$ and $\mu_{B_{i_b}}(z_2)$ stand for degrees of fulfilment of the corresponding fuzzy rule. The degrees of fulfilment for the whole set of fuzzy rules can be written in a compact form

as

$$\beta^0 = [\beta_1^0 \ \beta_2^0 \ \dots \ \beta_k^0]^T \in \mathbb{R}^k \quad (45)$$

or in a more convenient normalized form

$$\beta = \frac{\beta^0}{\sum_{i=1}^k \beta_i^0} \in \mathbb{R}^k. \quad (46)$$

Due to Equations (43) and (46), the first-order plant can be modeled in fuzzy form as

$$\dot{y}_p = -(\beta^T \mathbf{a}) y_p + (\beta^T \mathbf{b}) u + (\beta^T \mathbf{c}), \quad (47)$$

where $\mathbf{a} = [a_1 \ a_2 \ \dots \ a_k]^T$, $\mathbf{b} = [b_1 \ b_2 \ \dots \ b_k]^T$, and $\mathbf{c} = [c_1 \ c_2 \ \dots \ c_k]^T$ are vectors of unknown plant parameters in respective domains ($\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^k$).

To assume that the controlled system is of the first order is a quite huge idealization. Parasitic dynamics and disturbances are therefore included in the model of the plant. The fuzzy model of the first order is generalized by adding stable factor plant perturbations and disturbances, which results in the following model (Blažič *et al.*, 2003):

$$\begin{aligned} \dot{y}_p(t) &= -(\beta^T(t) \mathbf{a}) y_p(t) + (\beta^T(t) \mathbf{b}) u(t) + (\beta^T \mathbf{c}) \\ &\quad - \Delta_y(p) y_p(t) + \Delta_u(p) u(t) + d(t), \end{aligned} \quad (48)$$

where p is a differential operator d/dt , $\Delta_y(p)$ and $\Delta_u(p)$ are stable strictly proper linear operators, while d is bounded signal due to disturbances (Blažič *et al.*, 2003).

Equation (48) represents the class of plants to be controlled by the approach proposed in the following sections. The control is designed based on the model given by Equation (47) while the robustness properties of the algorithm prevent the instability due to parasitic dynamics and disturbances.

6.4.2. The Proposed Fuzzy Adaptive Control Algorithm

A fuzzy model reference adaptive control is proposed to achieve tracking control for the class of plants described in the previous section. The control goal is that the plant output follows the output y_m of the reference model. The latter is defined by a first order linear system $G_m(p)$:

$$y_m(t) = G_m(p) w(t) = \frac{b_m}{p + a_m} w(t), \quad (49)$$

where $w(t)$ is the reference signal while b_m and a_m are the constants that define desired behavior of the closed system. The tracking error

$$\varepsilon(t) = y_p(t) - y_m(t), \quad (50)$$

therefore represents some measure of the control quality. To solve the control problem simple control and adaptive laws are proposed in the following sub-sections.

6.4.2.1. Control law

The control law is very similar to the one proposed by Blažič *et al.* (2003, 2012):

$$u(t) = \left(\boldsymbol{\beta}^T(t) \hat{\mathbf{f}}(t) \right) w(t) - \left(\boldsymbol{\beta}^T(t) \hat{\mathbf{q}}(t) \right) y_p(t) + \left(\boldsymbol{\beta}^T(t) \hat{\mathbf{r}}(t) \right), \quad (51)$$

where $\hat{\mathbf{f}}(t) \in \mathbb{R}^k$, $\hat{\mathbf{q}}(t) \in \mathbb{R}^k$, and $\hat{\mathbf{r}}(t) \in \mathbb{R}^k$ are the control gain vectors to be determined by the adaptive law. This control law is obtained by generalizing the model reference adaptive control algorithm for the first-order linear plant to the fuzzy case. The control law also includes the third term that is new with respect to the one in Blažič *et al.* (2012). It is used to compensate the $(\boldsymbol{\beta}^T \mathbf{c})$ term in Equation (48).

6.4.2.2. Adaptive law

The adaptive law proposed in this chapter is based on the adaptive law from Blažič *et al.* (2003). The e_1 -modification was used in the leakage term in Blažič *et al.* (2012). An alternative approach was proposed in Blažič *et al.* (2012) where quadratic term is used the leakage. But a new adaptive law for \hat{r}_i is also proposed here:

$$\begin{aligned} \dot{\hat{f}}_i &= -\gamma_{fi} b_{\text{sign}} \varepsilon w \beta_i - \gamma_{fi} \sigma' w^2 \beta_i^2 (\hat{f}_i - \hat{f}_i^*) \quad i = 1, 2, \dots, k, \\ \dot{\hat{q}}_i &= \gamma_{qi} b_{\text{sign}} \varepsilon y_p \beta_i - \gamma_{qi} \sigma' y_p^2 \beta_i^2 (\hat{q}_i - \hat{q}_i^*) \quad i = 1, 2, \dots, k, \\ \dot{\hat{r}}_i &= -\gamma_{ri} b_{\text{sign}} \varepsilon \beta_i - \gamma_{ri} \sigma' \beta_i^2 (\hat{r}_i - \hat{r}_i^*) \quad i = 1, 2, \dots, k, \end{aligned} \quad (52)$$

where γ_{fi} , γ_{qi} , and γ_{ri} are positive scalars referred to as adaptive gains, $\sigma' > 0$ is the parameter of the leakage term, \hat{f}_i^* , \hat{q}_i^* , and \hat{r}_i^* are the *a priori* estimates of the control gains \hat{f}_i , \hat{q}_i , and \hat{r}_i respectively, and b_{sign} is defined as follows:

$$b_{\text{sign}} = \begin{cases} 1 & b_1 > 0, b_2 > 0, \dots, b_k > 0 \\ -1 & b_1 < 0, b_2 < 0, \dots, b_k < 0 \end{cases}. \quad (53)$$

If the signs of all elements in vector \mathbf{b} are not the same, the plant is not controllable for some $\boldsymbol{\beta}$ ($\boldsymbol{\beta}^T \mathbf{b}$ is equal to 0 for this $\boldsymbol{\beta}$) and any control signal does not have an effect.

It is possible to rewrite the adaptive law, Equation (52) in the compact form if the control gain vectors $\hat{\mathbf{f}}$, $\hat{\mathbf{q}}$, and $\hat{\mathbf{r}}$ are defined as

$$\begin{aligned} \hat{\mathbf{f}}^T &= [\hat{f}_1 \ \hat{f}_2 \ \dots \ \hat{f}_k], \\ \hat{\mathbf{q}}^T &= [\hat{q}_1 \ \hat{q}_2 \ \dots \ \hat{q}_k], \\ \hat{\mathbf{r}}^T &= [\hat{r}_1 \ \hat{r}_2 \ \dots \ \hat{r}_k]. \end{aligned} \quad (54)$$

Then the adaptive law, Equation (52), takes the following form:

$$\begin{aligned}\dot{\hat{\mathbf{f}}} &= -\Gamma_f b_{\text{sign}} \varepsilon w \beta - \Gamma_f \sigma' w^2 \text{diag}(\beta) \text{diag}(\beta) (\hat{\mathbf{f}} - \hat{\mathbf{f}}^*), \\ \dot{\hat{\mathbf{q}}} &= \Gamma_q b_{\text{sign}} \varepsilon y_p \beta - \Gamma_q \sigma' y_p^2 \text{diag}(\beta) \text{diag}(\beta) (\hat{\mathbf{q}} - \hat{\mathbf{q}}^*), \\ \dot{\hat{\mathbf{r}}} &= -\Gamma_r b_{\text{sign}} \varepsilon \beta - \Gamma_r \sigma' \text{diag}(\beta) \text{diag}(\beta) (\hat{\mathbf{r}} - \hat{\mathbf{r}}^*),\end{aligned}\quad (55)$$

where $\Gamma_f \in \mathbb{R}^{k \times k}$, $\Gamma_q \in \mathbb{R}^{k \times k}$, and $\Gamma_r \in \mathbb{R}^{k \times k}$ are positive definite matrices, $\text{diag}(\mathbf{x}) \in \mathbb{R}^{k \times k}$ is a diagonal matrix with the elements of vector \mathbf{x} on the main diagonal, while $\hat{\mathbf{f}}^* \in \mathbb{R}^k$, $\hat{\mathbf{q}}^* \in \mathbb{R}^k$, and $\hat{\mathbf{r}}^* \in \mathbb{R}^k$ are the *a priori* estimates of the control gain vectors.

6.4.2.3. The sketch of the stability proof

The reference model Equation (49) can be rewritten in the following form:

$$\dot{y}_m = -a_m y_m + b_m w. \quad (56)$$

By subtracting Equation (56) from Equation (48), the following tracking-error model is obtained

$$\begin{aligned}\dot{\varepsilon} &= -a_m \varepsilon + \left[(\beta^T \mathbf{b})(\beta^T \hat{\mathbf{f}}) - b_m \right] w - \left[(\beta^T \mathbf{b})(\beta^T \hat{\mathbf{q}}) + (\beta^T \mathbf{a}) - a_m \right] y_p \\ &\quad + \left[(\beta^T \mathbf{b})(\beta^T \hat{\mathbf{r}}) + (\beta^T \mathbf{c}) \right] + \Delta_u(p) u - \Delta_y(p) y_p + d.\end{aligned}\quad (57)$$

Now we assume that there exist constant control parameters \mathbf{f}^* , \mathbf{q}^* , and \mathbf{r}^* that stabilize the closed-loop system. This is a mild assumption and it is always fulfilled unless the unmodeled dynamics are unacceptably high. These parameters are only needed in the stability analysis and can be chosen to make the “difference” between the closed-loop system and the reference model small in some sense (the definition of this “difference” is not important for the analysis). The parameters \mathbf{f}^* , \mathbf{q}^* , and \mathbf{r}^* are sometimes called the “true” parameters because they result in the perfect tracking in the absence of unmodeled dynamics and disturbances. The parameter errors are

defined as:

$$\begin{aligned}\tilde{\mathbf{f}} &= \hat{\mathbf{f}} - \mathbf{f}^*, \\ \tilde{\mathbf{q}} &= \hat{\mathbf{q}} - \mathbf{q}^*, \\ \tilde{\mathbf{r}} &= \hat{\mathbf{r}} - \mathbf{r}^*.\end{aligned}\quad (58)$$

The expressions in the square brackets in Equation (57) can be rewritten similarly as in Blažič *et al.* (2003):

$$\begin{aligned}
[(\beta^T \mathbf{b})(\beta^T \hat{\mathbf{f}}) - b_m] &= b_{\text{sign}} \beta^T \tilde{\mathbf{f}} + \eta_f = b_{\text{sign}} \sum_{i=1}^k \beta_i \tilde{f}_i + \eta_f, \\
[(\beta^T \mathbf{b})(\beta^T \hat{\mathbf{q}}) + (\beta^T \mathbf{a}) - a_m] &= b_{\text{sign}} \beta^T \tilde{\mathbf{q}} + \eta_q = b_{\text{sign}} \sum_{i=1}^k \beta_i \tilde{q}_i + \eta_q, \\
[(\beta^T \mathbf{b})(\beta^T \hat{\mathbf{r}}) + (\beta^T \mathbf{c})] &= b_{\text{sign}} \beta^T \tilde{\mathbf{r}} + \eta_r = b_{\text{sign}} \sum_{i=1}^k \beta_i \tilde{r}_i + \eta_r,
\end{aligned} \tag{59}$$

where bounded residuals $\eta_f(t)$, $\eta_q(t)$, and $\eta_r(t)$ are introduced [the boundedness can be shown simply; see also (Blažič *et al.*, 2003)]. The following Lyapunov function is proposed for the proof of stability:

$$V = \frac{1}{2} \varepsilon^2 + \frac{1}{2} \sum_{i=1}^k \gamma_{fi}^{-1} \tilde{f}_i^2 + \frac{1}{2} \sum_{i=1}^k \gamma_{qi}^{-1} \tilde{q}_i^2 + \frac{1}{2} \sum_{i=1}^k \gamma_{ri}^{-1} \tilde{r}_i^2. \tag{60}$$

Calculating the derivative of the Lyapunov function along the solution of the system, Equation (57) and taking into account Equation (59) and adaptive laws, Equation (52), we obtain:

$$\begin{aligned}
\dot{V} &= \varepsilon \dot{\varepsilon} + \sum_{i=1}^k \gamma_{fi}^{-1} \tilde{f}_i \dot{\tilde{f}}_i + \sum_{i=1}^k \gamma_{qi}^{-1} \tilde{q}_i \dot{\tilde{q}}_i + \sum_{i=1}^k \gamma_{ri}^{-1} \tilde{r}_i \dot{\tilde{r}}_i \\
&= -a_m \varepsilon^2 + \eta_f w \varepsilon - \eta_q y_p \varepsilon + \eta_r \varepsilon + \varepsilon \Delta_u(p) u - \varepsilon \Delta_y(p) y_p + \varepsilon d \\
&\quad - \sum_{i=1}^k \sigma' w^2 \beta_i^2 (\hat{f}_i - \hat{f}_i^*) \tilde{f}_i - \sum_{i=1}^k \sigma' y_p^2 \beta_i^2 (\hat{q}_i - \hat{q}_i^*) \tilde{q}_i - \sum_{i=1}^k \sigma' \beta_i^2 (\hat{r}_i - \hat{r}_i^*) \tilde{r}_i.
\end{aligned} \tag{61}$$

In principle, the first term on the right-hand side of Equation (61) is used to compensate for the next six terms while the last three terms prevent parameter drift. The terms from the second one to the seventh one are formed as a product between the tracking error $\varepsilon(t)$ and a combined error $E(t)$ defined as:

$$E(t) = \eta_f(t) w(t) - \eta_q(t) y_p(t) + \eta_r(t) + \Delta_u(p) u(t) - \Delta_y(p) y_p(t) + d(t). \tag{62}$$

Equation (61) can be rewritten as:

$$\begin{aligned}
\dot{V} &= -a_m \left(\varepsilon^2 - \frac{E \varepsilon}{a_m} \right) \\
&\quad - \sum_{i=1}^k \sigma' w^2 \beta_i^2 (\hat{f}_i - \hat{f}_i^*) \tilde{f}_i - \sum_{i=1}^k \sigma' y_p^2 \beta_i^2 (\hat{q}_i - \hat{q}_i^*) \tilde{q}_i - \sum_{i=1}^k \sigma' \beta_i^2 (\hat{r}_i - \hat{r}_i^*) \tilde{r}_i.
\end{aligned} \tag{63}$$

The first term on the right-hand side of Equation (63) becomes negative if $|\varepsilon| > \frac{|E|}{a_m}$. If the combined error, were *a priori* bounded, the boundedness of the tracking error ε would be more or less proven. The problem lies in the fact that not only bounded signals ($w(t)$, $\eta_f(t)$, $\eta_q(t)$, $\eta_r(t)$, $d(t)$) are included in $E(t)$, but also the ones whose boundedness is yet to be

proven ($u(t)$, $y_p(t)$). If the system becomes unstable, the plant output $y_p(t)$ becomes unbounded and, consequently, the same applies to the control input $u(t)$. If $y_p(t)$ is bounded, it is easy to see from the control law that $u(t)$ is also bounded. Unboundedness of $y_p(t)$ is prevented by leakage terms in the adaptive law. In the last three terms in Equation (63) that are due to the leakage there are three similar expressions. They have the following form:

$$(\hat{f}_i(t) - \hat{f}_i^*)\tilde{f}_i(t) = (\hat{f}_i(t) - \hat{f}_i^*)(\hat{f}_i(t) - f_i^*). \quad (64)$$

It is simple to see that this expression is positive if either $\hat{f}_i > \max\{\hat{f}_i^*, f_i^*\}$ or $\hat{f}_i < \min\{\hat{f}_i^*, f_i^*\}$. The same reasoning applies to \hat{q}_i and \hat{r}_i . This means that the last three terms in Equation (63) become negative if the estimated parameters are large (or small) enough. The novelty of the proposed adaptive law with respect to the one in Blažič *et al.* (2003), is in the quadratic terms with y_p and w in the leakage. These terms are used to help cancelling the contribution of εE in Equation (63):

$$\varepsilon E = \varepsilon \eta_f w - \varepsilon \eta_q y_p + \varepsilon \eta_r + \varepsilon \Delta_u(p)u - \varepsilon \Delta_y(p)y_p + \varepsilon d. \quad (65)$$

Since $\varepsilon(t)$ is the difference between $y_p(t)$ and $y_m(t)$ and the latter is bounded, $\varepsilon = O(y_p)$ when y_p tends to infinity. By analyzing the control law and taking into account stability of parasitic dynamics $\Delta_u(s)$ and $\Delta_y(s)$, the following can be concluded:

$$u = O(y_p), \Delta_u(p)u = O(y_p) \Rightarrow \varepsilon E = O(y_p^2). \quad (66)$$

The third term on the right-hand side of Equation (63) is $-(\hat{q}_i - \hat{q}_i^*)\tilde{q}_i O(y_p^2)$ which means that the “gain” $(\hat{q}_i - \hat{q}_i^*)\tilde{q}_i$ with respect to y_p^2 of the negative contributions to \dot{V} can always become greater (as a result of adaptation) than the fixed gain of quadratic terms with y_p in Equation (65). The growth of the estimated parameters is also problematic because these parameters are control gains and high gains can induce instability in combination with parasitic dynamics. Consequently, σ' has to be chosen large enough to prevent this type of instability. Note that the stabilization in the presence of parasitic dynamics is achieved without using an explicit dynamic normalization that was used in Blažič *et al.* (2003).

The stability analysis of a similar adaptive law for linear systems was treated in Blažič *et al.* (2010) where it was proven that all the signals in the system are bounded and the tracking error converges to a residual set whose size depends on the modeling error if the leakage parameter σ' is chosen large enough with respect to the norm of parasitic dynamics. In the approach proposed in this chapter, the “modeling error” is $E(t)$ from Equation (62) and therefore the residual-set size depends on the size of the norm of the transfer functions $\|\Delta_u\|$ and $\|\Delta_y\|$, the size of the disturbance d , and the size of the bounded residuals $\eta_f(t)$, $\eta_q(t)$, and $\eta_r(t)$.

Only the adaptation of the consequent part of the fuzzy rules is treated in this chapter.

The stability of the system is guaranteed for any (fixed) shape of the membership functions in the antecedent part. This means that this approach is very easy to combine with existing evolving approaches for the antecedent part. If the membership functions are slowly evolving, these changes introduce another term to \dot{V} which can be shown not to be larger than $O(y_p^2)$. This means that the system stability is preserved by the robustness properties of the adaptive laws. If, however, fast changes of the membership functions occur, a rigorous stability analysis would have to be performed.

6.4.3. Simulation Example—Three-Tank System

A simulation example will be given that illustrates the proposed approach. A simulated plant was chosen since it is easier to make the same operating conditions than it would be when testing on a real plant. The simulated test plant consisted of three water tanks. The schematic representation of the plant is given in Figure 6.4. The control objective was to maintain the water level in the third tank by changing the inflow into the first tank.

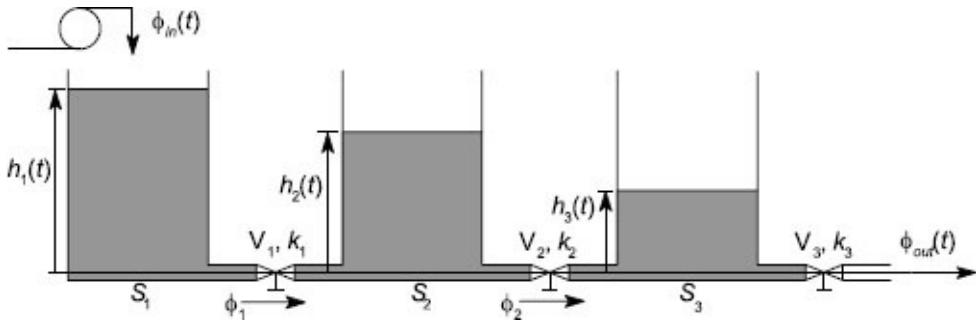


Figure 6.4: Schematic representation of the plant.

When modeling the plant, it was assumed that the flow through the valve was proportional to the square root of the pressure difference on the valve. The mass conservation equations for the three tanks are:

$$\begin{aligned} S_1 \dot{h}_1 &= \phi_{in} - k_1 \operatorname{sign}(h_1 - h_2) \sqrt{|h_1 - h_2|}, \\ S_2 \dot{h}_2 &= k_1 \operatorname{sign}(h_1 - h_2) \sqrt{|h_1 - h_2|} - k_2 \operatorname{sign}(h_2 - h_3) \sqrt{|h_2 - h_3|}, \\ S_3 \dot{h}_3 &= k_2 \operatorname{sign}(h_2 - h_3) \sqrt{|h_2 - h_3|} - k_3 \operatorname{sign}(h_3) \sqrt{|h_3|}, \end{aligned} \quad (67)$$

where ϕ_{in} is the volume inflow into the first tank, h_1 , h_2 , and h_3 are the water levels in three tanks, S_1 , S_2 , and S_3 are areas of the tanks cross-sections, and k_1 , k_2 , and k_3 are coefficients of the valves. The following values were chosen for the parameters of the system:

$$\begin{aligned} S_1 = S_2 = S_3 &= 2 \cdot 10^{-2} \text{m}^2 \\ k_1 = k_2 = k_3 &= 2 \cdot 10^{-4} \text{m}^{5/2} \text{s}^{-1}. \end{aligned} \quad (68)$$

The nominal value of inflow ϕ_{in} was set to $8 \cdot 10^{-5} \text{m}^3 \text{s}^{-1}$, resulting in steady-state values 0.48 m, 0.32 m, and 0.16 m for h_1 , h_2 , and h_3 , respectively. In the following, u and y_p denote deviations of ϕ_{in} and h_3 respectively from the operating point.

By analyzing the plant, it can be seen that the plant is nonlinear. It has to be pointed out that the parasitic dynamics are also nonlinear, not just the dominant part as was assumed in deriving the control algorithm. This means that this example will also test the ability of the proposed control to cope with nonlinear parasitic dynamics. The coefficients of the linearized system in different operating points depend on u , h_1 , h_2 , and h_3 even though that only y_p will be used as an antecedent variable z_1 which is again violation of the basic assumptions but still produces fairly good results.

The proposed control algorithm was compared to a classical model reference adaptive control (MRAC) with e_1 -modification. Adaptive gains γ_f , γ_q , and γ_r in the case of the proposed approach were the same as γ_f , γ_q , and γ_r , respectively, in the case of MRAC. A reference signal was chosen as a periodic piece-wise constant function which covered quite a wide area around the operating point ($\pm 50\%$ of the nominal value). There were 11 triangular fuzzy membership functions (the fuzzification variable was y_p) used; these were distributed evenly across the interval $[-0.1, 0.1]$. As already said, the evolving of the antecedent part was not done in this work. The control input signal u was saturated at the interval $[-8 \cdot 10^{-5}, 8 \cdot 10^{-5}]$. No prior knowledge of the estimated parameters was available to us, so the initial parameter estimates were 0 for all examples.

The design objective is that the output of the plant follows the output of the reference model $0.01/(s + 0.01)$. The reference signal was the same in all cases.

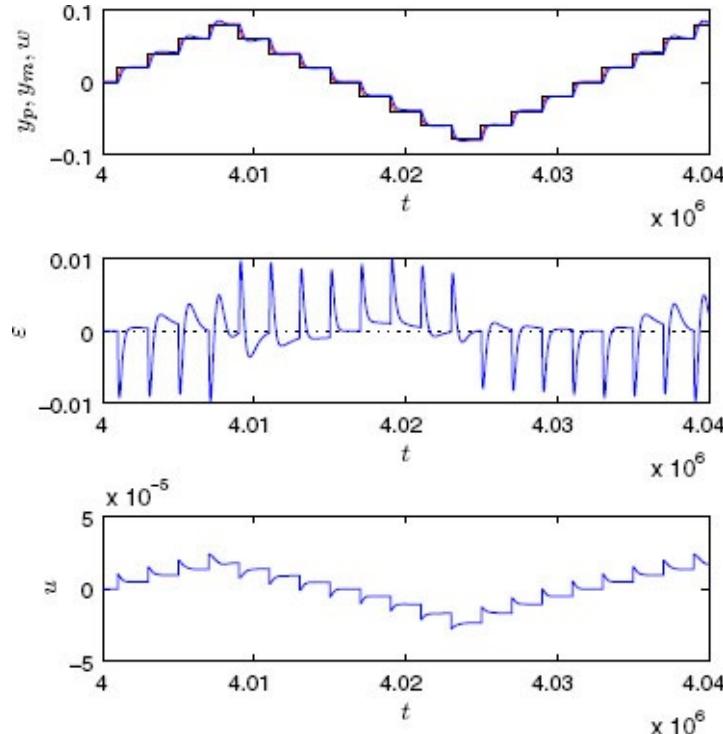


Figure 6.5: The MRAC controller—time plots of the reference signal and outputs of the plant and the reference model (upper figure), time plot of tracking error (middle figure), and time plot of the control signal (lower figure).

It consisted of a periodic signal. The results of the experiment with the classical MRAC controller with e_1 -modification are shown in [Figure 6.5](#).

We used the following design parameters: $\gamma_f = 10^{-4}$, $\gamma_q = 2 \cdot 10^{-4}$, $\gamma_r = 10^{-6}$, $\sigma' = 0.1$.

[Figures 6.6](#) and [6.7](#) show the results of the proposed approach, the former shows a period of system responses after the adaptation has settled, the latter depicts time plots of the estimated parameters. Since $\hat{\mathbf{f}}$, $\hat{\mathbf{q}}$, and $\hat{\mathbf{p}}$ are vectors, all elements of the vectors are depicted. Note that every change in the reference signal results in a sudden increase in tracking error ε (up to 0.01). This is due to the fact that zero tracking of the reference model with relative degree 1 is not possible if the plant has relative degree 3.

The experiments show that the performance of the proposed approach is better than the performance of the MRAC controller for linear plant which is expectable due to nonlinearity of the plant. Very good results are obtained in the case of the proposed approach even though that the parasitic dynamics are nonlinear and linearized parameters depend not only on the antecedent variable y_p but also on others. The spikes on ε in [Figure 6.6](#) are consequences of the fact that the plant of ‘relative degree’ 3 is forced to follow the reference model of relative degree 1. These spikes are inevitable no matter which controller is used.

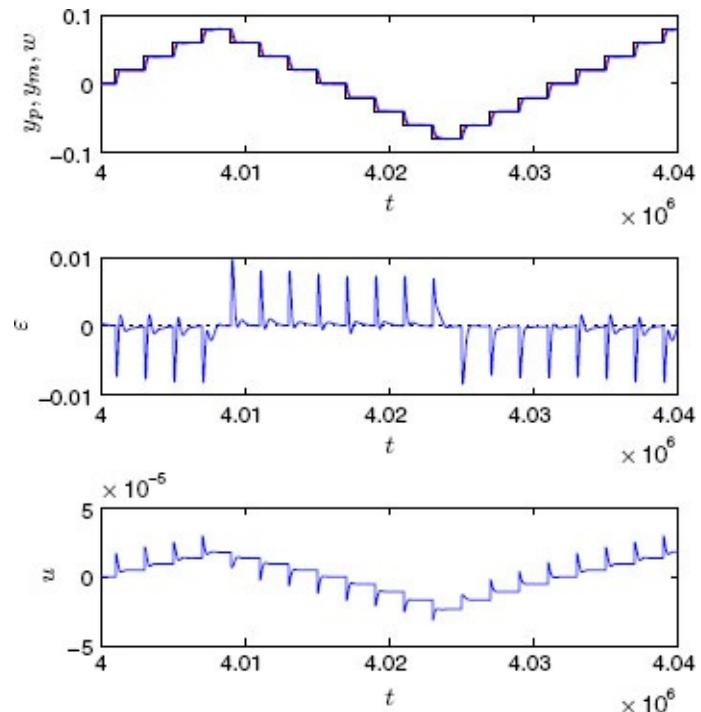


Figure 6.6: The proposed approach—time plots of the reference signal and outputs of the plant and the reference model (upper figure), time plot of tracking error (middle figure), and time plot of the control signal (lower figure).

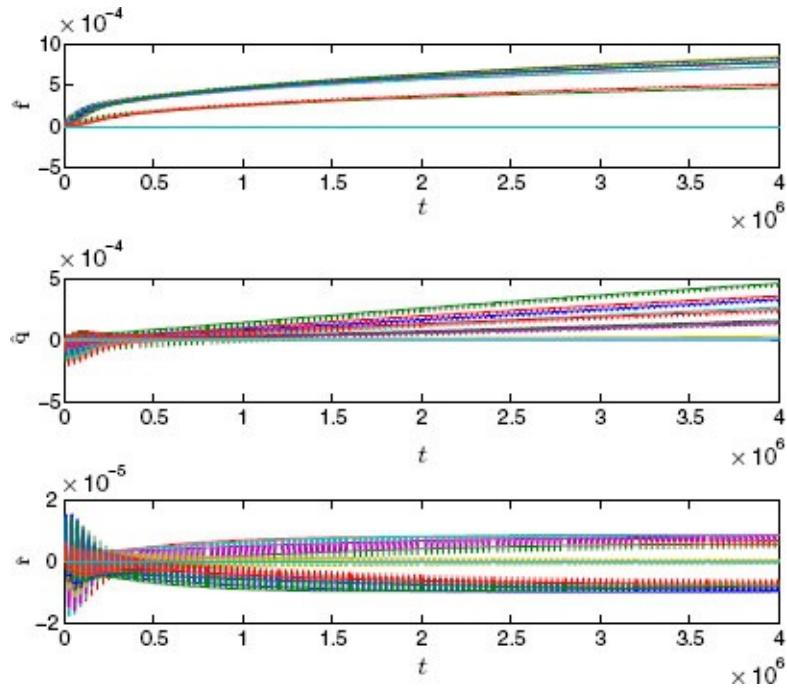


Figure 6.7: The proposed approach—time plots of the control gains.

The drawback of the proposed approach is relatively slow convergence since the parameters are only adapted when the corresponding membership is non-zero. This drawback can be overcome by using classical MRAC in the beginning when there are no parameter estimates or the estimates are bad. When the system approaches desired behavior, the adaptation can switch to the proposed one by initializing all elements of vectors \hat{t} , \hat{q} , and \hat{r} with estimated scalar parameters from the classical MRAC.

6.5. Conclusion

This chapter presents two approaches to the control of nonlinear systems. We chose these two solutions because they are easy to tune and easy to implement on the one hand, but they also guarantee the stability under some assumptions on the other. Both approaches also only deal with the rule consequents and are easy to extend to the variants with evolving antecedent part.

References

- Abonyi, J., Nagy, L. and Szeifert, F. (2001). Fuzzy model-based predictive control by instantaneous linearization. *Fuzzy Sets Syst.*, 120(1), pp. 109–122.
- Andone, D. and Hossu, A. (2004). Predictive control based on fuzzy model for steam generator. In *Proc. IEEE Int. Conf. Fuzzy Syst.*, Budapest, Hungary, 3, pp. 1245–1250.
- Angelov, P., Buswell, R. A., Wright, J. and Loveday, D. (2001). Evolving rule-based control. In *Proc. of EUNITE Symposium*, pp. 36–41.
- Angelov, P. and Filev, D. P. (2004). An approach to online identification of Takagi–Sugeno fuzzy models. *IEEE Syst. Man Cybern.*, pp. 484–498.
- Angelov, P., Sadeghi–Tehran, P. and Ramezani, R. (2011). An approach to automatic real-time novelty detection, object identification, and tracking in video streams based on recursive density estimation and evolving Takagi–Sugeno fuzzy systems. *Int. J. Intell. Syst.*, 26(3), pp. 189–205.
- Babuska, R. (1998). *Fuzzy Modeling for Control*. Kluwer Academic Publishers.
- Baranyi, P., Tikk, D., Yam, Y. and Patton, R. J. (2003). From differential equations to PDC controller design via numerical transformation. *Comput. Ind.*, 51(3), pp. 281–297.
- Bequette, B. W. (1991). Nonlinear control of chemical processes: A review. *Ind. Eng. Chem. Res.*, 30, pp. 1391–1413.
- Blažič, S. and Škrjanc, I. (2007). Design and stability analysis of fuzzy model-based predictive control—a case study. *J. Intell. Robot. Syst.*, 49(3), pp. 279–292.
- Blažič, S., Škrjanc, I. and Matko, D. (2002). Globally stable model reference adaptive control based on fuzzy description of the plant. *Int. J. Syst. Sci.*, 33(12), pp. 995–1012.
- Blažič, S., Škrjanc, I. and Matko, D. (2003). Globally stable direct fuzzy model reference adaptive control. *Fuzzy Sets Syst.*, 139(1), pp. 3–33.
- Blažič, S., Škrjanc, I. and Matko, D. (2010). Adaptive law with a new leakage term. *IET Control Theory Appl.*, 4(9), pp. 1533–1542.
- Blažič, S., Škrjanc, I. and Matko, D. (2012). A new fuzzy adaptive law with leakage. In *2012 IEEE Conf. Evolving Adapt. Intell. Syst. (EAIS)*. Madrid: IEEE, pp. 47–50.
- Blažič, S., Škrjanc, I. and Matko, D. (2013). A robust fuzzy adaptive law for evolving control systems. *Evolving Syst.*, 5(1), pp. 3–10. doi: 10.1007/s12530-013-9084-7.
- Cara, A. B., Lendek, Z., Babuska, R., Pomares, H. and Rojas, I. (2010). Online self-organizing adaptive fuzzy controller: Application to a nonlinear servo system. In *2010 IEEE Int. Conf. Fuzzy Syst. (FUZZ)*, Barcelona, pp. 1–8. doi: 10.1109/FUZZY.2010.5584027.
- Cervantes, L., Castillo, O. and Melin, P. (2011). Intelligent control of nonlinear dynamic plants using a hierarchical modular approach and type-2 fuzzy logic. In Batyrshin, I. and Sidorov, G. (eds.), *Adv. Soft Comput. Lect. Notes Comput. Sci.*, 7095, pp. 1–12.
- Clarke, D. W., Mohtadi, C. and Tuffs, P. S. (1987). Generalized predictive control—part 1, part 2. *Autom.*, 24, pp. 137–160.
- Cutler, C. R. and Ramaker, B. L. (1980). Dynamic matrix control—a computer control algorithm. In *Proc. ACC*. San Francisco, CA, paper WP5-B.
- De Keyser, R. M. C., Van de Valde, P. G. A. and Dumortier, F. A. G. (1988). A comparative study of self-adaptive long-range predictive control methods. *Autom.*, 24(2), pp. 149–163.
- Doyle, F. J., Ogunnaike, T. A. and Pearson, R. K. (1995). Nonlinear model-based control using second-order volterra models. *Autom.*, 31, pp. 697–714.
- Figueroa, J. L. (2001). Piecewise linear models in model predictive control. *Latin Am. Appl. Res.*, 31(4), pp. 309–315.
- Ge, S. and Wang, J. (2002). Robust adaptive neural control for a class of perturbed strict feedback nonlinear systems. *IEEE Trans. Neural Netw.*, 13(6), pp. 1409–1419.
- Goldberg, R. R. (1976). *Methods of Real Analysis*. New York, USA: John Wiley and Sons.

- Henson, M. A. (1998). Nonlinear model predictive control: current status and future directions. *Comput. Chem. Eng.*, 23, pp. 187–202.
- Henson, M. A. and Seborg, D. E. (1990). Input–output linerization of general processes. *AIChE J.*, 36, p. 1753.
- Ioannou, P. A. and Sun, J. (1996). *Robust Adaptive Control*. Upper Saddle River, New Jersey, USA: Prentice-Hall.
- Johanyák, Z. C. and Papp, O. (2012). A hybrid algorithm for parameter tuning in fuzzy model identification. *Acta Polytech. Hung.*, 9(6), pp. 153–165.
- Kim, J.-H. and Huh, U.-Y. (1998). Fuzzy model, based predictive control. In *Proc. IEEE Int. Conf. Fuzzy Syst.*, Anchorage, AK, pp. 405–409.
- Koo, K.-M. (2001). Stable adaptive fuzzy controller with time varying dead-zone. *Fuzzy Sets Syst.*, 121, pp. 161–168.
- Kosko, B. (1994). Fuzzy systems as universal approximators. *IEEE Trans. Comput.*, 43(11), pp. 1329–1333.
- Krstić, M., Kanellakopoulos, I. and Kokotović, P. (1995). *Nonlinear and Adaptive Control Design*. New York, NY, USA: John Wiley and Sons.
- Layne, J. R. and Passino, K. M. (1993). Fuzzy model reference learning control for cargo ship steering. *IEEE Control Syst. Mag.*, 13, pp. 23–34.
- Leith, D. J. and Leithead, W. E. (1998). Gain-scheduled and nonlinear systems: dynamics analysis by velocity-based linearization families. *Int. J. Control.*, 70(2), pp. 289–317.
- Leith, D. J. and Leithead, W. E. (1999). Analytical framework for blended model systems using local linear models. *Int. J. Control.*, 72(7–8), pp. 605–619.
- Li, N., Li, S. and Xi, Y. (2004). Multi-model predictive control based on the Takagi–Sugeno fuzzy models: a case study. *Inf. Sci. Inf. Comput. Sci.*, 165(3–4), pp. 247–263.
- Lin, C.-H. (1997). Siso nonlinear system identification using a fuzzy-neural hybrid system. *Int. J. Neural Syst.*, 8(3), pp. 325–337.
- Mamdani, E. (1974). Application of fuzzy algorithms for control of simple dynamic plant. *Proc. Inst. Electr. Eng.*, 121(12), pp. 1585–1588.
- Moon, T. K. and Stirling, W. C. (1999). *Mathematical Methods and Algorithms for Signal Processing*. Upper Saddle River, New Jersey, USA: Prentice Hall.
- Morningred, J. D., Paden, B. E. and Mellichamp, D. A. (1992). An adaptive nonlinear predictive controller. *Chem. Eng. Sci.*, 47, pp. 755–762.
- Padin, M. S. and Figueroa, J. L. (2000). Use of cpwl approximations in the design of a numerical nonlinear regulator. *IEEE Trans. Autom. Control*, 45(6), pp. 1175–1180.
- Passino, K. and Yurkovich, S. (1998). *Fuzzy Control*. Addison-Wesley.
- Pedrycz, W. (1993). *Fuzzy Control and Fuzzy Systems*. Taunton, UK: Research Studies Press.
- Perng, J.-W. (2012). Describing function analysis of uncertain fuzzy vehicle control systems. *Neural Comput. Appl.*, 21(3), pp. 555–563.
- Pomares, H., Rojas, I., Gonzlez, J., Rojas, F., Damas, M. and Fernndez, F. J. (2002). A two-stage approach to self-learning direct fuzzy controllers. *Int. J. Approx. Reason.*, 29(3), pp. 267–289.
- Precup, R.-E. and Hellendoorn, H. (2011). A survey on industrial applications of fuzzy control. *Comput. Ind.*, 62(3), pp. 213–226.
- Precup, R.-E. and Preitl, S. (2006). PI and PID controllers tuning for integral-type servo systems to ensure robust stability and controller robustness. *Electr. Eng.*, 88(2), pp. 149–156.
- Precup, R.-E., Tomescu, M. L. and Preitl, S. (2007). Lorenz system stabilization using fuzzy controllers. *Int. J. Comput., Commun. Control*, 2(3), pp. 279–287.
- Procyk, T. J. and Mamdani, E. H. (1979). A linguistic self-organizing process controller. *Autom.*, 15, pp. 15–30.
- Richalet, J. (1993). Industrial application of model based predictive control. *Autom.*, 29(5), pp. 1251–1274.
- Richalet, J., Rault, A., Testud, J. L. and Papon, J. (1978). Model predictive heuristic control: Applications to industrial processes. *Autom.*, 14, pp. 413–428.

- Roubos, J. A., Mollov, S., Babuska, R. and Verbruggen, H. B. (1999). Fuzzy model-based predictive control using takagi-sugeno models. *Int. J. Approx. Reason.*, 22(1–2), pp. 3–30.
- Sadeghi-Tehran, P., Cara, A. B., Angelov, P., Pomares, H., Rojas, I. and Prieto, A. (2012). Self-evolving parameter-free rule-based controller. In *IEEE Proc. 2012 World Congr. Comput. Intell.*, WCCI-2012, pp. 754–761.
- Sun, H.-R., Han, P. and Jiao, S.-M. (2004). A predictive control strategy based on fuzzy system. In *Proc. 2004 IEEE Int. Conf. Inf. Reuse Integr.*, pp. 549–552. doi: 10.1109/1R1.20041431518.
- Takagi, T. and Sugeno, M. (1985). Fuzzy identification of systems and its applications to modelling and control. *IEEE Trans. Syst., Man, Cybern.*, 15, pp. 116–132.
- Tanaka, K. and Wang, H. O. (2002). *Fuzzy Control Systems Design and Analysis: A Linear Matrix Inequality Approach*. New York: John Wiley & Sons Inc.
- Tanaka, K., Ikeda, T. and Wang, H. O. (1996). Robust stabilization of a class of uncertain nonlinear systems via fuzzy control: Quadratic stabilizability, h^∞ control theory, and linear matrix inequalities. *IEEE Trans. Fuzzy Syst.*, 4(1), pp. 1–13.
- Tang, Y., Zhang, N. and Li, Y. (1999). Stable fuzzy adaptive control for a class of nonlinear systems. *Fuzzy Sets Syst.*, 104, pp. 279–288.
- Tong, S. and Li, Y. (2012). Adaptive fuzzy output feedback tracking backstepping control of strict-feedback nonlinear systems with unknown dead zones. *IEEE Trans. Fuzzy Systems*, 20(1), pp. 168–180.
- Tong, S., Wang, T. and Tang, J. T. (2000). Fuzzy adaptive output tracking control of nonlinear systems. *Fuzzy Sets Syst.*, 111, pp. 169–182.
- Vaščák, J. (2012). Adaptation of fuzzy cognitive maps by migration algorithms. *Kybernetes*, 41(3/4), pp. 429–443.
- Škrjanc, I. and Matko, D. (2000). Predictive functional control based on fuzzy model for heat-exchanger pilot plant. *IEEE Trans. Fuzzy Systems*, 8(6), pp. 705–712.
- Wang, H. O., Tanaka, K. and Griffin, M. F. (1996). An approach to fuzzy control of nonlinear systems: Stability and design issues. *IEEE Trans. Fuzzy Syst.*, 4(1), pp. 14–23.
- Wang, L.-X. and Mendel, J. M. (1992). Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. *IEEE Trans. Neural Netw.*, 3(5), pp. 807–814.
- Ydstie, B. E. (1984). Extended horizon adaptive control. In *IFAC World Congr.* Budapest, Hungary, paper 14.4/E4.
- Ying, H. G. (1997). Necessary conditions for some typical fuzzy systems as universal approximators. *Autom.*, 33, pp. 1333–1338.
- Zadeh, L. A. (1973). Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Trans. Syst., Man Cybern.*, SMC-3(1), pp. 28–44.

Chapter 7

Fuzzy Fault Detection and Diagnosis

Bruno Sielly Jales Costa

This chapter presents a thorough review of the literature in the field of fault detection and diagnosis (FDD), focusing, latter, on the strategies and applications based on fuzzy rule-based systems. The presented methods are classified into three main research lines: quantitative model-based, qualitative model-based and process history-based methods, and such division offers the reader an immediate glance about possible directions in the field of study. Introductory concepts and basic applications of each group of techniques are presented in a unified benchmark framework, enabling a fair comparison between the strategies. Many of the traditional and state-of-the-art approaches presented in the literature are referred in this chapter, allowing the reader to have a general overview of possible fault detection and diagnosis strategies.

7.1. Introduction

For four decades, fuzzy systems have been successfully used in a large scope of different industrial applications. Among the different areas of study, it is imperative to mention the work of Kruse *et al.* (1994) in the field of computer science, Pedrycz and Gomide (2007) in the field of industrial engineering, Lughofe (2011) in the field of data stream mining, Abonyi (2003) in the field of process control, Kerre and Nachtegael (2000) in the field of image processing and Nelles (2001) in the field of system identification.

One of the main advantages of a fuzzy system, when compared to other techniques for inaccurate data mining, such as neural networks, is that its knowledge basis, which is composed of inference rules, is very easy to examine and understand (Costa *et al.*, 2012). This format of rules also makes it easy to maintain and update the system structure. Using a fuzzy model to express the behavior of a real system in an understandable manner is a task of great importance, since the main “philosophy of fuzzy sets theory is to serve the bridge between the human understanding and the machine processing” (Casillas *et al.*, 2003). Regarding this matter, interpretability of fuzzy systems was the object of study of Casillas *et al.* (2003), Lughofe (2013), Gacto *et al.* (2011), Zhou and Gan (2008) and others.

Fault Detection and Diagnosis (FDD) is, without a doubt, one of the more beneficial areas for fuzzy theory, among the industrial applications scope (Mendonça *et al.*, 2006; Dash *et al.*, 2003). As concrete examples of applications of fuzzy sets and systems in the context of FDD, one can mention Serdio *et al.* (2014) and Angelov *et al.* (2006), where fuzzy systems are placed among top performers in residual-based data-driven FDD, and Lemos *et al.* (2013) and Laukonen *et al.* (1995), presenting fuzzy approaches to be used in the context of data-stream mining based FDD.

While FDD is still widely performed by human operators, the core of the task consists, roughly, of a sequence of reasoning steps based on the collected data, as we are going to see along this chapter. Fuzzy reasoning can be applied in all steps and in several different ways in the FDD task.

Applications of FDD techniques in industrial environments are increasing in order to improve the operational safety, as well as to reduce the costs related to unscheduled stoppages. The importance of the FDD research in control and automation engineering relies on the fact that prompt detection of an occurring fault, while the system is still operating in a controllable region, usually prevents or, at least, reduces productivity losses and health risks (Venkatasubramanian *et al.*, 2003c).

Many authors have, very recently, contributed to the FDD field of study, with extensive studies, compilations and thorough reviews. Korbicz *et al.* (2004) cover the fundamentals of model-based FDD, being directed toward industrial engineers, scientists and academics, pursuing the reliability and FDD issues of safety-critical industrial

processes. Chiang *et al.* (2001) presents the theoretical background and practical techniques for data-driven process monitoring, which includes many approaches based on principal component analysis, linear discriminant analysis, partial least squares, canonical variate analysis, parameter estimation, observer-based methods, parity relations, causal analysis, expert systems and pattern recognition. Isermann (2009) introduces into the field of FDD systems the methods which have proven their performance in practical applications, including fault detection with signal-based and model-based methods, and fault diagnosis with classification and inference methods, in addition to fault-tolerant control strategies and many practical simulations and experimental results. Witczak (2014) presents a selection of FDD and fault-tolerant control strategies for nonlinear systems, from state estimation up to modern soft computing strategies, including original research results. Last but not the least, Simani *et al.* (2002) focuses on model identification oriented to the analytical approach of FDD, including sample case studies used to illustrate the application of each technique.

With the increasing complexity of the procedures and scope of the industrial activities, the Abnormal Event Management (AEM) is a challenging field of study nowadays. The human operator plays a crucial role in this matter since it has been shown that people responsible for AEM take incorrect decisions many times. Industrial statistics show that 70–90% of the accidents are caused by human errors (Venkatasubramanian *et al.*, 2003c; Wang and Guo, 2013). Moreover, there is much more behind the necessity of an automation of FDD processes; for instance, in several industrial environments, the efforts from the operators for a full coverage supervision of all variables and states of the system are very high, which results in severe costs for the company. Sometimes, a manual supervision is simply infeasible, for instance, in largely distributed systems (Chen *et al.*, 2006).

In this chapter, we present a short review on the FDD process, focusing, later, on the existing fuzzy techniques and applications.

7.2. Detection, Isolation and Identification

First, it is important to address some of the nomenclature and definitions in the field of research. The so-called *fault* is the departure from an operating state with an acceptable range of an observed variable or a calculated parameter associated with a process (Venkatasubramanian *et al.*, 2003c). A fault, hence, can be defined as a symptom (e.g., low flow of a liquid, high temperature on a pump) within the process. On the other hand, the event causing such abnormalities is called *failure*, which is also a synonym for *malfunction* or *root cause*.

In an industrial context, there are several different types of faults that could affect the normal operation of a plant. Among the different groups of malfunctions, one can list the following (Samantaray and Bouamama, 2008):

- Gross parameter changes, also known as parametric faults, refers to “disturbances to the process from independent variables, whose dynamics are not provided with that of the process” (Samantaray and Bouamama, 2008). As examples of parametric faults, one can list a change in the concentration of a reactant, a blockage in a pipeline resulting in a change of the flow coefficient and so on.
- Structural changes refers to equipment failures, which may change the model of the process. An appropriate corrective action to such abnormality would require the extraction of new modeling equations to describe the current faulty status of the process. Examples of structural changes are failure of a controller, a leaking pipe and a stuck valve.
- Faulty sensors and actuators, also known as additive faults (or depending on the model, multiplicative faults), refers to incorrect process inputs and outputs, and could lead the plant variables to beyond acceptable limits. Some examples of abnormalities in the input/output instruments are constant (positive or negative) bias, intermittent disturbances, saturation, out-of-range failure and so on.

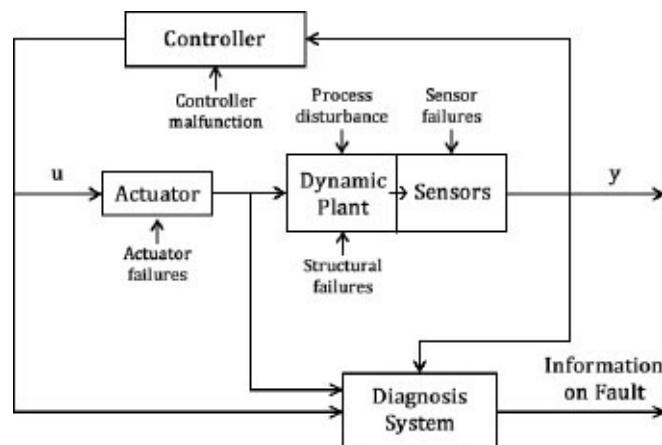


Figure 7.1: General FDD structure (Venkatasubramanian *et al.*, 2003c).

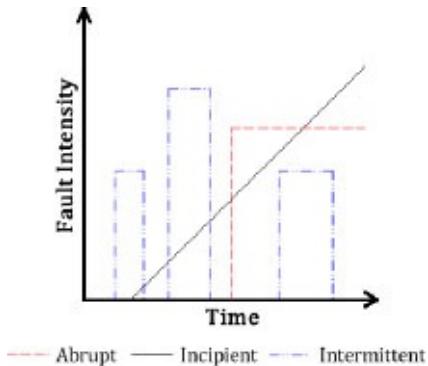


Figure 7.2: Types of faults regarding the time-variant aspect (Patan, 2008).

[Figure 7.1](#) depicts a general process and diagnosis system framework.

Faults can also be classified in a time-variant aspect as

- Abrupt: A fault that abruptly/instantly changes the value of a variable (or a group of variables) from a constant value to another. It is often related to hardware damage.
- Incipient refers to slow parametric changes, where the fault gradually develops to a higher degree. It is usually more difficult to detect due to its slow time characteristics, however, it is less severe than an abrupt fault (Edwards *et al.*, 2010). A good example of an incipient fault is the slow degradation of a hardware component.
- Intermittent: A fault that appears and disappears, repeatedly, over time. A typical example is a partially damaged wiring or a loose connector.

It is important to highlight that a general abnormality is only considered a fault, if it is possible to recover from it, with appropriate control action, either automatic or with the operator intervention. In the past, passive approaches, making use of robust control techniques to ensure that the closed-loop system becomes insensitive to some mild failure situations, were used as a popular fault tolerance strategy (Zhou and Ren, 2001). Nowadays, the active FDD and recovery processes are given as the best solution, once they provide fault accommodation, enabling an update in the control action in order to adjust the controller, in the presence of a fault, to a new given scenario. Fault accommodation, also known as fault compensation, is addressed in Lin and Liu (2007), Efimov *et al.* (2011) and many others.

The entire process of AEM is often divided in a series of steps (usually detection, identification and isolation), which in fault-tolerant design is a fault diagnosis scheme. Although the number of steps may vary from author to author, the general idea remains the same.

The detector system (first stage) continuously monitors the process variables (or attributes) looking for symptoms (deviations on the variables values) and sends these symptoms to the diagnosis system, which is responsible for the classification and identification process.

Fault detection or anomaly detection is the first stage and it has extreme importance to

FDD systems. In this stage, we are able to identify whether the system is working in a normal operating state or in a faulty mode. However, in this stage, vital information about the fault, such as physical location, length or intensity, are not provided to the operator (Silva, 2008).

The diagnosis stage presents its own challenges and obstacles, and can be handled independently from the first one. It demands different techniques and solutions, and can be divided in two sub-stages, called isolation and identification. The term isolation refers to determination of kind, location and time of detection of a fault, and follows the fault detection stage (Donders, 2002). Identification, on the other hand, refers to determination of size and time-variant behavior of a fault, and follows fault isolation.

The diagnosis stage, especially, is a logic decision-making process that generates qualitative data from quantitative data, and it can be seen as a classification problem. The task is to match each pattern of the symptom vector with one of the pre-assigned classes of faults, when existing, and the fault-free case (Frank and Köppen-Seliger, 1997). This process is also known in the literature as fault reasoning.

One last stage related to FDD applications is the task of recovering from an existing and detected fault. The action regarding the process reconfiguration needs to compensate the current malfunction in order to maintain the requirements for an acceptable operating state, when possible, or to determine the further sequence of events (controlled shutdown, for example). Although recovering/accommodation are related to the FDD scheme, we will focus only on the previous described stages.

In general, artificial intelligence-based techniques, such as neural networks, fuzzy systems and expert systems, can be applied in all stages of FDD. In the next sections, we will present some of the widely known approaches based on fuzzy systems.

7.2.1. Quantitative Model-Based FDD Techniques

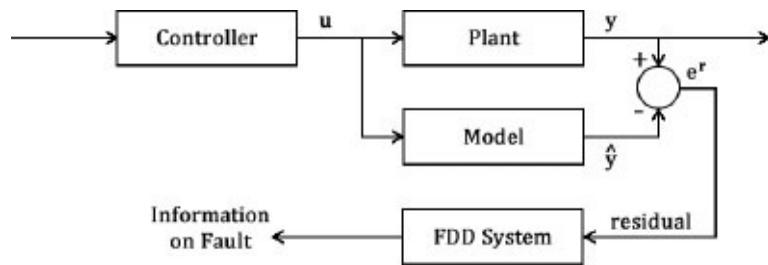
In order to detect and prevent an anomalous state of a process, often, some type of redundancy is necessary. It is used to compare the current and actual state of the process to a state that is expected under those circumstances. Although the redundancy can be provided by extra hardware devices, which is what usually happens in high security processes, analytical redundancy can be used where the redundancy is supplied by a process model instead (Frisk, 2001).

With regard to process models, there are methods that require detailed mathematical models, and there are methods that only require the qualitative description of the model, as we present in the next sub-section.

When the process model is available, the detection of a fault using quantitative model-based techniques depends only on the analysis of the *residual* signal. The residual (e^r) is the difference between the current output (y) of the system and the estimated output (\hat{y})

based on the given model. In general, the residual is expected to be “null” or “nearly-null”, when in a fault-free state, and considerably different from zero, in the presence of a fault. It should be noted that the design of an FDD system must consider the particularities of a real process (e.g., environmental noise, model uncertainties), which can slightly deviate the residual from zero and still not relate to a fault event.

Mathematical models can be available both to the normal state of operation of the process and to each previously known faulty state, indicating that model-based FDD systems are able not only to distinguish between fault-free and faulty states (detection), but also to identify different types and locations of faults (diagnosis). [Figure 7.3](#) illustrates the general structure of a quantitative model-based FDD system.



[Figure 7.3:](#) General structure of a quantitative model-based FDD system.

Many approaches to FDD using quantitative model-based methods were investigated in the literature. One can mention Venkatasubramanian *et al.* (2003c) and Isermann (2005) as two of the main references about the topic. In the first one, the authors present a systematic and comparative review of numerous quantitative model-based diagnostic methods from different perspectives, while in the latter one, the author includes a few detailed applications of such methods to a few different real industrial problems. Still regarding residual-based FDD approaches using analytical models, the reading of Chen and Patton (1999) and Simani *et al.* (2002) is highly recommended.

7.2.2. Qualitative Model-Based FDD Techniques

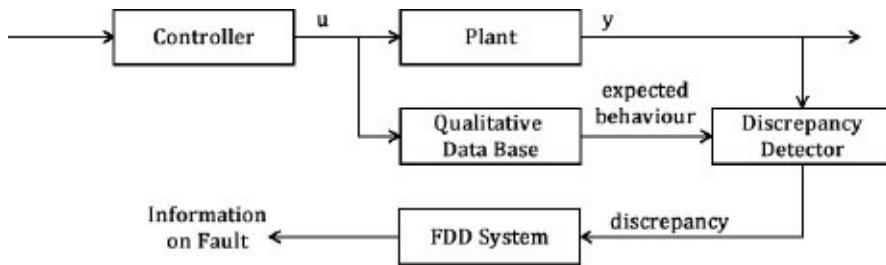
For this group of model-based techniques, the methods are based on the expertise of the operator, qualitative knowledge, and basic understanding about the physics, dynamics, and behavior of the process.

Qualitative models are particularly useful in the sense that even if accurate mathematical models are available for the process, it is often unpractical to obtain all information of the relevant physical parameters of the system, not to mention that external parameters, such as unpredictable disturbances, model uncertainties and so on, are not considered in quantitative models. Hence, FDD methods based on qualitative descriptors are particularly robust (Glass *et al.*, 1995).

Instead of crisp outputs and residual signals, qualitative models work with a qualitative database that feeds a discrepancy detector. The resulting signal, instead of a simple subtraction, is a qualitative discrepancy, based on the expected behavior, for the

given state and the actual output of the system, where the qualitative database is a collection of expert knowledge in form of linguistic descriptions about fault-free and faulty states. [Figure 7.4](#) details the general structure of a qualitative model-based FDD system.

Among the relevant work about the topic, it is imperative to mention Venkatasubramanian *et al.* (2003a). The authors present a complete review of the techniques based on qualitative model representations and search strategies in FDD, highlighting the relative advantages and disadvantages of these methods. Another work that is worth mentioning is Katipamula and Brambley (2005), the first of a two-part review, which summarizes some of the successful qualitative model-based techniques and, although applied exclusively to heating, ventilation and air-conditioning (HVAC) problems, the paper focuses on generic FDD and prognostics, providing a framework for categorizing, describing, and identifying methods, their primary strengths and weaknesses.



[Figure 7.4:](#) General structure of a qualitative model-based FDD system.

7.2.3. Process History-Based FDD Techniques

The third and last large group of methods for FDD refers to a particular type of techniques that is completely data-driven. Process history-based techniques do not require any knowledge, either quantitative or qualitative, about the process. Instead, they use massive historical information collected from the process. This data is, then, transformed and presented as *a priori* information to the FDD system through a process known as *feature extraction*.

Feature extraction (or feature selection) is responsible for reducing the dimensionality of the data, carefully extracting only the relevant information from the input dataset, which usually consists of the measured sensor outputs, namely observable variables (e.g., tank level, pump pressure), or calculated parameters, namely process attributes (e.g., error, pressure oscillation). Statistical methods, expert systems, and neural networks are often used in this type of approach. [Figure 7.5](#) details the general structure of a process history-based FDD system.

As the literature references, one can mention Venkatasubramanian *et al.* (2003b) and Yang *et al.* (2003) as two very important works on the topic. In the first one, the authors present the third part of the literature review, focusing on the process history-based FDD methods. As the last part of the extensive study, the authors suggest that “no single method

has all the desirable features one would like a diagnostic system to possess” and, in order to overcome the limitations of individual solution strategies, the use of hybrid FDD systems is often advised. The latter paper presents a survey on feature extraction, focusing on a variety of validated vibration feature extraction techniques, applied to rotating machinery.

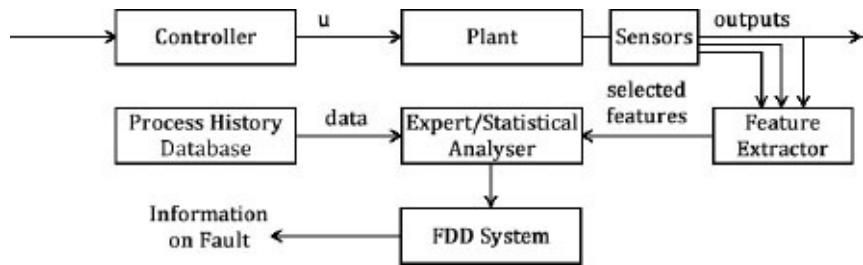


Figure 7.5: General structure of a process history-based FDD system.

7.3. Fuzzy Fault Detection and Identification

Fuzzy rule-based (FRB) systems are currently being investigated in the FDD and reliability research community as a powerful tool for modeling and decision-making (Serdio *et al.*, 2014; Angelov *et al.*, 2006; Lemos *et al.*, 2013; Laukonen *et al.*, 1995), together with neural networks and other more traditional techniques, such as nonlinear and robust observers, parity space methods and so on (Mendonça *et al.*, 2004a). Fuzzy sets theory makes possible the quantification of intrinsically qualitative statements, subjectivity and uncertainty.

The main concepts of fuzzy logic theory make it adequate for FDD. While the nonlinear fuzzy modeling can be very useful in the fault detection work, the transparent and human logic-related inference system is highly suitable for the fault diagnosis stage, which may not only include the expertise from the human operator, but also learn from experimental and/or simulation data. Another benefit of using fuzzy systems in FDD applications is the good performance in reproducing nonlinear mappings, and their abilities of generalization, since fuzzy systems are universal approximators, i.e., being able to model any degree of nonlinearity with an arbitrary desired degree of accuracy (Castro and Delgado, 1996). Hence, fuzzy logic-based systems for fault diagnosis can be advantageous, since they allow the incorporation of prior knowledge and their inference engines are easily understandable to the human operator (Mendonça *et al.*, 2004b).

The process of FDD, especially in its latter stage, can be viewed as a classification problem, which comes with certain particularities, when compared to other groups of applications. When dealing with a classification problem, it is useful to think about the system output as a fuzzy value, instead of a crisp value, skipping the defuzzification step. This way, the output of the FRB system can be presented as a label, which will represent the class of fault assigned to the current state of the process/plant.

Considering an input vector of crisp values $x \in R^n$, composed by the values of the selected process variables/attributes/features, a fuzzy inference rule basis \mathfrak{R} , with R rules, for a generic FDD system can be represented by

$$\begin{aligned} \mathfrak{R}^1 : & \text{ IF } (x_1 \text{ IS } A_{1,1} \text{ AND } x_2 \text{ IS } A_{1,2} \text{ AND } \dots \text{ AND } x_n \text{ IS } A_{1,n}) \text{ THEN} \\ & (y_1 \text{ IS } "Fault 1"), \\ \mathfrak{R}^2 : & \text{ IF } (x_1 \text{ IS } A_{2,1} \text{ AND } x_2 \text{ IS } A_{2,2} \text{ AND } \dots \text{ AND } x_n \text{ IS } A_{2,n}) \text{ THEN} \\ & (y_2 \text{ IS } "Fault 2"), \\ & \vdots \\ \mathfrak{R}^R : & \text{ IF } (x_1 \text{ IS } A_{R,1} \text{ AND } x_2 \text{ IS } A_{R,2} \text{ AND } \dots \text{ AND } x_n \text{ IS } A_{R,n}) \text{ THEN} \\ & (y_R \text{ IS } "Fault R"), \end{aligned}$$

where A is the set of fuzzy values for the input variables and y is the output of the system.

Note that the output y is inferred as the label representing each given class of fault, which can include the nature (e.g., structural, disturbances), the location (e.g., tank 1,

pump, valve A), the type (e.g., leakage, off-set), the degree (e.g., mild, severe) of the fault, as well as can represent the normal state of operation of the plant. Such labels, of course, require linguistic encoding, based on the expert knowledge from the operator. A few unsupervised or semi-supervised approaches (for instance, the one to be presented in [Section 7.3.5](#)) are able to automatically create new rules from the knowledge extracted from data using non-specific labels. These labels, such as “Fault 1” and “Fault A” can, later, be correctly specified by the human operator to include the operation mode/fault class related to that rule.

The inference in a fuzzy rule-based FDD system can be produced using the well-known “winner-takes-it-all” rule (Angelov and Zhou, 2008):

$$L = L^{i*}, \quad i^* = \operatorname{argmax}_{i=1}^R (\gamma^i), \quad (1)$$

where γ^i represents the degree of membership of the input vector \vec{x} to the fuzzy set A^i , considering R inference rules.

As a general example of the application of FRB systems in FDD, we are going to use a benchmark problem, which will be presented and solved with different approaches in the next sub-sections.

7.3.1. Benchmark Problem: First-Order Liquid Level Control in Experimental Tanks

The selected problem is presented and well described in Costa *et al.* (2013) and used in many different applications (Costa *et al.*, 2010, 2012). The referred plant for this study is a two coupled water tanks module, developed by Quanser (2004).

The plant consists of a pump with a water basin. The pump thrusts water vertically to two quick connect, normally closed orifices “Out1” and “Out2”. Two tanks mounted on the front plate are configured such that the flow from the first tank flows into the second tank and outflow from the second tank flows into the main water basin. The graphic representation of the plant is presented in [Figure 7.6](#).

For didactic purposes, in this example, we refer to the simulated version of the referred plant, whose behavior is highly similar to the real version of the same didactic plant, however free of unpredictable environment noise and unrelated disturbances. Although the plant allows second-order control, since it is possible to control the level of the second tank, we will address only first-order aspect of the application, hence, measuring the level on the first tank.

The system consists, then, of two variables: (1) the voltage/control signal (u) applied to the motor/pump — input — which, for safety reasons, is limited to 0–15V DC, and (2) the level (y) of the tank 1 — output — which can vary from 0 to 30cm.

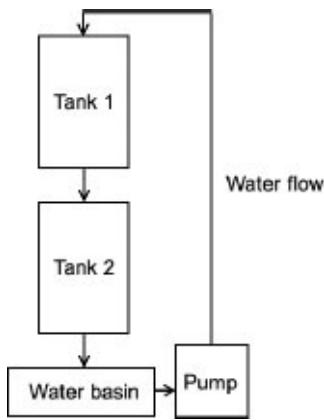


Figure 7.6: Graphic representation of the benchmark plant.

For the control application, which is not in the scope of this chapter, we use a very simple Proportional-Integral-Derivative (PID) controller, where the control signal (u) is calculated by

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t), \quad (2)$$

where K_p is the proportional gain, K_i is the integral gain, K_d is the derivative gain, t is the current time instant and τ is the variable of integration. In this application, the sampling period for the discrete implementation is 1 s. The error, e , is calculated by

$$e = r - y, \quad (3)$$

where r is the reference/set point of the control application.

For all following examples, we will consider $r = 5\text{cm}$, $K_p = 10$, $K_i = 0.1$ and $K_d = 0.1$. The resulting control chart, for a normal state of operation of the plant, is shown in [Figure 7.7](#) and will serve as reference for the fault-free case.

In this example, we will consider a finite set of pre-specified faults, logically generated in the simulation environment. The total of six faults cover different natures, locations, types and degrees of faults that are easily found in common industrial applications, and is more than enough to present a basic review of the different types of fuzzy FDD systems. The set of defined faults is presented in [Table 7.1](#).

The referred faults were independently and sequentially generated within the interval of data samples $k = [500, 800]$. [Figure 7.8](#) presents the control behavior of the system in the presence of the faults (a) F_1 and (b) F_2 (actuator positive off-sets), [Figure 7.9](#) shows the control behavior of the system for the faults (a) F_3 and (b) F_4 (tank leakages) and [Figure 7.10](#) illustrates the control behavior of the system for the faults (a) F_5 and (b) F_6 (actuator saturations).

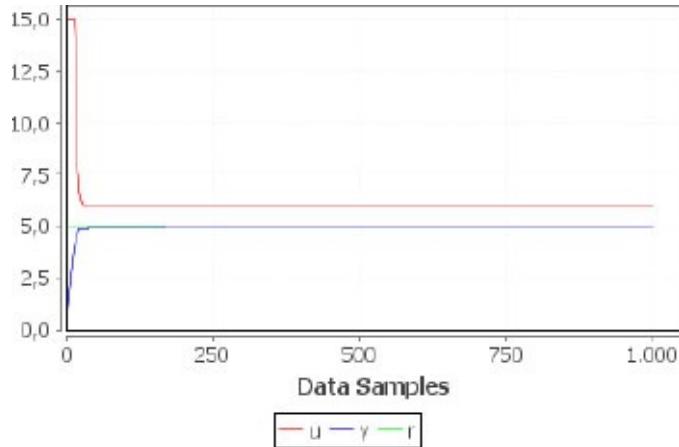


Figure 7.7: Normal state of operation of the plant.

Table 7.1: Pre-defined set of faults.

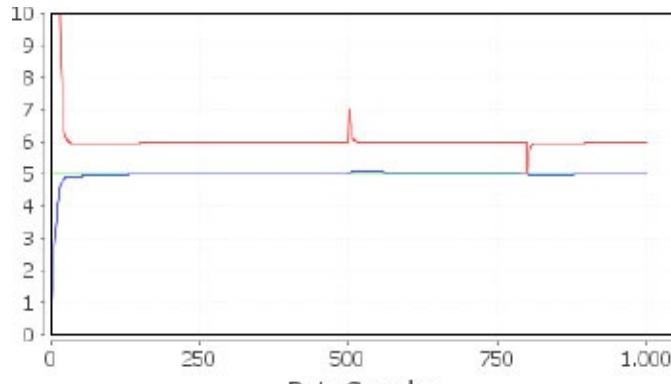
Fault	Nature	Location	Type	Degree
F_1	Actuator	Pump	Off-set	Mild
F_2	Actuator	Pump	Off-set	Severe
F_3	Structural	Tank 1	Leakage	Mild
F_4	Structural	Tank 1	Leakage	Severe
F_5	Actuator	Pump	Saturation	Mild
F_6	Actuator	Pump	Saturation	Severe

In the following sub-sections, we will present two different basic fuzzy-based approaches for the proper detection and classification of the given set of faults and a short review of other applications in the literature.

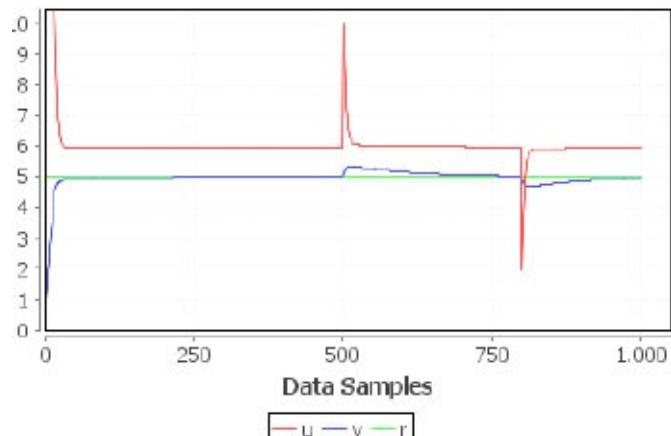
7.3.2. Quantitative Model-Based Fuzzy FDD

In this first example, we develop a fuzzy classifier able to detect and identify different types and levels of faults based on (1) the previous knowledge of the mathematical model of the plant and (2) the expertise of the human operator. This approach was addressed in the literature by Zhao *et al.* (2009), Kulkarni *et al.* (2009), Mendonça *et al.* (2009) and many others.

The mathematical model of the two tanks module, which is necessary for quantitative model-based FDD approaches, is described in Meneghetti (2007). The flow provided by the pump, which is driven by a DC motor, is directly proportional to the voltage applied to the motor. For a first-order configuration, all the water flows to tank 1. This variable is called input flow F_1^{in} and can be calculated by



(a) Mild degree



(b) Severe degree

Figure 7.8: Actuator positive off-set.

$$F_1^{\text{in}} = K_m V_p [\text{cm}^3/\text{s}], \quad (4)$$

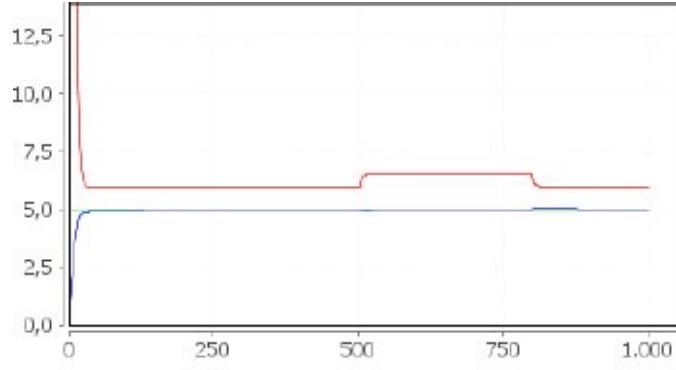
where V_p is the voltage applied to the motor and K_m is the pump constant, which, in this case, is $K_m = 250$.

The speed at which the liquid flows through the output orifice is given by Bernoulli equation for small orifices (Cengel *et al.*, 2012):

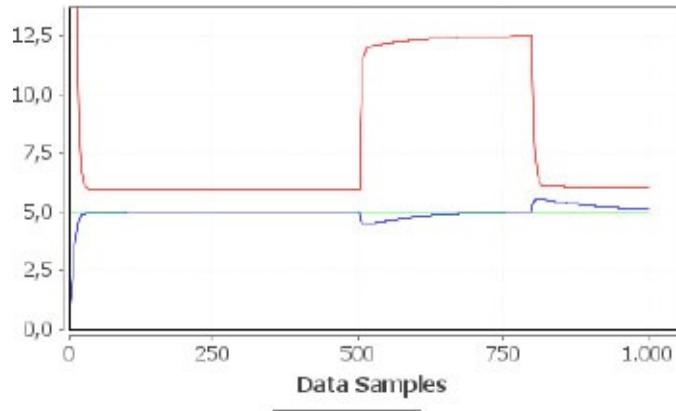
$$v^{\text{out}} = \sqrt{2gL_1} [\text{cm/s}], \quad (5)$$

where g is the gravity acceleration in $[\text{cm/s}^2]$ and L_1 is the water level in the tank 1. The *output flow* F_1^{out} , can be calculated by

$$F_1^{\text{out}} = a_1 v^{\text{out}} [\text{cm}^3/\text{s}], \quad (6)$$



(a) Mild degree



(b) Severe degree

Figure 7.9: Tank leakage.

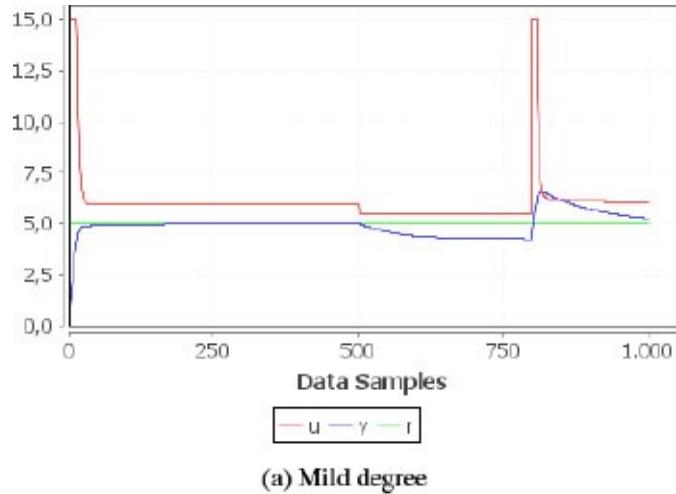
where a_1 is the area of the tank 1 output orifice in cm^2 , which, in this case, is $a_1 = 0.47625 \text{ cm}^2$.

The level variation rate on tank 1 (\dot{L}_1) is given by the ratio between the volumetric variation rate (\dot{V}) and the area of the tank base (A_1) for $\dot{V} = F_1^{\text{in}} - F_1^{\text{out}}$.

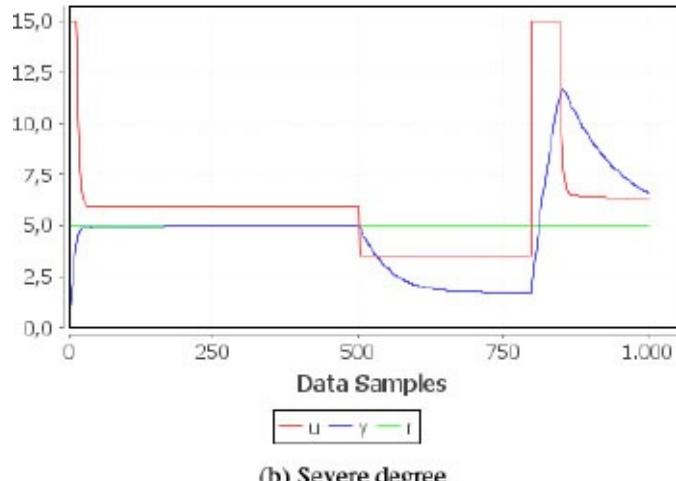
Based on the given model of the plant, one can generate the residual of the output signal by analyzing the discrepancy between the outputs of the model and the actual process as

$$e_y^r = y - \hat{y}, \quad (7)$$

where e_y^r is the residual of the tank level (observable variable), \hat{y} is the model output for the input control signal u , and y is the output measured by the sensor for the same input control signal.



(a) Mild degree

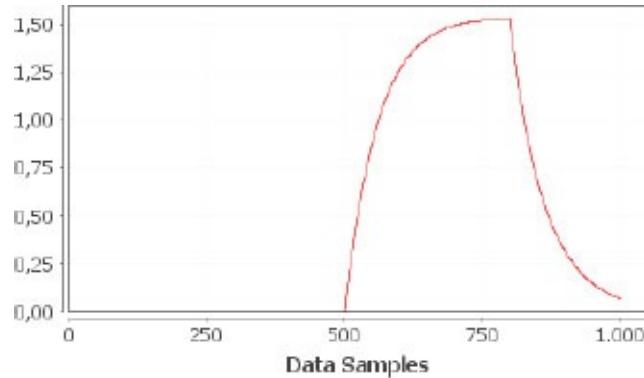


(b) Severe degree

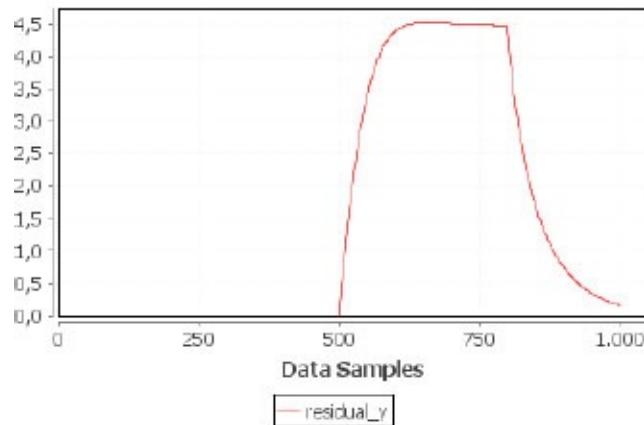
Figure 7.10: Actuator saturation.

It is important to highlight that, in this case, within a normal state of operation of the plant (for $k < 500$ or $k > 800$), Equation (7) will generate a null residual, since we are working with a simulated and disturbance/noise-free application. In real online applications, due to the presence of noise and unpredicted disturbances, the residuals should be handled within a dynamic band or through other tolerance approaches. [Figures 7.11–7.13](#) present the graphical representation of the residual evaluation signal e_y^r for the all previously described faults.

Note that all faults presented here can be classified as abrupt, since they are characterized by the nominal signal changing, abruptly by an unknown positive or negative value. For handling incipient faults, Carl *et al.* (2012) proposes that incipient faults can be approximated by a linear profile because they evolve slowly in time through the estimation of the time of injection and the slope of the signal.



(a) Mild degree

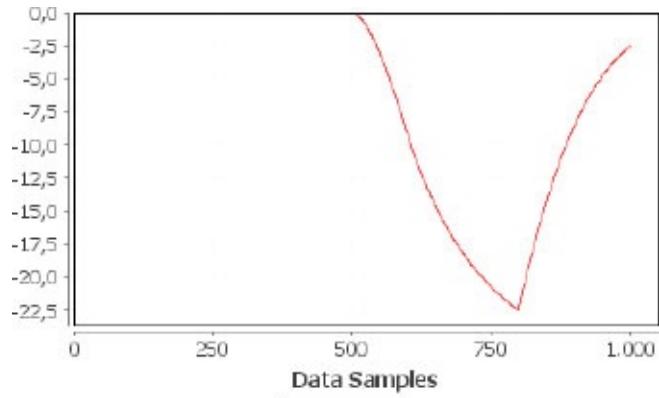


(b) Severe degree

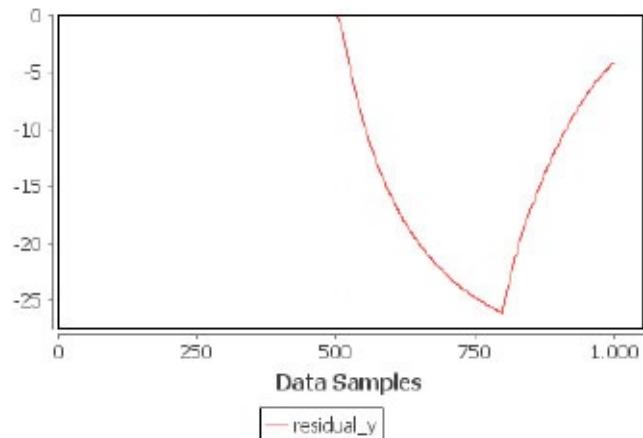
Figure 7.11: Residual evaluation function of the fault “Actuator positive off-set”.

Based on the behavior visualized in the charts, one can make a few assumptions about the fault classification from the generated residuals:

1. The faults F_1 and F_2 , which are positive off-sets of the actuator, can be promptly detected and easily distinguished from the other ones, since the generated residual is positive, while, in the other cases, it is negative.
2. The faults F_3 and F_4 , which are tank leakages, are difficult to distinguish from the faults F_5 and F_6 , which are actuator saturations, since both residuals generated are negative. Although the ranges of the signals are considerably different, only the residual of the output e_y^r may not be enough for an effective classification. The use of additional signals, such as the control signal, might be recommended.



(a) Mild degree

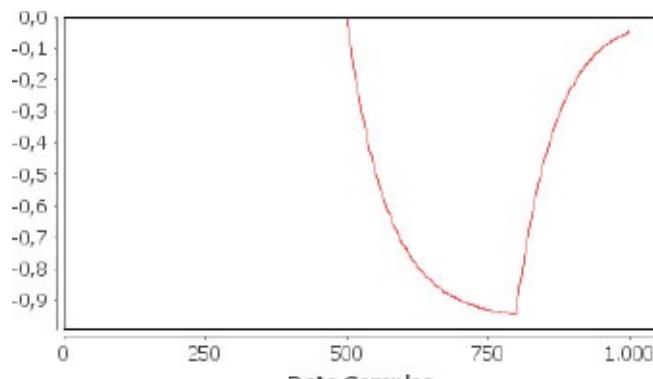


(b) Severe degree

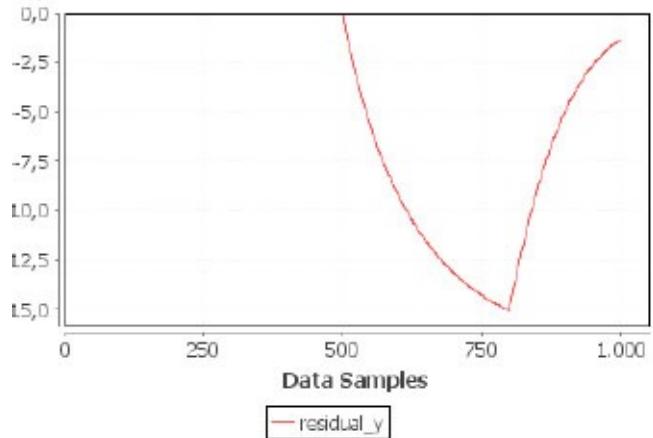
Figure 7.12: Residual evaluation function of the fault “Tank leakage”.

A Mamdani-type FRB system able to address this problem can be composed by one input variable (e_y^r), seven fuzzy sets as membership functions of the input variable (“negative_very_high”, “negative_high”, “negative_low”, “negative_very_low”, “zero”, “positive_low” and “positive_high”), one output variable ($Fault$) and seven fuzzy sets as membership functions of the output variable (F_1 , F_2 , F_3 , F_4 , F_5 , F_6 and “Normal”). One can model the input variable as shown in [Figure 7.14](#) and the output variable as illustrated in [Figure 7.15](#).

For didactic purposes, triangular and trapezoidal membership functions were used to model all fuzzy sets in these examples. Other types of functions are also encouraged (e.g., Gaussian, bell), especially when using automatic/adaptive approaches.



(a) Mild degree



(b) Severe degree

Figure 7.13: Residual evaluation function of the fault “actuator saturation”.

One can, then, propose the following fuzzy rule basis R for the detection and classification of the defined faults:

$\mathfrak{R}^1 : \text{IF } (e_y^r \text{ IS positive}_{\text{low}} \text{ THEN } (y_1 \text{ IS } "F_1 - ActuatorOffset - Mild")),$

$\mathfrak{R}^2 : \text{IF } (e_y^r \text{ IS positive}_{\text{high}} \text{ THEN } (y_2 \text{ IS } "F_2 - ActuatorOffset - Severe")),$

$\mathfrak{R}^3 : \text{IF } (e_y^r \text{ IS negative}_{\text{high}} \text{ THEN } (y_3 \text{ IS } "F_3 - TankLeakage - Mild")),$

$\mathfrak{R}^4 : \text{IF } (e_y^r \text{ IS negative_very_high } \text{ THEN } (y_4 \text{ IS } "F_4 - TankLeakage - Severe")),$

$\mathfrak{R}^5 : \text{IF } (e_y^r \text{ IS negative_very_low } \text{ THEN }$

$(y_5 \text{ IS } "F_5 - ActuatorSaturation - Mild")),$

$\mathfrak{R}^6 : \text{IF } (e_y^r \text{ IS negative}_{\text{low}} \text{ THEN } (y_6 \text{ IS } "F_6 - ActuatorSaturation - Severe")),$

$\mathfrak{R}^7 : \text{IF } (e_y^r \text{ IS zero } \text{ THEN } (y_7 \text{ IS } "Normaloperation")).$

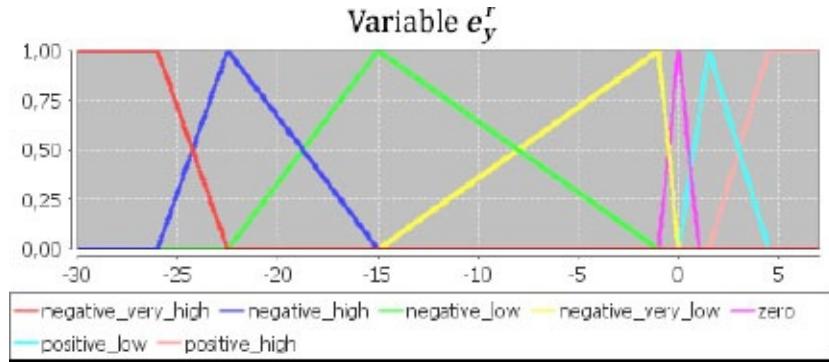


Figure 7.14: Membership functions of the input variable (e_y^r).

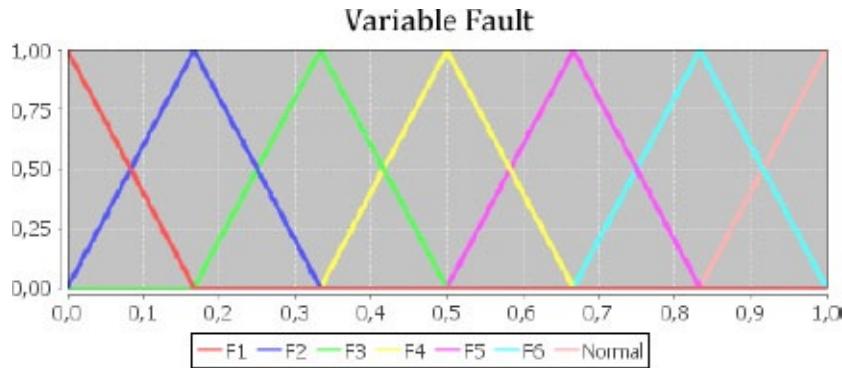


Figure 7.15: Membership functions of the output variable ($Fault$).

Now, for example, let us consider $e_y^r = 25$, at a given time instant, and the “winner-takes-it-all” rule for the output. The whole inference process, for all defined rules, is illustrated in [Figure 7.16](#).

Note that in the left column, two rules (rules 3 and 4) are activated by the input value 25. The membership value of the output of rule 4 is greater than the one for rule 3. It is important to highlight that for the proposed type of fuzzy classifiers, the defuzzification process is ignored. This way, the output of the FRB system is the label “ F_4 ”, which is equivalent to the “Severe Leakage in Tank 1” fault.

7.3.3. Qualitative Model-Based Fuzzy FDD

As discussed in [sub-section 7.2.1](#), although model-based FDD approaches are theoretically efficient, in practice, the dependence on the mathematical model is not appropriate for real applications. Even when this model is given, many times it does not consider variables that are usually present in industrial environments (e.g., process inertia, environment disturbances).

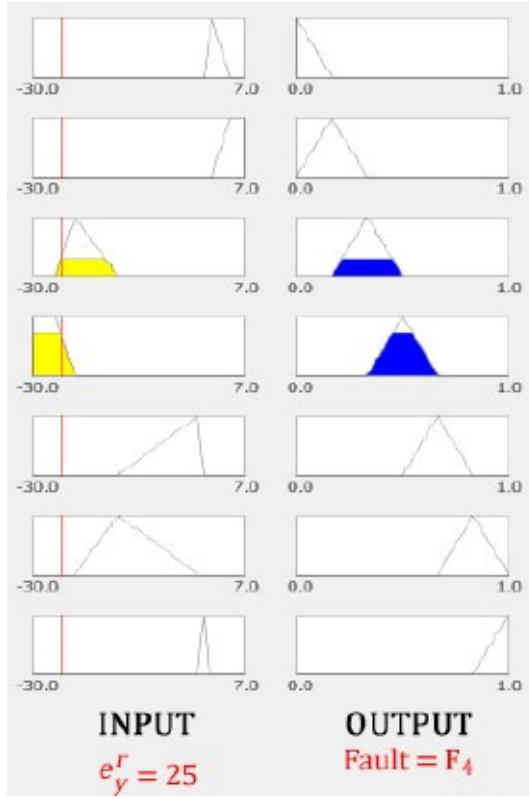


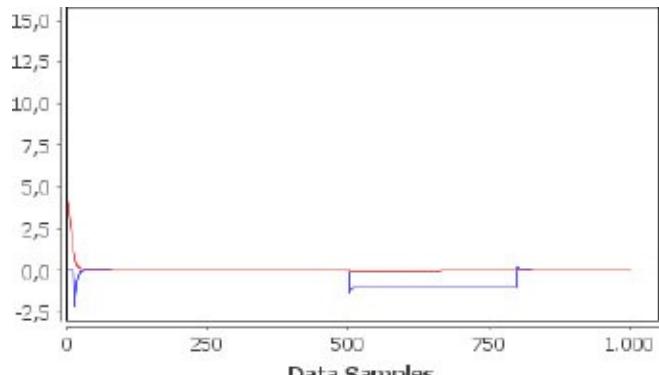
Figure 7.16: Detailed inference process for $e_y^r = 25$.

Starting from a similar point of view, one can consider using an FDD approach that is based on the previous knowledge of the process, however, the mathematical aspect is not necessary. A dynamics-based FDD system is very appropriate when the operator knows well the dynamics of the plant, its physics and behavior. This approach was addressed in the literature by Patton *et al.* (2000), Insfran *et al.* (1999) and many others.

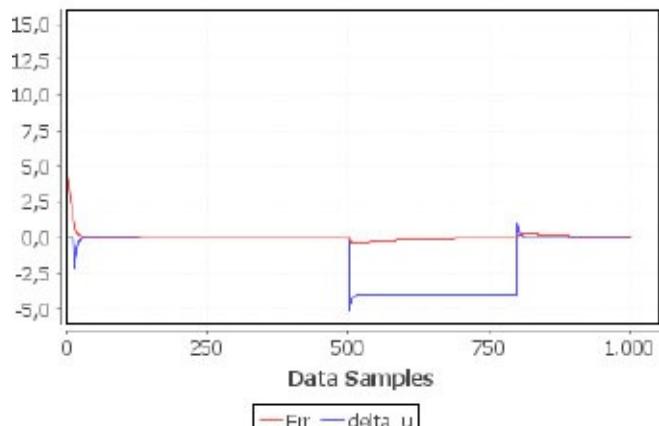
Considering the previous FDD problem, one can make a few observations about the process in [Figure 7.7](#), which illustrates the process in a normal state of operation:

1. In steady state ($k > 180$), the level (y) reaches the set-point (r) value, thus, the error (e) is zero ($e = r - y$).
2. Again, considering the steady state, the control signal (u) becomes constant/non-oscillatory, which means that the control signal variation (Δu) from time step $k - 1$ to time step k is zero ($\Delta u = u_k - u_{k-1}$).
3. Observations (1) and (2) are not true when the system is in a faulty state, as can be seen in [Figures 7.17–7.19](#).

Considering two input variables *Err* (e) and *delta_u* (Δu), four fuzzy sets as membership functions of the variable *Err* (“negative”, “zero”, “positive_low” and “positive_high”), six fuzzy sets as membership functions of the variable *delta_u* (“negative_high”, “negative_low”, “zero”, “positive_low”, “positive_medium” and “positive_high”) and the same output variable (*Fault*) as the one presented in [Figure 7.15](#), one can model the variables of the system as presented in [Figures 7.20](#) and [7.21](#), which represent the input variables *Err* and *delta_u*, respectively.



(a) Mild degree



(b) Severe degree

Figure 7.17: Error and control signal variation on the fault “actuator positive off-set”.

Based on these membership functions, one can, then, propose the following Mamdani-type fuzzy rule basis \mathfrak{R} for the detection and classification of the defined faults:

$\mathfrak{R}^1 : \text{IF } (\text{Err IS zero AND } \text{delta}_u \text{ IS negative_low THEN } (y_1 \text{ IS } "F_1 - ActuatorOffset - Mild"),$

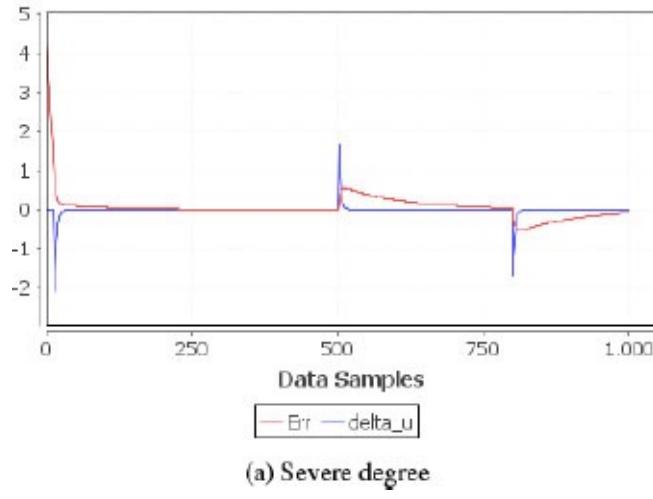
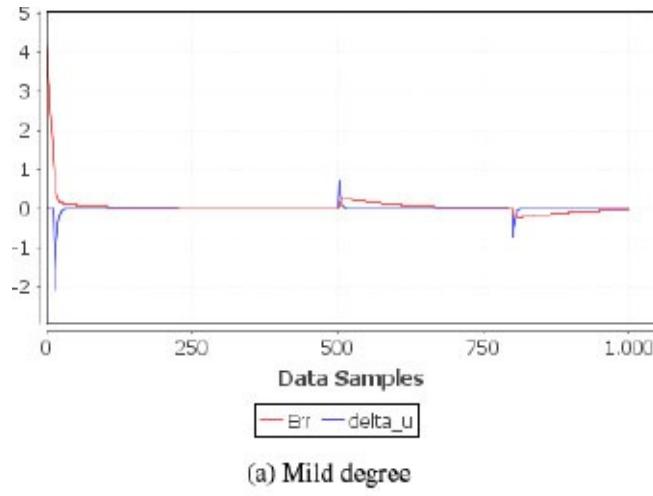


Figure 7.18: Error and control signal variation on the fault “tank leakage”.

\mathcal{R}^2 : **IF** (*Err* **IS** negative **AND** *delta_u* **IS** negative_high **THEN**

(y_2 **IS** “ F_2 – ActuatorOffset – Severe”),

\mathcal{R}^3 : **IF** (*Err* **IS** positive_low **AND** *delta_u* **IS** positive_low **THEN**

(y_3 **IS** “ F_3 – TankLeakage – Mild”),

\mathcal{R}^4 : **IF** (*Err* **IS** positive_low **AND** *delta_u* **IS** positive_medium **THEN**

(y_4 **IS** “ F_4 – TankLeakage – Severe”),

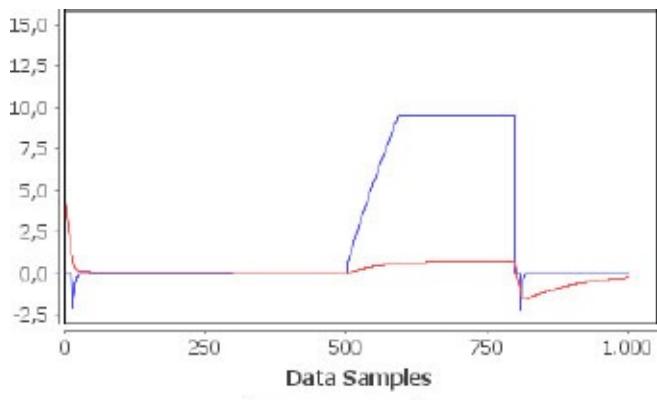
\mathcal{R}^5 : **IF** (*Err* **IS** positive_low **AND** *delta_u* **IS** positive_high **THEN**

(y_4 **IS** “ F_5 – ActuatorSaturation – Mild”),

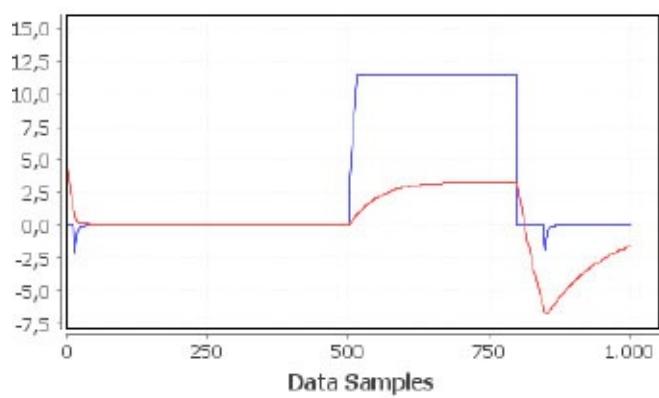
\mathcal{R}^6 : **IF** (*Err* **IS** positive_high **AND** *delta_u* **IS** positive_high **THEN**

(y_4 **IS** “ F_6 – ActuatorSaturation – Severe”),

\mathcal{R}^7 : **IF** (*Err* **IS** zero **AND** *delta_u* **IS** zero **THEN** (y_4 **IS** “Normaloperation”).



(a) Mild degree



(b) Severe degree

Figure 7.19: Error and control signal variation on the fault “actuator saturation”.

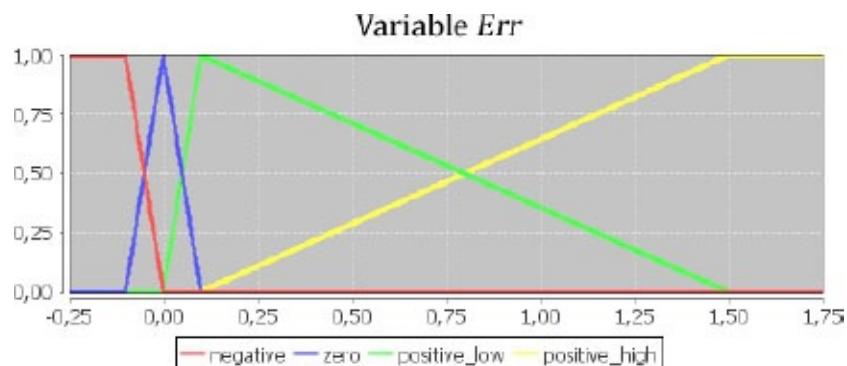


Figure 7.20: Membership functions of the input variable (*Err*).

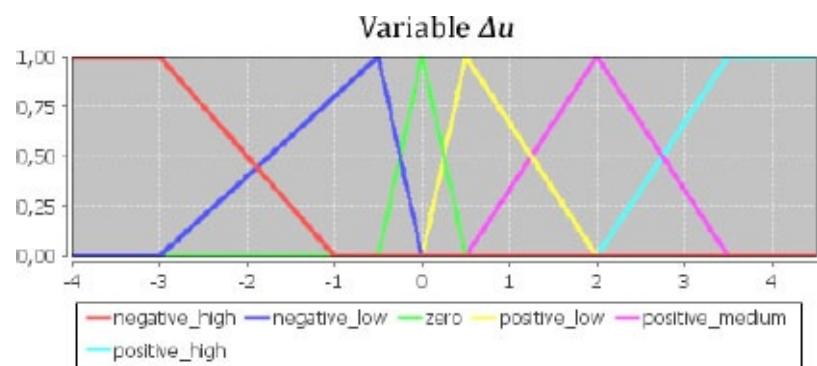


Figure 7.21: Membership functions of the input variable (*delta_u*).

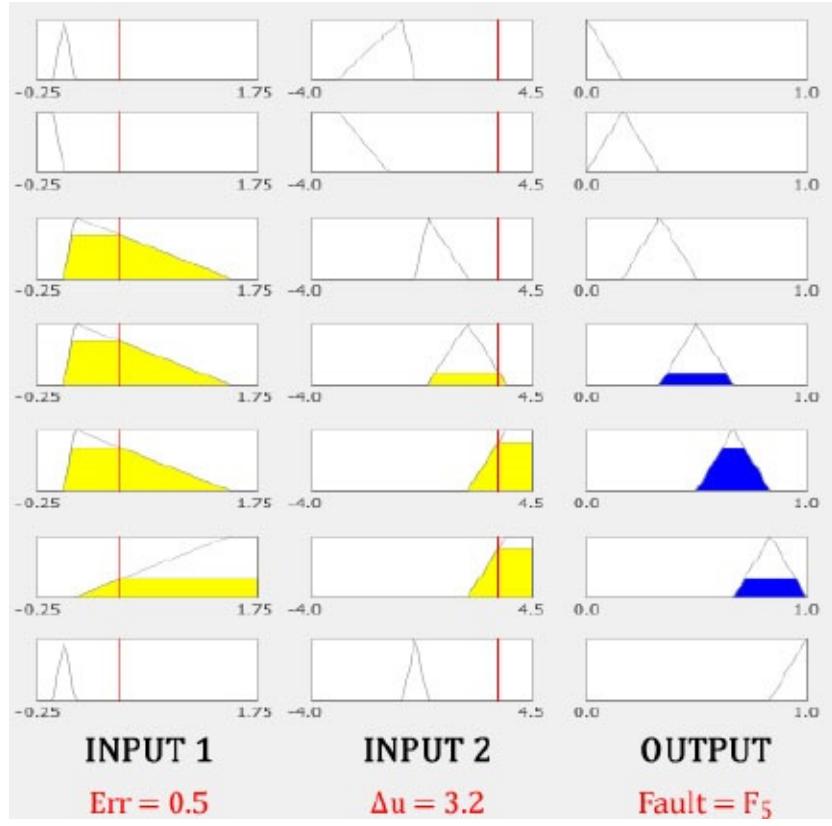


Figure 7.22: Detailed inference process for $Err = 0.5$ and $\Delta u = 3.2$.

As an illustrative example, let us consider $Err = 0.5$ and $\Delta u = 3.2$, at a given time instant, and the “winner-takes-it-all” rule for the output. The detailed inference process, for all defined rules, is illustrated in [Figure 7.22](#).

Note that although the mathematical model of the process, in this particular case, is known, it was not needed in any part of the development of the system. Instead, we used some *a priori* information about the behavior of the system, relating the error and control signals to the normal or faulty states of operation.

7.3.4. Process History-Based Fuzzy FDD

If neither the mathematical and physics/dynamics/behavior models are available, the FDD system can still be based on the data itself. Process history-based FDD systems are commonly used in applications where either there is no prior knowledge available, quantitative or qualitative, or the operator chooses to rely only on the acquired historical process data.

Feature extraction — the main task of process history-based techniques — methods, which are used to transform large amounts of data in prior knowledge about the process, are many times based on the aspects of fuzzy theory. In this sub-section, we are not addressing the particular example presented in sub-section 7.3.1. Instead, we are addressing a few successful approaches of process history-based fuzzy FDD methods.

In Bae *et al.* (2005), a basic data mining application for FDD of induction motors is presented. The method is based on wavelet transform and classification models with

current signals. Through a so-called “fuzzy measure of similarity”, features of faults are extracted, detected and classified.

A multiple signal fault detection system for automotor vehicles is presented in Murphrey *et al.* (2003). The paper describes a system that involves signal segmentation and feature extraction, and uses a fuzzy learning algorithm based on the signals acquired from good functional vehicles only. It employs fuzzy logic at two levels of detection and has been implemented and tested in real automation applications.

In Hu *et al.* (2005), a two-stage fault diagnosis method based on empirical mode decomposition (EMD), fuzzy feature extraction and support vector machines (SVM) is described. In the first stage, intrinsic mode components are obtained with EMD from original signals and converted into fuzzy feature vectors, and then the mechanical fault can be detected. In the following stage, these extracted fuzzy feature vectors are input into the multi-classification SVM to identify the different abnormal cases. The proposed method is applied to the classification of a turbo-generator set under three different operating conditions.

7.3.5. Unsupervised/Semi-supervised Fuzzy Rule-Based FDD

A new fully unsupervised autonomous FRB approach to FDD has been recently proposed (Costa *et al.*, 2014a, 2014b). The algorithm is divided into two sequential stages, detection and identification. In the first stage, the system is able to detect through a recursive density estimation method, whether there is a fault or not. If a fault is detected, the second stage is activated and, after a spatial distribution analysis, the fault is classified by a fuzzy inference system in a fully unsupervised manner. That means that the operator does not need to know all types, or even the number of possible faults. The fuzzy rule basis is updated at each data sample read, and the number of rules can grow if a new class of fault is detected. The classification is performed by an autonomous label generator, which can be assisted by the operator.

The detection algorithm is based on the recursive density estimation (RDE) (Angelov, 2012), which allows building, accumulating, and self-learning a dynamically evolving information model of “normality”, based on the process data for particular specific plant, and considering the normal/accident-free cases only. Similarly, like other statistical methods [e.g., statistical process control (SPC)] (Cook *et al.*, 1997), RDE is an online statistical technique. However, it does not require that the process parameters follow Gaussian/normal distributions nor make other prior assumptions.

The fault identification algorithm is based on the self-learning and fully unsupervised evolving classifier algorithm called *AutoClass*, which is an AnYa-like FRB classifier and, unlike the traditional FRB systems (e.g., Mamdani, Takagi–Sugeno), AnYa does not require the definition of membership functions. The antecedent part of the inference rule uses the concepts of data clouds (Angelov and Yager, 2012) and relative data density,

representing exactly the real distribution of the data.

A data cloud is a collection of data samples in the n -dimensional space, similar to the well-known data clusters, however, it is different since a data cloud forming is non-parametric and it does not have a specific shape or boundary (Angelov, 2012) and, instead, it follows the exact real data distribution. A given data sample can belong to all the data clouds with a different degree $\gamma \in [0; 1]$, thus the fuzzy aspect of the model is preserved.

The consequent of the inference rule in *AutoClass* is a zero-order Takagi–Sugeno crisp function, i.e., a class label $L_i = [1, K]$. The inference rules follow the construct of an AnYa-like FRB system (Angelov and Yager, 2012):

$$\mathfrak{R}^i : \text{IF}(\vec{x} \sim \aleph^i) \text{THEN}(L^i),$$

where \mathfrak{R}^i is the i -th rule, $\vec{x} = [x_1, x_2, \dots, x_n]^T$ is the input data vector, $\aleph^i \in \mathfrak{R}^n$ is the i -th data cloud and \sim denotes the fuzzy membership expressed linguistically as “is associated with” or “is close to”. The inference in *AutoClass* is produced using the “winner-takes-it-all” rule.

The degree of membership of the input vector x_k to the data cloud \aleph^i , defined as the relative local density, can be recursively calculated as (Angelov, 2012)

$$\gamma_k^i = \frac{1}{1 + \|x_k - \mu_k\|^2 + X_k - \|\mu_k\|^2} \quad (8)$$

where μ_k and X_k are updated by

$$\mu_k = \frac{k-1}{k} \mu_{k-1} + \frac{1}{k} x_k, \quad \mu_1 = x_1, \quad (9)$$

$$X_k = \frac{k-1}{k} X_{k-1} + \frac{1}{k} \|x_k\|^2, \quad X_1 = \|x_1\|^2. \quad (10)$$

Both stages of the proposed FDD approach can start “from scratch”, from the very first data sample acquired, with no previous knowledge about the plant model or dynamics, training or complex user-defined parameters or thresholds. The generated fuzzy rules have no specific parameters or shapes for the membership functions and the approach is entirely data-driven.

The approach was implemented and tested in a real liquid level control application, using a Laboratory pilot plant for industrial process control, similar to the plant presented in [sub-section 7.3.1](#), however, in industrial proportions. The fault-free behavior of the system, after a change of reference, is shown in [Figure 7.23](#).

A larger set of faults was, then, generated, including actuator, leakage, stuck valves and disturbance-related faults. [Figure 7.24](#) shows the detection stage of a sequence of fault events. While monitoring the oscillation of the error and control signals, the algorithm is able to detect the beginning (black bars) and end (grey bars) of the faults F_2 , F_4 , F_1 and

F_9 . In this stage, the system is responsible only for distinguishing the normal operation from a fault and considers only the steady state regime (which in Figure 7.23, for example, starts around 75s).

After a fault is detected, the identification/classification stage, based on the fuzzy algorithm *AutoClass* is activated. The data is spatially distributed in an n -dimensional feature space. In this particular example, two features were used — here called Feature 1 and Feature 2, thus $x = \{Feature\ 1, Feature\ 2\}$ —, where the first one is the period and the second one is the amplitude of the control signal.

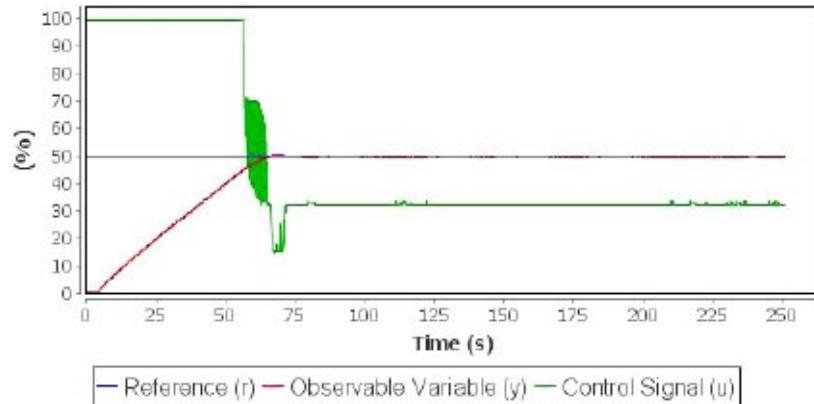


Figure 7.23: Fault-free state of operation of the plant.

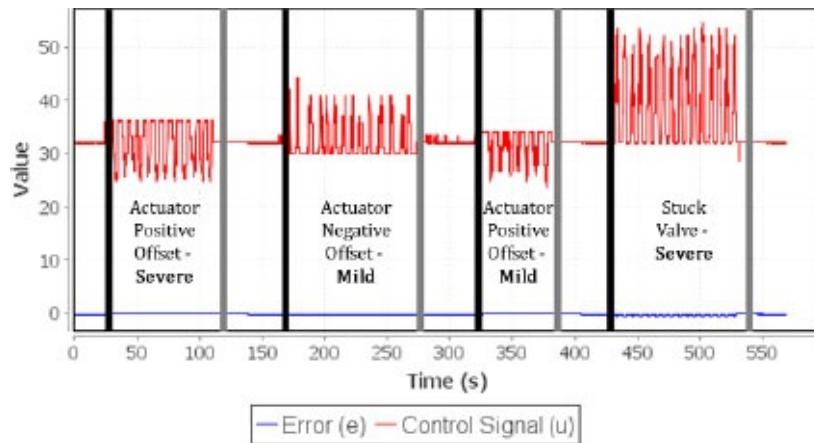


Figure 7.24: Detection of a sequence of faults using RDE.

Figure 7.25 shows the n -dimensional feature space after the reading of all data samples.

The fuzzy rule-based classifier, autonomously generated from the data stream, which consists of 5,600 data samples, is presented as follows:

$$\mathfrak{R}^1 : \text{IF } (\vec{x} \sim \mathbb{X}^1) \text{ THEN ("Class 1")},$$

$$\mathfrak{R}^2 : \text{IF } (\vec{x} \sim \mathbb{X}^2) \text{ THEN ("Class 2")},$$

$$\mathfrak{R}^3 : \text{IF } (\vec{x} \sim \mathbb{X}^3) \text{ THEN ("Class 3")},$$

with

$$\mathbb{X}^1 : c_1 = [0.416, 3.316] \text{ and } ZI_1 = [0.251, 0.756],$$

$$\mathbb{X}^2 : c_2 = [-0.513, 2.706] \text{ and } ZI_2 = [0.250, 0.601],$$

$$\mathcal{N}^3 : c_3 = [-0.416, 1.491] \text{ and } ZI_3 = [0.197, 0.451],$$

where c_i is the focal point (mean) and ZI_i is the zone of influence of the cloud.

The fault classification procedure, which is only executed after the detection of a fault by the RDE-based detection algorithm (that is why no rule for the fault-free case was created), is quite unique in the sense that it autonomously and in a completely unsupervised manner (automatic labels) identifies the types of faults. Traditional models, such as neural networks, start to drift and a re-calibration is needed. This unsupervised and self-learning method does not suffer from such disadvantage because it is adapting and evolving.

It should be noted that the class labels are generated automatically in a sequence (“Class 1”, “Class 2” and so on) as different faults are detected. Of course, these labels do not represent the actual type or location of the fault, but they are very useful to distinguish different faults. Since there is no training or pre-definition of faults or models, the correct labeling can be performed in a semi-supervised manner by the human operators without requiring prompt/synchronized actions from the user. Moreover, in a semi-supervised approach, the operator should be able to merge, split and rename the generated clouds/rules/classes of faults, enabling the classification of faults/operation states that cannot be represented by compact/convex data clouds.

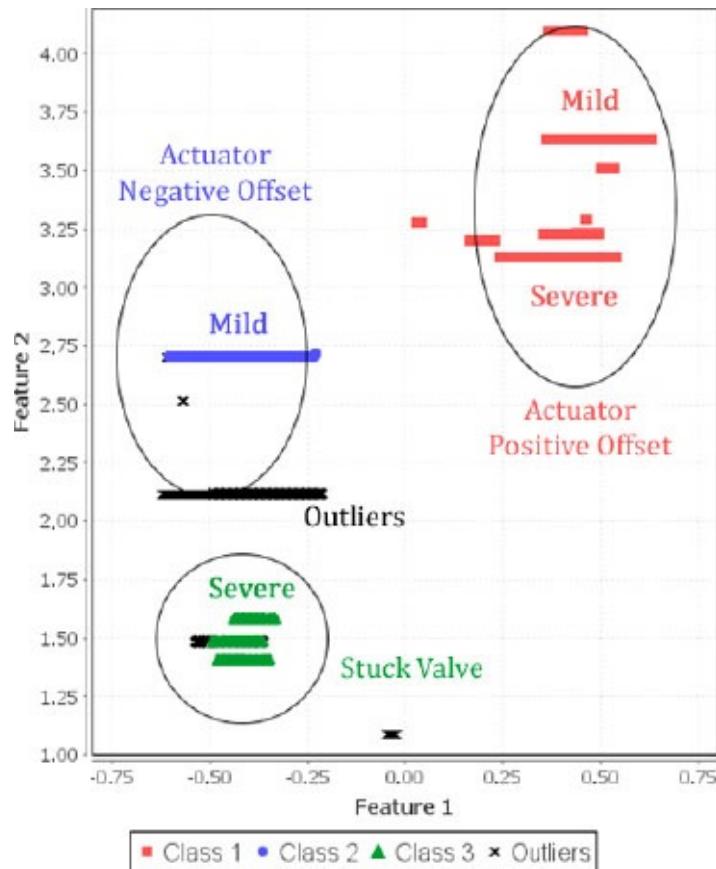


Figure 7.25: Classification and automatic labeling of the faults using *AutoClass*.

It is also important to highlight that even though the detection and classification process were presented separately, they are performed simultaneously, online, from the

acquired data sample.

7.3.6. Other Approaches to Fuzzy FDD

Most of the methods for FDD addressed in the literature are problem-driven and, although classifiable into one of the three main groups of methods (quantitative, qualitative or process history-based), they still can be very different in many aspects. That said, we can still recommend the following literature.

Observation-based fault detection in robots is discussed in Sneider and Frank (1996). The supervision method proposed makes use of non-measurable process information. This method is reviewed and applied to the fault detection problem in an industrial robot, using dynamic robot models, enhanced by the inclusion of nonlinear friction terms. A fuzzy logic residual evaluation approach of model-based fault detection techniques is investigated for processes with unstructured disturbances, arising from model simplification.

In Simani (2013), the author proposes an approach based on analytical redundancy, focused on fuzzy identification oriented to the design of a set of fuzzy estimators for fault detection and identification. Different aspects of the fault detection problem are treated in the paper, such as model structure, parameter identification and residual generation and fault diagnosis techniques. The proposed approach is applied to a real diesel engine.

A model-based approach to FDD using fuzzy matching is proposed in Dexter and Benouarets (1997). The scheme uses a set of fuzzy reference models, obtained offline from simulation, which describes normal and, as an extension of the example given in [Section 7.3.2](#), also describes faulty operation. A classifier based on fuzzy matching evaluates the degree of similarity every time the online fuzzy model is identified. The method also deals with any ambiguity, which may result from normal or faulty states of operation, or different types of faults, with similar symptoms at a given operating state.

A method for design of unknown fuzzy observers for Takagi–Sugeno (TS) models is addressed in Akhenak *et al.* (2009). The paper presents the development of a robust fuzzy observer in the presence of disturbances, which is used for detection and isolation of faults which can affect a TS model. The proposed methodology is applied to a simulated environment by estimating the yaw rate and the fault of an automatic steering vehicle.

In Gmytrasiewicz *et al.* (1990), Tanaka *et al.* (1983) and Peng *et al.* (2008), different approaches to fault diagnosis for systems based on fuzzy fault-tree analysis are discussed. These methods aim to diagnose component failures from the observation of fuzzy symptoms, using the information contained in a fault-tree. While in conventional fault-tree analysis, the failure probabilities of components of a system are treated as exact values in estimating the failure probability of the top event, a fuzzy fault-tree employs the possibility, instead of the probability, of failure, namely a fuzzy set defined in probability

space.

Fault diagnosis based on trend patterns shown in the sensor measurements is presented in Dash *et al.* (2003). The process of trend analysis involves graphic representation of signal trends as temporal patterns, extraction of the trends, and their comparison, through a fuzzy estimation of similarity, to infer the state of the process. The technique is illustrated with its application for the fault diagnosis of an exothermic reactor.

In Lughofe and Guardiola (2008), and Skrjanc (2009), the authors present different approaches to the usage of fuzzy confidence intervals for model outputs in order to normalize residuals with the model uncertainty. While in the first paper, the authors evaluate the results based on high-dimensional measurement data from engine test benchmarks, in the latter one, the proposed method is used for a nonlinear waste-water treatment plant modeling.

In Oblak *et al.* (2007), the authors introduce an application of the interval fuzzy model in fault detection for nonlinear systems with uncertain interval-type parameters. An application of the proposed approach in a fault-detection system for a two-tank hydraulic plant is presented to demonstrate the benefits of the proposed method.

A fuzzy-genetic algorithm for automatic FDD in HVAC systems is presented in Lo *et al.* (2007). The proposed FDD system monitors the HVAC system states continuously by a fuzzy system with optimal fuzzy inference rules generated by a genetic algorithm. Faults are represented in different levels and are classified in an online manner, concomitantly with the rule basis tuning.

Last, but not least, the reader is referred to Rahman *et al.* (2010), Calado and Sá da Costa (2006), which present the literature reviews of neuro-fuzzy applications in FDD systems. These surveys present many applications of fault detection, isolation and classification, using neuro-fuzzy techniques, either single or combined, highlighting the advantages and disadvantages of each presented approach.

7.4. Open Benchmarks

The study and validation of new and existing FDD techniques is, usually, related to the use of well-known and already validated benchmarks. They are advantageous in the sense that they are enabling one to perform the experiments using real industrial data and serving as a fair basis of comparison to other techniques.

Among the most used benchmarks for FDD, we surely need to mention Development and Applications of Methods for Actuator Diagnosis in Industrial Control Systems (DAMADICS), first introduced in Bartys *et al.* (2006). DAMADICS is an openly available benchmark system based on the industrial operation of the sugar factory Cukrownia Lublin SA, Poland. The benchmark considers many details of the physical and electro-mechanical properties of a real industrial actuator valve operating under challenging process conditions.

With DAMADICS, it is possible to simulate 19 abnormal events, along with the normal operation, from three actuators. A faulty state is composed by the type of the fault followed by the failure mode, which can be abrupt or incipient. DAMADICS was successfully used as testing platform in many applications, such as Puig *et al.* (2006), Almeida and Park (2008), and Frisk *et al.* (2003).

Another important benchmark, worth mentioning, is presented in Blanke *et al.* (1995). The benchmark is based on an electro-mechanical position servo, part of a shaft rotational speed governor for large diesel engines, located in a test facility at Aalborg University, Denmark. Potential faults include malfunction of velocity, position measurements or motor power drive and, can be fast or incipient, depending on the parameters defined by the operator. The benchmark also enables the study of robustness, sensor noise and unknown inputs.

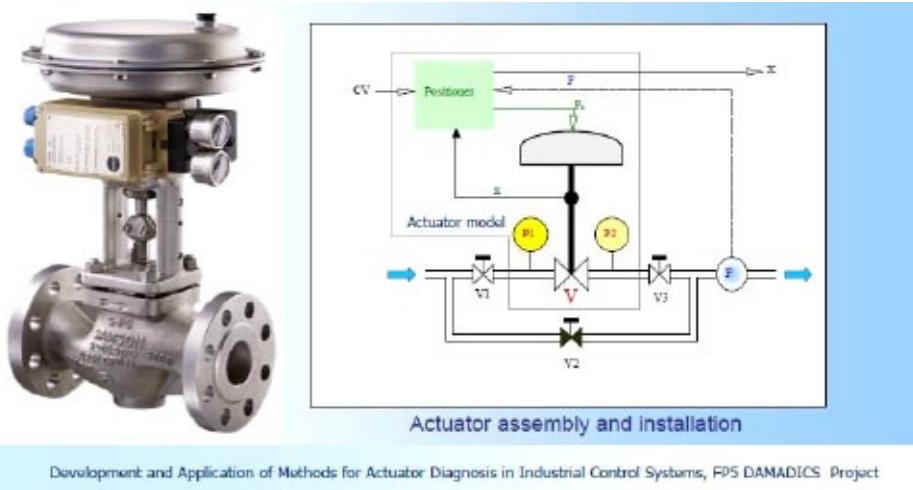


Figure 7.26: Actuator model of DAMADICS benchmark (Damadics, 1999).

In Odgaard *et al.* (2009), the authors introduce a benchmark model for fault tolerant control of wind turbines, which presents a diversity of faulty scenarios, including sensor/actuator, pitch system, drive train, generator and converter system malfunctions.

Last, but not least, Goupil and Puyou (2013) introduces a high-fidelity aircraft benchmark, based on a generic twin engine civil aircraft model developed by Airbus, including the nonlinear rigid-body aircraft model with a complete set of controls, actuator and sensor models, flight control laws, and pilot inputs.

7.5. Conclusions

A full overview of FDD methods was presented in this chapter with special attention to the fuzzy rule-based techniques. Basic and essential concepts of the field of study were presented, along with the review of numerous techniques introduced in the literature, from more traditional strategies, well-fitted in one of the three main categories of FDD techniques, to advanced state-of-the-art approaches, which combine elements and characteristics of multiple categories. A benchmark simulated study was introduced and used to illustrate and compare different types of methods, based either on quantitative/qualitative knowledge or on the data acquired from the process. A few other benchmark applications introduced in the literature were also referred, encouraging the reader to use them as tools for analysis/comparison. As previously suggested in other works, the use of hybrid FDD techniques, combining the best features of each group, is very often advised, since all groups of methods present their own advantages and disadvantages, and single methods frequently lack a number of desirable features necessary for an ideal FDD system.

References

- Abonyi, J. (2003). *Fuzzy Model Identification for Control*. Boston, USA: Birkhäuser.
- Akhenak, A., Chadli, M., Ragot, J. and Maquin, D. (2009). Design of observers for Takagi–Sugeno fuzzy models for fault detection and isolation. In *Proc. Seventh IFAC Symp. Fault Detect., Supervision Saf. Tech. Process., SAFEPROCESS,9*.
- Almeida, G. M. and Park, S. W. (2008). Fault detection and diagnosis in the DAMADICS benchmark actuator system — a hidden Markov model approach. In *Proc. 17th World Congr. Int. Fed. Autom. Control*, Seoul, Korea, July 6–11, pp. 12419–12424.
- Angelov, P. (2012). *Autonomous Learning Systems: From Data to Knowledge in Real Time*. Hoboken, New Jersey: John Wiley and Sons.
- Angelov, P., Giglio, V., Guardiola, C., Lughofer, E. and Lujan, J. M. (2006). An approach to model-based fault detection in industrial measurement systems with application to engine test benches. *Meas. Sci. Technol.*, 17(7), pp. 1809–1818.
- Angelov, P. and Yager, R. (2012). A new type of simplified fuzzy rule-based systems. *Int. J. Gen. Syst.*, 41, pp. 163–185.
- Angelov, P. and Zhou, X. (2008). Evolving fuzzy-rule-based classifiers from data streams. *IEEE Trans. Fuzzy Syst.*, 16, pp. 1462–1475.
- Bae, H., Kim S., Kim, J. M. and Kim, K. B. (2005). Development of flexible and adaptable fault detection and diagnosis algorithm for induction motors based on self-organization of feature extraction. In *Proc. 28th Ann. German Conf. AI, KI 2005*, Koblenz, Germany, September 11–14, pp. 134–147.
- Bartyś, M., Patton, R., Syfert, M., de lasHeras, S. and Quevedo, J. (2006). Introduction to the DAMADICS actuator FDI benchmark study. *Control Eng. Pract.*, 14(6), pp. 577–596.
- Blanke, M., Bøgh, S. A., Jørgensen, R. B. and Patton, R. J. (1995). Fault detection for a diesel engine actuator: a benchmark for FDI. *Control Eng. Pract.*, 3(12), pp. 1731–1740.
- Calado, J. and Sá da Costa, J. (2006). Fuzzy neural networks applied to fault diagnosis. In *Computational Intelligence in Fault Diagnosis*. London: Springer-Verlag, pp. 305–334.
- Carl, J. D., Tantawy, A., Biswas, G. and Koutsoukos, X. (2012). Detection and estimation of multiple fault profiles using generalized likelihood ratio tests: a case study. *Proc. 16th IFAC Symp. Syst. Identif.*, pp. 386–391.
- Casillas, J., Cordon, O., Herrera, F. and Magdalena, L. (2003). *Interpretability Issues in Fuzzy Modeling*. Berlin, Heidelberg: Springer-Verlag.
- Castro, J. L. and Delgado, M. (1996). Fuzzy systems with defuzzification are universal approximators. *IEEE Trans. Syst., Man Cybern., Part B: Cybern.*, 26(1), pp. 149–152.
- Cengel, Y. A., Turner, R. H. and Cimbala, J. M. (2012). *Fundamentals of Thermal-Fluid Sciences, Fourth Edition*. USA: McGraw Hill, pp. 471–503.
- Chen, H., Jiang, G. and Yoshihira, K. (2006). Fault detection in distributed systems by representative subspace mapping. *Pattern Recognit., ICPR 2006, 18th Int. Conf.*, 4, pp. 912–915.
- Chen, J. and Patton, R. J. (1999). *Robust Model-Based Fault Diagnosis for Dynamic Systems*. Boston, Massachusetts: Kluwer Academic Publishers.
- Chiang, L. H. and Russell, E. L. and Braatz, R. D. (2001). *Fault Detection and Diagnosis in Industrial Systems*. London: Springer.
- Chow, E. Y. and Willsky, A. S. (1984). Analytical redundancy and the design of robust failure detection systems. *IEEE Trans. Autom. Control*, 29(7), pp. 603–614.
- Cook, G., Maxwell, J., Barnett, R. and Strauss, A. (1997). Statistical process control application to weld process. *IEEE Trans. Ind. Appl.*, 33, pp. 454–463.
- Costa, B., Bezerra, C. G. and Guedes, L. A. (2010). Java fuzzy logic toolbox for industrial process control. In *Braz. Conf. Autom. (CBA)*, Bonito-MS, Brazil: Brazilian Society for Automatics (SBA).
- Costa, B., Skrjanc, I., Blazic, S. and Angelov, P. (2013). A practical implementation of self-evolving cloud-based control

- of a pilot plant. *IEEE Int. Conf., Cybern. (CYBCONF)*, June 13–15, pp. 7, 12.
- Costa, B. S. J., Angelov, P. P. and Guedes, L. A. (2014a). Fully unsupervised fault detection and identification based on recursive density estimation and self-evolving cloud-based classifier. *Neurocomputing* (Amsterdam), 1, p. 1.
- Costa, B. S. J., Angelov, P. P. and Guedes, L. A. (2014b). Real-time fault detection using recursive density estimation. *J. Control, Autom. Elec. Syst.*, 25, pp. 428–437.
- Costa, B. S. J., Bezerra, C. G. and de Oliveira, L. A. H. G. (2012). A multistage fuzzy controller: toolbox for industrial applications. *IEEE Int. Conf., Ind. Technol. (ICIT)*, March 19–21, pp. 1142, 1147.
- Damadics Project. (1999). In <http://sac.upc.edu/proyectos-de-investigacion/proyectosue/damadics>.
- Dash, S., Rengaswamy, R. and Venkatasubramanian, V. (2003). Fuzzy-logic based trend classification for fault diagnosis of chemical processes. *Comput. Chem. Eng.*, 27(3), pp. 347–362.
- Dexter, A. L. and Benouarets, M. (1997). Model-based fault diagnosis using fuzzy matching. *Trans. Syst., Man Cybern., Part A*, 27(5), pp. 673–682.
- Ding, S. X. (2008). *Model-Based Fault Diagnosis Techniques: Design Schemes, Algorithms, and Tools*. Berlin, Heidelberg: Springer.
- Donders, S. (2002). *Fault Detection and Identification for Wind Turbine Systems: A Closed-Loop Analysis*. Master's Thesis. University of Twente, The Netherlands: Faculty of Applied Physics, Systems and Control Engineering.
- Edwards, C., Lombaerts, T. and Smaili, H. (2010). Fault tolerant flight control: a benchmark challenge. *Lect. Notes Control Inf. Sci.*, 399, Springer.
- Efimov, D., Zolghadri, A. and Raïssi, T. (2011). Actuator fault detection and compensation under feedback control. *Autom.*, 47(8), pp. 1699–1705.
- Frank, P. M. and Köppen-Seliger, B. (1997). Fuzzy logic and neural network applications to fault diagnosis. *Int. J. Approx. Reason.*, 16(1), pp. 67–88.
- Frank, P. M. and Wünnenberg, J. (1989). Robust fault diagnosis using unknown input observer schemes. In Patton, R. J., Frank, P. M. and Clark, R. N. (eds.), *Fault Diagnosis in Dynamic Systems: Theory and Applications*. NY: Prentice Hall.
- Frisk, E. (2001). *Residual Generation for Fault Diagnosis*. PhD Thesis. Linköping University, Sweden: Department of Electrical Engineering.
- Frisk, E., Krys, M. and Cocquempot, V. (2003). Improving fault isolability properties by structural analysis of faulty behavior models: application to the DAMADICS benchmark problem. In *Proc. IFAC SAFEPROCESS'03*.
- Gacto, M. J., Alcalá, R. and Herrera, F. (2011). Interpretability of linguistic fuzzy rule-based systems: an overview of interpretability measures. *Inf. Sci.*, 20, pp. 4340–4360.
- Glass, A. S., Gruber, P., Roos, M. and Todtli, J. (1995). Qualitative model-based fault detection in air-handling units. *IEEE Control Syst.*, 15(4), pp. 11, 22.
- Gmytrasiewicz, P., Hassberger, J. A. and Lee, J. C. (1990). Fault tree-based diagnostics using fuzzy logic. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(11), pp. 1115–1119.
- Goupil, P. and Puyou, G. (2013). A high-fidelity airbus benchmark for system fault detection and isolation and flight control law clearance. *Prog. Flight Dynam., Guid., Navigat., Control, Fault Detect., Avionics*, 6, pp. 249–262.
- Hu, Q., He, Z. J., Zi Y., Zhang, Z. S. and Lei, Y. (2005). Intelligent fault diagnosis in power plant using empirical mode decomposition, fuzzy feature extraction and support vector machines. *Key Eng. Mater.*, 293–294, pp. 373–382.
- Insfran, A. H. F., da Silva, A. P. A. and Lambert Torres, G. (1999). Fault diagnosis using fuzzy sets. *Eng. Intell. Syst. Elec. Eng. Commun.*, 7(4), pp. 177–182.
- Isermann, R. (2009). *Fault Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance*. Berlin, Heidelberg: Springer.
- Isermann, R. (2005). Model-based fault-detection and diagnosis — status and applications. *Annu. Rev. Control*, 29(1), pp. 71–85.
- Katipamula, S., and Brambley, M. R. (2005). Methods for fault detection, diagnostics, and prognostics for building systems: a review, part I. *HVAC&R RESEARCH*, 11(1), pp. 3–25.

- Kerre, E. E. and Nachtegael, M. (2000). *Fuzzy Techniques in Image Processing*. Heidelberg, New York: Physica-Verlag.
- Korbicz, J., Koscielny, J. M., Kowalcuk, Z. and Cholewa, W. (2004). *Fault Diagnosis: Models, Artificial Intelligence and Applications*. Berlin, Heidelberg: Springer-Verlag.
- Kruse, R., Gebhardt, J. and Palm, R. (1994). *Fuzzy Systems in Computer Science*. Wiesbaden: Vieweg-Verlag.
- Kulkarni, M., Abou, S. C. and Stachowicz, M. (2009). Fault detection in hydraulic system using fuzzy logic. In *Proc. World Congr. Eng. Comput. Sci. 2, WCECS'2009*, San Francisco, USA, October 20–22.
- Laukonen, E. G., Passino, K. M., Krishnaswami, V., Lub, G.-C. and Rizzoni, G. (1995). Fault detection and isolation for an experimental internal combustion engine via fuzzy identification. *IEEE Trans. Control Syst. Technol.*, 3(9), pp. 347–355.
- Lemos, Caminhas, W and Gomide, F. (2013). Adaptive fault detection and diagnosis using an evolving fuzzy classifier. *Inf. Sci.*, 220, pp. 64–85.
- Liang, Y., Liaw, D. C. and Lee, T. C. (2000). Reliable control of nonlinear systems. *IEEE Trans. Autom. Control*, 45(4), 706–710.
- Lin, L. and Liu, C. T. (2007). Failure detection and adaptive compensation for fault tolerable flight control systems. *IEEE Trans. Ind. Inf.*, 3(4), pp. 322, 331.
- Lo, C. H., Chan, P. T., Wong, Y. K., Rad, A. B. and Cheung, K. L. (2007). Fuzzy-genetic algorithm for automatic fault detection in HVAC systems. *Appl. Soft Comput.*, 7(2), pp. 554–560.
- Lughofer, E. (2011). *Evolving Fuzzy Systems — Methodologies, Advanced Concepts and Applications*. Berlin, Heidelberg: Springer.
- Lughofer, E. (2013). On-line assurance of interpretability criteria in evolving fuzzy systems: achievements, new concepts and open issues. *Inf. Sci.*, 251, pp. 22–46.
- Lughofer, E. and Guardiola, C. (2008). Applying evolving fuzzy models with adaptive local error bars to on-line fault detection. *Proc. Genet. Evolving Fuzzy Syst.* Germany: Witten-Bommerholz, pp. 35–40.
- Mendonça, L. F., Sousa, J. M. C. and Sá da Costa, J. M. G. (2004a). Fault detection and isolation using optimized fuzzy models. In *Proc. 11th World Congr. IFSA'2005*. Beijing, China, pp. 1125–1131.
- Mendonça, L. F., Sousa, J. M. C. and Sá da Costa, J. M. G. (2004b). Fault detection and isolation of industrial processes using optimized fuzzy models. In Palade, V., Bocaniala, C. D. and Jain, L. C. (eds.), *Computational Intelligence in Fault Diagnosis*. Berlin Heidelberg: Springer, pp. 81–104.
- Mendonça, L. F., Sousa, J. M. C. and Sá da Costa, J. M. G. (2009). An architecture for fault detection and isolation based on fuzzy methods. *Expert Syst. Appl. Int. J.*, 36(2), pp. 1092–1104.
- Mendonça, L., Sousa, J. and da Costa, J. S. (2006). Fault detection and isolation of industrial processes using optimized fuzzy models. In Palade, V., Jain, L. and Bocaniala, C. D. (eds.), *Computational Intelligence in Fault Diagnosis, Advanced Information and Knowledge Processing*. London: Springer, pp. 81–104.
- Meneghetti, F. (2007). Mathematical modeling of dynamic systems. *Lab. Keynotes no. 2*. Natal, Brazil: Federal University of Rio Grande do Norte (UFRN).
- Murphrey, Y. L., Crossman, J. and Chen, Z. (2003). Developments in applied artificial intelligence. In *Proc. 16th Int. Conf. Ind. Eng. Appl. Artif. Intell. Expert Syst., IEA/AIE 2003*. Loughborough, UK, June 23–26, pp. 83–92.
- Nelles, O. (2001). *Nonlinear System Identification*. Berlin: Springer.
- Oblak, S., Škrjanc, I. and Blažič, S. (2007). Fault detection for nonlinear systems with uncertain parameters based on the interval fuzzy model. *Eng. Appl. Artif. Intell.*, 20(4), pp. 503–510.
- Odgaard, P. F., Stoustrup, J. and Kinnaert, M. (2009). Fault tolerant control of wind turbines — a benchmark model. In *Proc. Seventh IFAC Symp. Fault Detect., Supervision Saf. Tech. Process.* Barcelona, Spain, June 30–July 3, pp. 155–160.
- Patan, K. (2008). Artificial neural networks for the modelling and fault diagnosis of technical processes. *Lect. Notes Control Inf. Sci.*, 377, Springer.
- Patton, R. J., Frank, P. M. and Clark, R. N. (2000). *Issues of Fault Diagnosis for Dynamic Systems*. London: Springer-Verlag.

- Pedrycz, W. and Gomide, F. (2007). *Fuzzy Systems Engineering: Toward Human-Centric Computing*. Hoboken, New Jersey: John Wiley & Sons.
- Peng, Z., Xiaodong, M., Zongrun, Y. and Zhaoxiang, Y. (2008). An approach of fault diagnosis for system based on fuzzy fault tree. *Int. Conf. Multi Media Inf. Technol., MMIT'08*, December 30–31, pp. 697–700.
- Puig, V., Stancu, A., Escobet, T., Nejjari, F., Quevedo, Q. and Patton, R. J. (2006). Passive robust fault detection using interval observers: application to the DAMADICS benchmark problem. *Control Eng. Pract.*, 14(6), pp. 621–633.
- Quanser (2004). Coupled tanks user manual.
- Rahman, S. A. S. A., Yusof, F. A. M. and Bakar, M. Z. A. (2010). The method review of neuro-fuzzy applications in fault detection and diagnosis system. *Int. J. Eng. Technol.*, 10(3), pp. 50–52.
- Samantaray, A. K. and Bouamama, B. O. (2008). *Model-based Process Supervision: A Bond Graph Approach, First Edition*. New York: Springer.
- Serdio, F., Lugofer, E., Pichler, K., Buchegger, T. and Efendic, H. (2014). Residual-based fault detection using soft computing techniques for condition monitoring at rolling mills. *Inf. Sci.*, 259, pp. 304–320.
- Silva, D. R. C. (2008). *Sistema de Detecção e Isolamento de Falhas em Sistemas Dinâmicos Baseado em Identificação Paramétrica (Fault Detection and Isolation in Dynamic Systems Based on Parametric Identification)*. PhD Thesis. Federal University of Rio Grande do Norte (UFRN), Brazil: Department of Computer Engineering and Automation.
- Simani, S. (2013). Residual generator fuzzy identification for automotive diesel engine fault diagnosis. *Int. J. Appl. Math. Comput. Sci.*, 23(2), pp. 419–438.
- Simani, S., Fantuzzi, C. and Patton, R. J. (2002). *Model-based Fault Diagnosis in Dynamic Systems Using Identification Techniques*. Berlin, Heidelberg: Springer-Verlag.
- Skrjanc, I. (2009). Confidence interval of fuzzy models: an example using a waste-water treatment plant. *Chemometr. Intell. Lab. Syst.*, 96, pp. 182–187.
- Sneider, H. and Frank, P. M. (1996). Observer-based supervision and fault detection in robots using nonlinear and fuzzy logic residual evaluation. *IEEE Trans. Control Syst. Technol.*, 4(3), pp. 274, 282.
- Tanaka, H., Fan, L. T., Lai, F. S. and Toguchi, K. (1983). Fault-tree analysis by fuzzy probability. *IEEE Trans. Reliab.*, R-32(5), pp. 453, 457.
- Venkatasubramanian, V., Rengaswamy, K. and Kavuri, S. N. (2003a). A review of process fault detection and diagnosis-part II: Qualitative models and search strategies. *Comput. Chem. Eng.*, 27, pp. 293–311.
- Venkatasubramanian, V., Rengaswamy, K., Yin, K. and Kavuri, S. N. (2003b). A review of process fault detection and diagnosis-part III: Process history based methods. *Comput. Chem. Eng.*, 27, pp. 327–346.
- Venkatasubramanian, V., Rengaswamy, K., Yin, K. and Kavuri, S. N. (2003c). A review of process fault detection and diagnosis-part I: Quantitative model-based methods. *Comput. Chem. Eng.*, 27, pp. 313–326.
- Wang, P. and Guo, C. (2013). Based on the coal mine's essential safety management system of safety accident cause analysis. *Am. J. Environ. Energy Power Res.*, 1, pp. 62–68.
- Witczak, M. (2014). *Fault Diagnosis and Fault-Tolerant Control Strategies for Nonlinear Systems: Analytical and Soft Computing Approaches*. Berlin, Heidelberg: Springer-Verlag.
- Yang, H., Mathew, J. and Ma, L. (2003). Vibration feature extraction techniques for fault diagnosis of rotating machinery: a literature survey. In *Proc. Asia-Pac. Vib. Conf.*, Gold Coast, Australia, November 12–14.
- Zhao, Y., Lam, J. and Gao, H. (2009). Fault detection for fuzzy systems with intermittent measurements. *IEEE Trans. Fuzzy Syst.*, 17(2), pp. 398–410.
- Zhou, K. and Ren, Z. (2001). A new controller architecture for high performance, robust and fault-tolerant control. *IEEE Trans. Autom. Control*, 46(10), pp. 1613–1618.
- Zhou, S. M. and Gan, J. Q. (2008). Low-level interpretability and high-level interpretability: a unified view of data-driven interpretable fuzzy systems modelling. *Fuzzy Sets Syst.*, 159(23), pp. 3091–3131.

Part II

Artificial Neural Networks and Learning Systems

Chapter 8

The ANN and Learning Systems in Brains and Machines

Leonid Perlovsky

8.1. The Chapter Preface

This chapter overviews mathematical approaches to learning systems and recent progress toward mathematical modeling and understanding of the mind mechanisms, higher cognitive and emotional functions, including cognitive functions of the emotions of the beautiful and the music. It is clear today that any algorithmic idea can be realized in a neural-network like architecture. Therefore, from a mathematical point of view, there is no reason to differentiate between Artificial Neural Networks (ANNs) and other algorithms. For this reason, words “algorithms,” “neural networks,” “machine learning,” “artificial intelligence,” “computational intelligence” are used in this chapter interchangeably. It is more important to acknowledge that the brain-mind is still a much more powerful learning device than popular ANNs and algorithms and try to understand why this is so. Correspondingly, this chapter searches for fundamental principles, which set the brain-mind apart from popular algorithms, for mathematical models of these principles, and for algorithms built on these models.

What does the mind do differently from ANN and algorithms? We are still far away from mathematical models deriving high cognitive abilities from properties of neurons, therefore mathematical modeling of the “mind” often is more appropriate than modeling the details of the brain, when the goal is to achieve “high” human-like cognitive abilities. The reader would derive a more informed opinion by reading this entire book. This chapter devotes attention to fundamental cognitive principles of the brain-mind, and their mathematical modeling in parallel with solving engineering problems. I also discuss experimental evidence derived from psychological, cognitive, and brain imaging experiments about mechanisms of the brain-mind to the extent it helps the aim: identifying fundamental principles of the mind.

The search for the fundamental principles of the brain-mind and their mathematical models begins with identifying mathematical difficulties behind hundreds of ANNs and algorithms. After we understand the fundamental reasons for the decades of failure of artificial intelligence, machine learning, ANN, and other approaches to modeling the human mind, and to developing mathematical techniques with the human mind power, then we turn to discussing the fundamental principles of the brain-mind (Perlovsky, 2010e).

The next section analyzes and identifies several mathematical difficulties common to wide groups of algorithms. Then we identify a single fundamental mathematical reason for computers failing behind the brain-mind. This reason is reliance of computational intelligence on classical logic.

This is an ambitious statement and the chapter analyzes mathematical as well as cognitive reasons for logic being the culprit of decades of mathematical and cognitive failures. Fundamental mathematical as well as psychological reasons are identified for “how and why” logic that used to be considered a cornerstone of science, turned out to be

inadequate, when attempting to understand the mind.

Then, we formulate a mathematical approach that has overcome limitations of logic. It resulted in hundreds of times improvement in solving many classical engineering problems, and it solved problems that remained unsolvable for decades. It also explained what used to be psychological mysteries for long time. In several cognitive and brain-imaging experiments, it has been demonstrated to be an adequate mathematical model for brain-mind processes. Amazingly, this mathematical technique is simpler to code and use than many popular algorithms.

8.2. A Short Summary of Learning Systems and Difficulties They Face

Mathematical ideas invented in the 1950s and 1960s for learning are still used today in many algorithms, therefore let me briefly overview these ideas and identify sources of the mathematical difficulties (Perlovsky, 2001, 2002a).

Computational approaches to solving complex engineering problems by modeling the mind began almost as soon as computers appeared. These approaches in the 1950s followed the known brain neural structure. In 1949, Donald Hebb published what became known as the Hebb-rule: neuronal synaptic connections grow in strength, when they are used in the process of learning. Mathematicians and engineers involved in developing learning algorithms and devices in the early 1950s were sure soon computers would surpass by far the human minds in their abilities. Everybody knows today that Frank Rosenblatt developed a first ANN capable of learning, called Perceptron. Perceptron, however, could only learn to solve fairly simple problems. In 1969, Marvin Minsky and Seymour Papert mathematically proved limits to Perceptron learning.

Statistical pattern recognition algorithms were developed in parallel (Duda *et al.*, 2000). They characterized patterns by features. D features formed a D -dimensional classification space; features from learning samples formed distributions in this space, statistical methods were used to characterize the distributions and derive a classifier. One approach to a classifier design defined a plane, or a more complex surface in a classification space, which separated classes. Another approach became known as a nearest neighbor or a kernel method. In this case, neighborhoods in a classification space near known examples from each class are assigned to the class. The neighborhoods are usually defined using kernel functions (often bell-shaped curves, Gaussians). Use of Gaussian Mixtures to define neighborhoods is a powerful method; first attempts toward deriving such algorithms have been complex, and efficient convergence has been a problem (Titterington *et al.*, 1985). Eventually good algorithms have been derived (Perlovsky and McManus, 1991). Today Gaussian Mixtures are widely used (Mengersen *et al.*, 2011). Yet, these methods turned out to be limited by the dimensionality of classification space.

The problem with dimensionality was discovered by Richard Bellman (1962), who called it “the curse of dimensionality.” The number of training samples has to grow exponentially (or combinatorially) with the number of dimensions. The reason is in the geometry of high-dimensional spaces: there are “no neighborhoods”, most of the volume is concentrated on the periphery (Perlovsky, 2001). Whereas kernel functions are defined so that the probability of belonging to a class rapidly falls with the distance from a given example, in high-dimensional spaces volume growth may outweigh the kernel function fall; if kernels fall exponentially (like Gaussian), the entire “neighborhood” resides on a thin shell where the kernel fall is matched by the volume rise. Simple problems have been solved efficiently, but learning more complex problems seems impossible.

Marvin Minsky (1965) and many colleagues suggested that learning was too complex and premature. Artificial intelligence should use knowledge stored in computers. Systems storing knowledge in a form of “if, ..., then, ...,” rules are called expert systems and are still being used. But when learning is attempted, rules often depend on other rules and grow into combinatorially large trees of rules.

A general approach attempting to combine existing knowledge and learning was model-based learning popular in the 1970s and 1980s. This approach used parametric models to account for existing knowledge, while learning was accomplished by selecting the appropriate values for the model parameters. This approach is simple when all the data comes from only one mode, for example, estimation of a Gaussian distribution from data, or estimation of a regression equation. When multiple processes have to be learned, algorithms have to split data among models and then estimate model parameters. I briefly describe one algorithm, multiple hypotheses testing (MHT), which is still used today (Singer *et al.*, 1974). To fit model parameters to the data, MHT uses multiple applications of a two-step process. First, an association step assigns data to models. Second, an estimation step estimates parameters of each model. Then a goodness of fit is computed (such as likelihood). This procedure is repeated for all assignments of data to models, and at the end model parameters corresponding to the best goodness of fit are selected. The number of associations is combinatorially large, therefore MHT encounters combinatorial complexity and could only be used for very simple problems.

In the 1970s, the idea of self-learning neural system became popular again. Since the 1960s, Stephen Grossberg continued research into the mechanisms of the brain-mind. He led a systematic exploitation of perceptual illusions for deciphering neural-mathematical mechanisms of perception—similar to I. Kant using typical errors in judgment for deciphering *a priori* mechanisms of the mind. But Grossberg’s ideas seemed too complex for a popular following. Adaptive Resonance Theory (ART) became popular later (Carpenter and Grossberg, 1987); it incorporated ideas of interaction between bottom-up (BU), and top-down (TD), signals considered later.

Popular attention was attracted by the idea of Backpropagation, which overcame earlier difficulties of the Perceptron. It was first invented by Arthur Bryson and Yu-Chi Ho in 1969, but was ignored. It was reinvented by Paul Werbos in 1974, and later in 1986 by David Rumelhart, Geoffrey Hinton, and Ronald Williams. The Backpropagation algorithm is capable of learning connection weights in multilayer feedforward neural networks. Whereas an original single layer Perceptron could only learn a hyperplane in a classification space, two layer networks could learn multiple hyperplanes and therefore define multiple regions, three layer networks could learn classes defined by multiple regions.

Multilayer networks with many weights faced the problem of overfitting. Such networks can learn (fit) classes of any geometrical shape and achieve a good performance

on training data. However, when using test data, which were not part of the training procedure, the performance could significantly drop. This is a general problem of learning algorithms with many free parameters learned from training data. A general approach to this problem is to train and test a neural network or a classifier on a large number of training and test data. As long as both training and testing performance continue improving with increasing number of free parameters, this indicates valid learning; but when increasing number of parameters results in poorer performance, this is a definite indication of overfitting. A valid training-testing procedure could be exceedingly expensive in research effort and computer time.

A step toward addressing the overfitting problem in an elegant and mathematically motivated way has been undertaken in the Statistical Learning Theory (SLT) (Vapnik, 1999). SLT seems one of the very few theoretical breakthroughs in learning theory. SLT promised to find a valid performance without overfitting in a classification space of any dimension from training data alone. The main idea of SLT is to find a few most important training data points (support vectors) needed to define a valid classifier in a classification sub-space of a small dimension. Support Vector Machines (SVMs) became very popular, likely due to a combination of elegant theory, relatively simple algorithms, and good performance.

However, SVM did not realize the theoretical promise of a valid optimal classifier in a space of any dimension. A complete theoretical argument why this promise has not been realized is beyond the scope of this chapter. A simplified summary is that for complex problems a fundamental parameter of the theory, the Vapnik–Chervonenkis dimension, turns out to be near its critical value. I would add that SLT does not rely on any cognitive intuition about brain-mind mechanisms. It does not seem that the SLT principles are used by the brain-mind. It could have been expected that if SLT would be indeed capable of a general optimal solution of any problem using a simple algorithm, its principles would have been discovered by biological algorithms during billions of years of evolution.

The problem of overfitting due to a large number of free parameters can be approached by adding a penalty function to the objective function to be minimized (or maximized) in the learning process (Setiono, 1997; Nocedal and Wright, 2006). A simple and efficient method is to add a weighted sum of squares of free parameters to a log likelihood or alternatively to the sum of squares of errors; this method is called Ridge regression. Practically, Ridge regression often achieves performance similar to SVM.

Recently, progress for a certain class of problems has been achieved using gradient boosting methods (Friedman *et al.*, 2000). The idea of this approach is to use an ensemble of weak classifiers, such as trees or stumps (short trees) and combine them until performance continues improving. These classifiers are weak in that their geometry is very simple. A large number of trees or stumps can achieve good performance. Why a large number of classifiers with many parameters do not necessarily over fit the data? It could

be understood from SLT; one SLT conclusion is that overfitting occurs not just due to a large number of free parameters, but due to an overly flexible classifier parameterization, when a classifier can fit every little “wiggle” in the training data. It follows that a large number of weak classifiers can potentially achieve good performance. A cognitively motivated variation of this idea is Deep Learning, which uses a standard back-propagation algorithm with standard, feed-forward multilayer neural networks with many layers (here is the idea of “deep”). Variations of this idea under the names of gradient boosting, ensembles of trees, and deep learning algorithms are useful, when a very large amount of labeled training data is available (millions of training samples), while no good theoretical knowledge exists about how to model the data. This kind of problem might be encountered in data mining, speech, or handwritten character recognition (Hinton *et al.*, 2012; Meier and Schmidhuber, 2012).

8.3. Computational Complexity and Gödel

Many researchers have attempted to find a general learning algorithm to a wide area of problems. These attempts continue from the 1950s until today. Many smart people spent decades perfecting a particular algorithm for a specific problem, and when they achieve success they are often convinced that they found a general approach. The desire to believe in existence of a general learning algorithm is supported by the fact that the human mind indeed can solve a lot of problems. Therefore, cognitively motivated algorithms such as Deep Learning can seem convincing to many people. If developers of the algorithm succeed in convincing many followers, their approach may flourish for five or even 10 years, until gradually researchers discover that the promise of finding a general learning algorithm has not been fulfilled (Perlovsky, 1998).

Other researchers have been inspired by the fact that the mind is much more powerful than machine learning algorithms, and they have studied mechanisms of the mind. Several principles of mind operations have been discovered, nevertheless mathematical modeling of the mind faced same problems as artificial intelligence and machine learning: mathematical models of the mind have not achieved cognitive power comparable to mind. Apparently, mind learning mechanisms are different from existing mathematical and engineering ideas in some fundamental way.

It turned out that indeed there is a fundamental mathematical principle explaining in unified way previous failures of attempts to develop a general learning algorithm and model learning mechanisms of the mind. This fundamental principle has been laying bare and well known to virtually everybody in full view of the entire mathematical, scientific, and engineering community. Therefore, in addition to explaining this fundamental mathematical reason I will also have to explain why it has not been noticed long ago. It turned out that this explanation reveals a fundamental psychological reason preventing many great mathematicians, engineers, and cognitive scientists from noticing “the obvious” (Perlovsky, 2013c).

The relationships between logic, cognition, and language have been a source of longstanding controversy. The widely accepted story is that Aristotle founded logic as a fundamental mind mechanism, and only during the recent decades science overcame this influence. I would like to emphasize the opposite side of this story. Aristotle thought that logic and language are closely related. He emphasized that logical statements should not be formulated too strictly and language inherently contains the necessary degree of precision. According to Aristotle, logic serves to communicate already made decisions (Perlovsky, 2007c). The mechanism of the mind relating language, cognition, and the world Aristotle described as forms. Today we call similar mechanisms mental representations, or concepts, or simulators in the mind (Perlovsky 2007b; Barsalou, 1999). Aristotelian forms are similar to Plato’s ideas with a marked distinction, forms are dynamic: their initial states, before learning, are different from their final states of

concepts (Aristotle, 1995). Aristotle emphasized that initial states of forms, forms-as-potentialities, are not logical (i.e., vague), but their final states, forms-as-actualities, attained in the result of learning, are logical. This fundamental idea was lost during millennia of philosophical arguments. It is interesting to add Aristotelian idea of vague forms-potentialities has been resurrected in fuzzy logic by Zadeh (1965); and dynamic logic described here is an extension of fuzzy logic to a process “from vague to crisp” (Perlovsky, 2006a, 2006b, 2013d). As discussed below, the Aristotelian process of dynamic forms can be described mathematically by dynamic logic; it corresponds to processes of perception and cognition, and it might be the fundamental principle used by the brain-mind, missed by ANNs and algorithms (Perlovsky, 2012c).

Classical logic has been the foundation of science since its very beginning. All mathematical algorithms, including learning algorithms and ANNs use logic at some step, e.g., fuzzy logic uses logic when deciding on the degree of fuzziness; all learning algorithms and ANNs use logic during learning: training samples are presented as logical statements. Near the end of the 19th century, logicians founded formal mathematical logic, the formalization of classical logic. Contrary to Aristotelian warnings they strived to eliminate the uncertainty of language from mathematics. Hilbert (1928) developed an approach named formalism, which rejected intuition as a matter of scientific investigation and was aimed at formally defining scientific objects in terms of axioms or rules. In 1900, he formulated famous Entscheidungsproblem: to define a set of logical rules sufficient to prove all past and future mathematical theorems. This was a part of “Hilbert’s program”, which entailed formalization of the entire human thinking and language.

Formal logic ignored the dynamic nature of Aristotelian forms and rejected the uncertainty of language. Hilbert was sure that his logical theory described mechanisms of the mind. “The fundamental idea of my proof theory is none other than to describe the activity of our understanding, to make a protocol of the rules according to which our thinking actually proceeds” Hilbert (1928). However, Hilbert’s vision of formalism explaining mysteries of the human mind came to an end in the 1930s, when Gödel (2001) proved internal inconsistency or incompleteness of formal logic. This development called Gödel theory is considered among most fundamental mathematical results of the previous century. Logic, that was believed to be a sure way to derive truths, a foundation of science, turned out to be basically flawed. This is a reason why theories of cognition and language based on formal logic are inherently flawed.

How exactly does Gödel’s incompleteness of logic affect every day logical arguments, cognitive science, and mathematical algorithms?

Gödel, as most mathematical logicians, considered infinite systems, in which every entity is defined with absolute accuracy, every number is specified with infinite precision. Therefore, usual everyday conversations are not affected directly by Gödel’s theory. However, when scientists, psychologists, cognitive scientists attempt to understand the

mind, perfectly logical arguments can lead to incorrect conclusions. Consider first mathematical algorithms. When Gödel's argument is applied to a finite system, such as computer or brain, the result is not fundamental incompleteness, but computational complexity (Perlovsky, 2013c). An algorithm that upon logical analysis seems quite capable of learning how to solve a certain class of problems in reality has to perform "too many" computations. How many is too many? Most learning algorithms have to consider combinations of some basic elements. The number of combinations grows very fast. Combinations of 2 or 3 are "few". But consider 100, not too big a number; however the number of combinations of 100 elements is 100^{100} , this exceeds all interactions of all elementary particles in the Universe in its entire lifetime, any algorithm facing this many computations is incomputable.

It turns out that algorithmic difficulties considered previously are all related to this problem. For example, a classification algorithm needs to consider combinations of objects and classes. Neural network and fuzzy logic have been specifically developed for overcoming this problem related to logic, but as mentioned they still use logic at some step. For example, training is an essential step in every learning system, training includes logical statements, e.g., "this is a chair". The combinatorial complexity follows as inadvertently as incompleteness in any logical system.

Combinatorial complexity is inadvertent and practically as "bad" as Gödel's incompleteness.

8.4. Mechanisms of the Mind. What the Mind does Differently?: Dynamic Logic

Although logic “does not work”, the mind works and recognizes objects around us. This section considers fundamental mechanisms of the mind, and mathematics necessary to model them adequately. Gradually, it will become clear why Gödel’s theory and the fundamental flaw of logic, while been known to all scientists since the 1930s, have been ignored when thinking about the mind and designing theories of artificial intelligence (Perlovsky, 2010a, 2013c).

Among fundamental mechanisms of the mind are mental representations. To simplify, we can think about them as mental imagery, or memories; these are mechanisms of concepts, which are fundamental for understanding the world: objects, scenes, events, as well as abstract ideas. Concepts model events in the world, for this reason they are also called mental models. We understand the world by matching concept-models to events in the world. In a “simple” case of visual perception of objects, concept-models of objects in memory are matched to images of objects on the retina.

Much older mechanisms are instincts (for historical reasons psychologists prefer to use the word “drives”). According to Grossberg–Levine theory of instincts and emotions (1987), instincts work like internal bodily sensors, e.g., our bodies have sensors measuring sugar level in blood. If it is below a certain level, we feel hunger. Emotions of hunger are transferred by neuron connections from instinctual areas in the brain to decision-making areas, and the mind devotes more attention to finding food.

This instinctual-emotional theory has been extended to learning. To find food, to survive we need to understand objects around us. We need to match concept-models of food to surrounding objects. This ability for understanding surroundings is so important for survival, that we have an inborn ability, an instinct that drives the mind to match concept-models to surrounding objects. This instinct is called the knowledge instinct (Perlovsky and McManus, 1991; Perlovsky, 2001, 2006a, 2007d). The neural areas of the brain participating in the knowledge instinct are discussed in (Levine and Perlovsky, 2008, 2010; Perlovsky and Levine, 2012). A mathematical model of the knowledge instinct (to simplify) is a similarity measure between a concept and the corresponding event. Satisfaction of any instinct is felt emotionally. There are specific emotions related to the knowledge instinct, these are aesthetic emotions (Perlovsky, 2001, 2014; Perlovsky *et al.*, 2011). Relations of aesthetic emotions to knowledge have been discovered by Kant (1790). Today it is known that these emotions are present in every act of perception and cognition. An experimental demonstration of existence of these emotions has been reported in (Perlovsky *et al.*, 2010). Their relations to emotions of the beautiful are discussed later.

Mental representations are organized in an approximate hierarchy from perceptual

elements to objects, to scenes, to more and more abstract concepts (Grossberg, 1988). Cognition and learning of representations involves interactions between a higher and lower level of representations (more than two layers may interact). This interaction involves bottom-up signals, BU (from lower to higher levels) and top-down, TD (from higher to lower levels). In a simplified view of object perception, an object image is projected from eye retina to the visual cortex (BU), in parallel, representations of expected objects are projected from memory to the visual cortex (TD). In an interaction between BU and TD projections, they are matched. When a match occurs, the object is perceived.

We discussed in previous sections that artificial intelligence algorithms and neural networks for decades have not been able to model this mechanism, logic used in algorithms caused the problem of computational complexity. For example, ART neural network (Carpenter and Grossberg, 1987) matched BU and TD signals using a mathematical procedure of nearest neighbor. This procedure relies on logic at some algorithmic step (e.g., selecting neighbors) and faces combinatorial complexity. The MHT is another approach used for matching the BU and TD signals, as we discussed it faces the combinatorial complexity due to a logical step of assignment of data to models.

To solve the problem, a mathematical technique of dynamic logic (DL), has been created to avoid logic and follow the Aristotelian process of forms from potentialities to actualities. Instead of stationary statements of classical logic, DL is a process-logic, a process “from vague to crisp”. This process starts with vague states, “potentialities”; the initial representations in DL are vague. DL thus predicts that mental representations and their initial projections to the visual cortex are vague. In interaction with crisp BU projections from retina, TD projections become crisp and match the BU projections, creating “actualities”. How does this process avoid logic and overcome combinatorial complexity?

Compare DL to MHT, which uses logic for data-model assignment. Due to the vagueness of DL initial representations, all data are associated with all models-representations. Thus, a logical assignment step is avoided. DL does not need to consider combinatorially large number of assignments, and combinatorial complexity is avoided. What remains is to develop a mathematical procedure that gradually improves models and concurrently makes associations less vague. Before formulating this mathematical procedure in the next section, let us discuss experimental evidence that confirms the fundamental DL prediction: representations are vague.

Everyone can conduct a simple 1/2 minute experiment to glimpse into neural mechanisms of representations and BU-TD signal interactions. Look at an object in front of your eyes. Then close your eyes and imagine this object. The imagined object is not as clear and crisp as the same object with opened eyes. It is known that visual imaginations are produced by TD signals, projecting representations to the visual cortex. Vagueness of the imagined object testifies to the vagueness of its representation. Thus, the fundamental

DL prediction is experimentally confirmed: representations are vague.

When you open your eyes, the object perception becomes crisp in all its details. This seems to occur momentarily, but this is an illusion of consciousness. Actually the process “from vague to crisp” takes quite long by neuronal measures, 0.6s, hundreds to thousands of neuronal interactions. But our consciousness works in such a way that we are sure, there is no “vague to crisp” process. We are not conscious of this process, and usually we are not conscious about vague initial states either.

This prediction of DL has been confirmed in brain imaging experiments (Bar *et al.*, 2006). These authors demonstrated that indeed initial representations are vague and usually unconscious. Also, the vague to crisp process is not accessible to consciousness. Indeed Aristotle’s formulation of cognition as a process from vague potentialities to logical actualities was ahead of his time.

8.5. Mathematical Formulation of DL

DL maximizes a similarity L between the BU data $X(n)$, $n = 1, \dots, N$, and TD representations-models $M(m)$, $m = 1, \dots, M$,

$$L = \prod_{n \in N} \sum_{m \in M} r(m) l(X(n)|M(m)). \quad (1)$$

Here $l(X(n)|M(m))$ are conditional similarities, later I denote them $l(n|m)$ for shortness; they can be defined so that under certain conditions they become the conditional likelihoods of data given the models, L becomes the total likelihood, and DL performs the maximum likelihood estimation. From the point of view of modeling the mind processes, DL matches BU and TD signals and implements the knowledge instinct. Coefficients $r(m)$, model rates, define a relative proportion of data described by model m ; for $l(n|m)$ to be interpretable as conditional likelihoods $r(m)$ must satisfy a condition,

$$\sum_{m \in M} r(m) = 1. \quad (2)$$

A product over data index n does not assume that data are probabilistically independent (as some simplified approaches do, to overcome mathematical difficulties), relationships among data are introduced through models. Models $M(m)$ describe parallel or alternative states of the system (the mind has many representations in its memories). Note, Equation (1) accounts for all possible alternatives of the data associations through all possible combinations of data and models-representations. Product over data index n of the sums of M models results in M^N items, this huge number is the mathematical reason for CC.

Learning consists in estimating model parameters, which values are unknown and should be estimated along with $r(m)$ in the process of learning. Among standard estimation approaches that we discussed is MHT (Singer *et al.*, 1974), which considers every item among M^N . Logically, this corresponds to considering separately every alternative association between data and models and choosing the best possible association (maximizing the likelihood). It is known to encounter CC.

DL avoids this logical procedure and overcomes CC as follows. Instead of considering logical associations between data and models, DL introduces continuous associations,

$$f(m|n) = r(m) l(n|m) \sqrt{\sum_{m' \in M} r(m') l(n|m')}. \quad (3)$$

For decades, associations between models and data have been considered an essentially discrete procedure. Representing discrete associations as continuous variables, Equation (3) is the conceptual breakthrough in DL. The DL process for the estimation of model parameters S_m begins with arbitrary values of these unknown parameters with one

restriction; parameter values should be defined so that partial similarities have large variances. These high-variance uncertain states of models, in which models correspond to any pattern in the data, correspond to the Aristotelian potentialities. In the process of estimation, variances are reduced so that models correspond to actual patterns in data, Aristotelian actualities. This DL-Aristotelian process “from vague to crisp” is defined mathematically as follows (Perlovsky, 2001, 2006a, 2006b):

$$\begin{aligned} \frac{Df(m|n)}{dt} &= f(m|n) \sum_{m' \in M} [f(m|m') - f(m'|n)] \left[\frac{\partial \ln l(n|m')}{\partial M_{m'}} \right] \left(\frac{\partial M_{m'}}{\partial S_{m'}} \right) \frac{dS_{m'}}{dt}, \\ \frac{dS_m}{dt} &= \sum_{n \in N} f(m|n) \left[\frac{\partial \ln l(n|m)}{\partial M_m} \right] \frac{\partial M_m}{\partial S_{m,m'm}} = 1 \text{ if } m = m', 0 \text{ otherwise.} \end{aligned} \quad (4)$$

Parameter t here is an internal time of the DL process; in digital computer implementation it is proportional to an iteration number.

A question might come up why DL, an essentially continuous process, seemingly very different from logic is called logic. This topic is discussed from a mathematical viewpoint in (Vityaev *et al.*, 2011, 2013; Kovalerchuk *et al.*, 2012). Here I would add that DL explains how logic emerges in the mind from neural operations: vague and illogical DL states evolve in the DL process to logical (or nearly logical) states. Classical logic is (approximately) an end-state of the DL processes.

Relations between DL, logic, and computation are worth an additional discussion (Perlovsky, 2013c). Dynamic logic is computable. Operations used by computers implementing dynamic logic algorithms are logical. But these logical operations are at a different level than human thinking. Compare the text of this chapter as stored in your computer and the related computer operations (assuming you have the book in e-version) to the human understanding of this chapter. The computer’s operations are logical, but on a different level from your “logical” understanding of this text. A computer does not understand the meaning of this chapter the way a human reader does. The reader’s logical understanding is on top of 99% of the brain’s operations that are not “logical” at this level. Our logical understanding is an end state of many illogical and unconscious dynamic logic processes.

8.6. A Recognition-Perception DL Model

Here, I illustrate the DL processes “from vague to crisp” for object perception. These processes model interactions among TD and BU signals. In these interactions, vague top-level representations are matched to crisp bottom-level representations. As mentioned, Aristotle discussed perception as a process from forms-as-potentialities to forms-as-actualities 2400 years ago. Amazingly, Aristotle was closer to the truth than many ANNs and computational intelligence algorithms used today.

From an engineering point of view, this example solves a problem of finding objects in strong clutter (unrelated objects of no interest, noise). This is a classical engineering problem. For cases of clutter signals stronger than object signals this problem has not been solved for decades.

For this illustration, I use a simple example, still unsolvable by other methods. In this example, DL finds patterns-objects in noise-clutter. Finding patterns below noise can be an exceedingly complex problem. I briefly repeat here why for decades existing algorithms could not solve this type of problems. If an exact pattern shape is not known and depends on unknown parameters, these parameters should be found by fitting the pattern model to the data. However, when the locations and orientations of patterns are not known, it is not clear which subset of the data points should be selected for fitting. A standard approach for solving this kind of problem, which has already been mentioned, is multiple hypotheses testing, MHT (Singer *et al.*, 1974); this algorithm searches through all logical combinations of subsets and models and encounters combinatorial complexity.

In the current example, we are looking for ‘smile’ and ‘frown’ patterns in noise shown in [Figure 8.1a](#) without noise, and in [Figure 8.1b](#) with noise, as actually measured (object signals are about 2–3 times below noise and cannot be seen). This example models the visual perception “from vague to crisp”. Even so, it is usually assumed that human visual system works better than any computer algorithm, this is not the case here. The DL algorithm used here models human visual perception, but human perception has been optimized by evolution for different types of patterns; the algorithm here is not as versatile as human visual system, it has been optimized for few types of patterns encountered here.

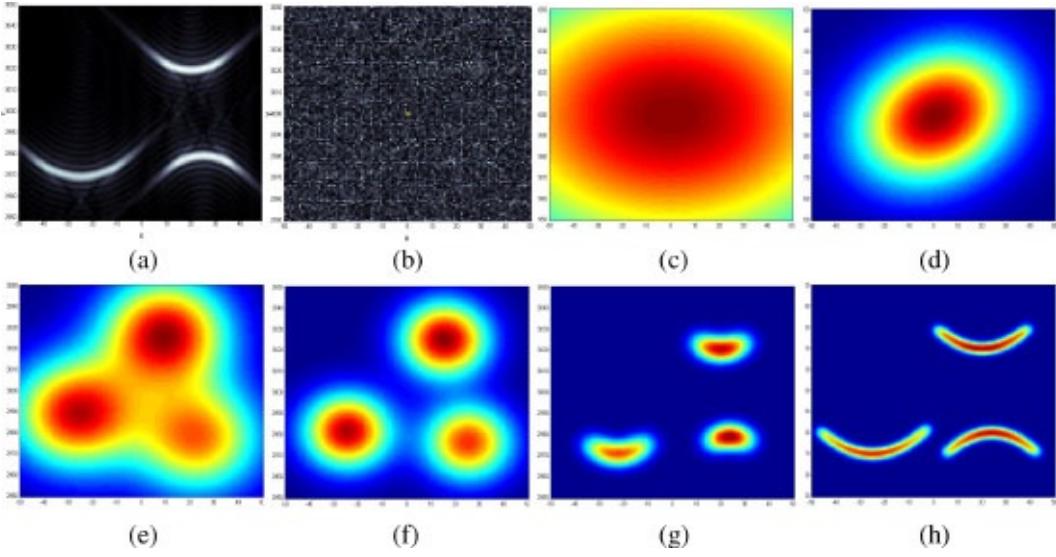


Figure 8.1: Finding ‘smile’ and ‘frown’ patterns in noise, an example of dynamic logic operation: (a) true ‘smile’ and ‘frown’ patterns are shown without noise; (b) actual image available for recognition (signals are below noise, signal-to-noise ratio is between $\frac{1}{2}$ and $\frac{1}{4}$, 100 times lower than usually considered necessary); (c) an initial fuzzy blob-model, the vagueness corresponds to uncertainty of knowledge; (d) through (h) show improved models at various steps of DL [Equations (3) and (4) are solved in 22 steps]. Between stages (d) and (e) the algorithm tried to fit the data with more than one model and decided, that it needs three blob-models to ‘understand’ the content of the data. There are several types of models: one uniform model describing noise (it is not shown) and a variable number of blob-models and parabolic models, which number, location, and curvature are estimated from the data. Until about stage (g) the algorithm ‘thought’ in terms of simple blob models, at (g) and beyond, the algorithm decided that it needs more complex parabolic models to describe the data. Iterations stopped at (h), when similarity (1) stopped increasing.

Several types of pattern models-representations are used in this example: parabolic models describing ‘smiles’ and ‘frown’ patterns (unknown size, position, curvature, signal strength, and number of models), circular-blob models describing approximate patterns (unknown size, position, signal strength, and number of models), and noise model (unknown strength). Mathematical description of these models and corresponding conditional similarities $l(n|m)$ are given in Perlovsky *et al.* (2011).

In this example, the image size is 100×100 points ($N = 10,000$ BU signals, corresponding to the number of receptors in an eye retina), and the true number of models is 4 (3+ noise), which is not known. Therefore, at least $M = 5$ models should be fit to the data, to decide that 4 fits best. This yields complexity of logical combinatorial search, $M^N = 10^{5000}$; this combinatorially large number is much larger than the size of the Universe and the problem was unsolvable for decades. Figure 8.1 illustrates DL operations: (a) true ‘smile’ and ‘frown’ patterns without noise, unknown to the algorithm; (b) actual image available for recognition; (c) through (h) illustrate the DL process, they show improved models (actually $f(m|n)$ values) at various steps of solving DL Equations (3), and (4), a total of 22 steps until similarity continues improving (noise model is not shown; figures (c) through (h) show association variables, $f(m|n)$, for blob and parabolic models). By comparing (h) to (a), one can see that the final states of the models match patterns in the signal. Of course, DL does not guarantee finding any pattern in noise of any strength. For example, if the amount and strength of noise would increase 10-fold, most likely the

patterns would not be found. DL reduced the required number of computations from combinatorial 10^{5000} to about 10^9 . By solving the CC problem, DL was able to find patterns under the strong noise. In terms of signal-to-noise ratio, this example gives 10,000% improvement over the previous state-of-the-art.

The main point of this example is to illustrate the DL process “from vague-to-crisp,” how it models the open-close eyes experiment described in [Section 8.4](#), and how it models visual perception processes demonstrated experimentally in (Bar *et al.*, 2006). This example also emphasizes that DL is a fundamental and revolutionary improvement in mathematics and machine learning (Perlovsky 2009c, 2010c).

8.7. Toward General Models of Structures

The next “breakthrough” required for machine learning and for modeling cognitive processes is to construct algorithms with similar fast learning abilities that do not depend on specific parametric shapes, and that could address the structure of models. I describe such an algorithm in this section. For concreteness, I consider learning situations constructed from some objects among many other objects. I assume that the learning system can identify objects, however, which objects are important for constructing which situation and which objects are randomly present is unknown. Such a problem faces CC of a different kind from that considered above. Logical choice here would have to find structure of models: objects that form situations. Instead, DL concentrates on continuous parameterization of the model structure, which objects are associated with which situations (Ilin and Perlovsky, 2010).

In addition to situation learning, the algorithm given below solves the entire new wide class of problems that could not have been previously solved: structure of texts as they are built from words, structure of interaction between language and cognition, higher cognitive functions, symbols, grounded symbols, perceptual symbol system, creative processes, and the entire cognitive hierarchy (Barsalou, 1999; Perlovsky, 2007a, 2007b; Perlovsky and Ilin, 2012; Perlovsky and Levine, 2012; Perlovsky *et al.*, 2011). Among applications related to learning text structure, I demonstrate autonomous learning of malware codes in Internet messages (Perlovsky and Shevchenko, 2014). Each of these problems if approached logically would result in combinatorial complexity. Apparently, the DL ability to model continuous associations could be the very conceptual breakthrough that brings power of the mind to mathematical algorithms.

This further development of DL turns identifying a structure into a continuous problem. Instead of the logical consideration of a situation as consisting of its objects, so that every object either belongs to a situation or does not, DL considers every object as potentially belonging to a situation. Starting from the vague potentiality of a situation, to which every object could belong, the DL learning process evolves this into a model-actuality containing definite object, and not containing others.

Every observation, $n = 1, \dots, N$, contains a number of objects, $j = 1, \dots, J$. Let us denote objects as $x(n, j)$, here n enumerates observations, and j enumerates objects. As previously, $m = 1, \dots, M$ enumerates situation-models. Model parameters, in addition to $r(m)$ are $p(m, j)$, potentialities of object j belonging to situation-models m . Data $x(n, j)$ have values 0 or 1; potentialities $p(m, j)$ start with vague value near 0.5 and in the DL process of learning they converge to 0 or 1. Mathematically this construct can be described as

$$l(n|m) = \prod_{j=1}^J p(m, j)^{x(n,j)}(1 - p(m, j))^{(1-x(n,j))}. \quad (5)$$

A model parameter $p(m, j)$, modeling a potentiality of object j being part of model m , starts the DL process with initial value near 0.5 (exact values 0.5 for all $p(m, j)$ would be a stationary point of the DL process, Equation (4)). Value $p(m, j)$ near 0.5 gives potentiality values (of $x(n, j)$ belonging to model m) with a maximum near 0.5, in other words, every object has a significant chance to belong to every model. If $p(m, j)$ converge to 0 or 1 values, these would describe which objects j belong to which models m .

Using conditional similarities, Equation (5) in the DL estimation process, Equation (4) can be simplified to an iterative set of Equations (3) and

$$p(m, j) = \sum_n f(m|n)x(n, j) \Bigg/ \sum_{n'} x(n', j); \quad (6)$$

both Equations (3) and (6) should be recomputed at each iteration.

Illustration of this DL algorithm is shown in [Figure 8.2](#). Here, 16,000 simulated observations are shown on the left. They are arranged in their sequential order along the horizontal axis, n . For this simplified example, we simulated 1,000 total possible objects, they are shown along the vertical axis, j . Every observation has or does not have a particular object as shown by a white or black dot at the location (n, j) . This figure looks like random noise corresponding to pseudo-random content of observations. On the right figure, observations are sorted so that observations having similar objects appear next to each other. These similar objects appear as white horizontal streaks and reveal several groups of data, observations of specific situations. Most of observation contents are pseudo-random objects; about one-half of observations contain certain situations. These observations with specific repeated objects reveal specific situations, they identify certain situation-models.

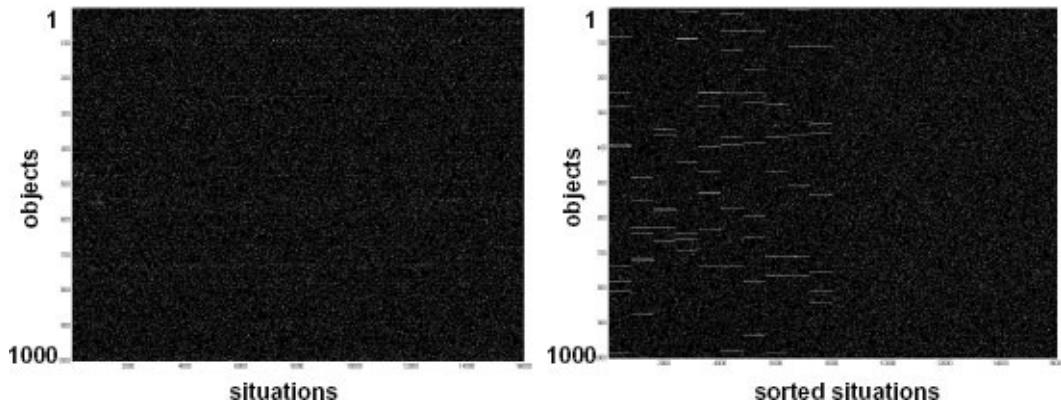


Figure 8.2: On the left are 16,000 observed situations arranged in their sequential order along the horizontal axis, n . The total number of possible objects is 1,000, they are shown along the vertical axis, j . Every observation has or does not have a particular object as shown by a white or black dot at the location (n, j) . This figure looks like random noise corresponding to pseudo-random content of observations. On the right figure, observations are sorted so that observations having similar objects appear next to each other. These similar objects appear as white horizontal streaks. Most of observation contents are pseudo-random objects; about a half of observations have several similar objects. These observations with several specific objects observe specific situations, they reveal certain situation-models.

Since the data for this example have been simulated, we know the true number of

various situations, and the identity of each observation as containing a particular situation-model. All objects have been assigned correctly to their situations without a single error. Convergence is very fast and took two to four iterations (or steps) to solve Equation (4).

This algorithm has been also applied to autonomous finding of malware codes in Internet messages (Perlovsky and Shevchenko, 2014) by identifying groups of messages different from normal messages. In this application, messages are similar to observations in the previous application of situation learning; groups of messages correspond to situations, and n -grams correspond to objects. We applied this algorithm to a publicly available dataset of malware codes, KDD (Dua and Du, 2011; Gesher, 2013; Mugan, 2013). This dataset includes 41 features extracted from Internet packets and one class attribute enumerating 21 classes of four types of attacks. The DL algorithm identified all classes of malware and all malware messages without a single false alarm. This performance in terms of accuracy and speed is better than other published algorithms. An example of the algorithm performance on this data is given in [Figure 8.3](#). This application is a step toward autonomous finding of malware codes in Internet messages.

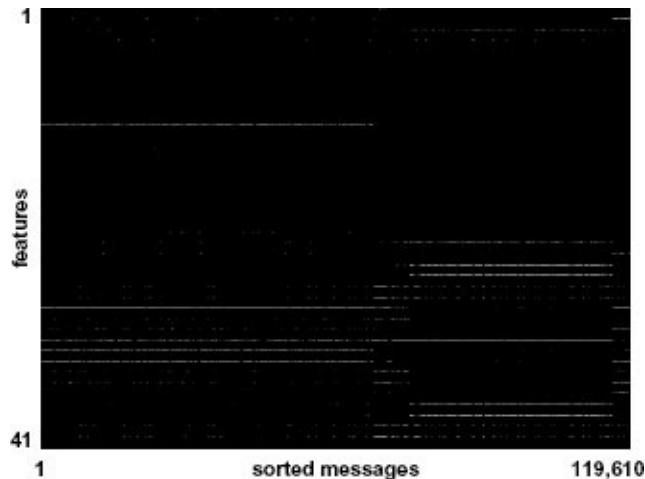


Figure 8.3: This figure shows sorted messages from six groups (one normal and five malware) of the KDD dataset (similar to [Figure 8.2](#) right. I do not show unsorted messages, similar to [Figure 8.2](#) left). Along the horizontal axis: 67,343 normal messages and a five malware codes 2,931 portsweep, 890 warezclient, 3,633 satan, 41,214 neptune, 3,599 ipsweep. Groups of malware are significantly different in size, and not all features important for a group are necessarily present in each vector belonging to the group, therefore the look of the figure is not as clear cut as [Figure 8.2](#) right. Nevertheless all vectors are classified without errors, and without false alarms.

8.8. Modeling Higher Cognitive Functions, Including the Beautiful

Examples in previous sections addressed classical engineering problems that have not been solved for decades. In parallel they modeled the mind processes corresponding to classical psychological phenomena of perception and cognition. In this section, I begin considering “higher” cognitive functions, the entire hierarchy of the mind processes from objects to the very “top” of the mental hierarchy. These processes have not been so far understood in psychology, even so some of their aspects (e.g., language, the beautiful, music) have been analyzed in philosophy and psychology for thousands of years. From the engineering point of view, these problems are becoming important now as engineers attempt to construct human-like robots. As shown in the following sections, constructing robots capable of abstract thinking requires understanding of how cognition interacts with language. I also demonstrate that the beautiful and the music have specific fundamental cognitive functions, and understanding their mechanisms is paramount for constructing human-like robots (even at a level well below than the full power of the human mind).

The mathematical model of learning situations constructed from objects, considered in the previous section, is applicable to modeling the entire hierarchy of the mind. The mind is organized in an approximate hierarchy (Grossberg, 1988) from visual percepts, to objects, to situations, to more and more abstract concepts at higher levels, which contents I analyze later. The mathematical structure of the model given by Equations (3)–(5) is applicable to the entire hierarchy because it does not depend on specific designations of “objects”, “situations”, or “ n -grams”, used above. It is equally applicable to modeling interaction of BU and TD signals between any higher and lower levels in the hierarchy, and therefore modeling neural mechanisms of learning of abstract concepts from lower level concepts.

Let us turn to details of the knowledge instinct and related aesthetic emotions briefly discussed in [Section 8.4](#). Interactions between BU and TD signals are driven by the inborn mechanism, instinct. At lower levels of the hierarchy involving object recognition, this instinct is imperative for finding food, detecting danger, and for survival (Perlovsky, 2007d). Therefore, the knowledge instinct at these levels acts autonomously. Tentative analysis of brain regions participating in the knowledge instinct have been analyzed in (Levine and Perlovsky, 2008, 2010; Perlovsky and Levine, 2012). The mathematical model of the knowledge instinct is given in the previous section, as DL maximization of similarity between mental representations (TD signals) at every level and those coming from a lower level BU signals. Motivation for improving knowledge and satisfaction or dissatisfaction with knowledge are felt as aesthetic emotions. Their connection to knowledge makes them “more spiritual” than basic emotions related to bodily instincts. Existence of these emotions has been experimentally confirmed (Perlovsky *et al.*, 2010). At higher levels of the mental hierarchy a person might experience “higher” aesthetic emotions. These are related to the beautiful as discussed below (Perlovsky, 2000).

Concept-representations at every level emerged in evolution with a specific purpose, to form higher-level more abstract concepts by unifying some subsets of lower-level ones. For example, a mental representation of “professor office” unifies lower level representations such as chairs, desk, computer, shelves, books, etc. At every higher level more general and abstract representations are formed. We know from DL and confirming experiments that higher-level representations are vaguer and less conscious than objects perceived with opened eyes. This vagueness and lesser consciousness is the “price” for generality and abstractness Perlovsky (2010d).

Continuing these arguments toward the top of the hierarchy we can come up with the hypothesis of the contents of representations at the top of the hierarchy. The “top” representation evolved with the purpose to unify the entire life experience and it is perceived as the meaning of life. Does it really exist? Let us repeat that top representations are vague and mostly unconscious, therefore the meaning of life does not exist the same way as a concrete object that can be perceived with opened eyes. Nevertheless, it really exists, similar to other abstract concepts-representation independently of subjective will. It is not up to one’s will to accept or deny an objectively existing architecture of the brain-mind.

Every person can work toward improving one’s understanding of the meaning of his or her life. Appreciation that one’s life has a meaning is of utmost importance; thousands of years ago it could have been important for the individual survival. Today it is important for concentrating ones efforts on the most important goals, for attaining one’s highest achievements. Improving these highest representations, even if it only results in the improved appreciation of existence of the meaning, leads to satisfaction of the knowledge instinct. The corresponding aesthetic emotions near the top of the hierarchy are felt as emotions of the beautiful (Perlovsky, 2000, 2002b, 2010b, 2010e, 2010f, 2014a).

This theory of beautiful is the scientific development of Kant’s aesthetics (1790). Kant has been the first who related the beautiful to knowledge. But without the theory of the knowledge instinct, and the hierarchy of aesthetic emotions, he could not complete his aesthetic theory to his satisfaction. He could only formulate what the beauty “is not”, in particular, he emphasized that the beautiful does not respond to any specific interests, it is purposeful, but its purpose is not related to any specific need. Today I reformulate this: emotions of the beautiful do not satisfy any of the bodily instincts, they satisfy the “higher and more spiritual” instinct for knowledge. Several times Kant has tried to come close to this scientific understanding, he has emphasized that the beautiful is purposive in some highest way, that it corresponds to some highest human interests, and that a better more precise formulation is needed, but he had to concede that “today we cannot” do this. This Kantian intuition was ahead of his time by far. His immediate follower Shiller (1895) misunderstood Kant and interpreted him as if the beautiful is disinterested, and therefore art exists for its own sake. This misunderstanding of the beautiful, its function in cognition, and the meaning of art persists till today. In tens of thousands of papers on art

and aesthetics, the beautiful is characterized as disinterested, and “art for its own sake” (Perlovsky, 2010f).

I would add that often emotions of the beautiful are mixed up with sexual feelings. Of course, sex is among the most powerful instincts, it may involve all our abilities. However, emotions related to sex are driven by the instinct for procreation and therefore they fundamentally differ from the beautiful, which is driven by the instinct for knowledge.

Let us summarize the emotion of the beautiful as an aesthetic emotion related to the satisfaction of the knowledge instinct near the top of the mental hierarchy, related to the understanding of the meaning of life. It is a subjective emotion affected by the entire individual life experience, at the same time it objectively exists, being related to the fundamental structure of the human mind.

8.9. Language, Cognition, and Emotions Motivating their Interactions

Many properties of cognition and language have been difficult to understand because interactions between these two human abilities have not been understood. Do we think with language, or is language used only for communicating completed thoughts? Language and thinking are so closely related and intertwined that answering this question, requires a mathematical model corresponding to the known neural structures of the brain-mind. For many years, mathematical modeling of language and cognition has proceeded separately, without neurally-based mathematical analysis of how these abilities interact. So, we should not be surprised that existing robots are far away from human-like abilities for language and cognition. Constructing humanoid robots requires mathematical models of these abilities. A model of language-cognition interaction described in this section gives a mathematical foundation for understanding these abilities and for developing robots with human-like abilities. It also gives a foundation for understanding why higher human cognitive abilities, including ability for the beautiful sometimes may seem mysterious (Perlovsky, 2004, 2005, 2009a).

A cognitive hierarchy of the mind has been illustrated in [Figure 8.4](#). However, analysis in this section demonstrates that such a hierarchy cannot exist without language. The human mind requires a dual hierarchy of language-cognition illustrated in [Figure 8.5](#) (Perlovsky, 2005, 2007a, 2009a, 2009b, 2011, 2013b; Perlovsky and Ilin, 2010, 2013; Tikhanoff *et al.*, 2006). The dual hierarchy model explains many facts about thinking, language, and cognition, which has remained unexplainable and would be considered mysteries, if not so commonplace. Before we describe how the dual hierarchy is modeled by equations discussed in [Section 8.6](#), let us list here some of the dual model explanations and predictions (so that the mathematical discussion later will be associated with specific human abilities).

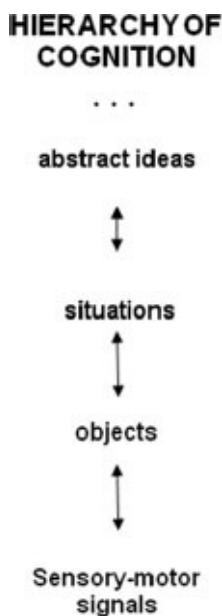


Figure 8.4: The hierarchy of cognition from sensory-motor signals at the “lower” levels to objects, situations, and more abstract concepts. Every level contains a large number of mental representations. Vertical arrows indicate interactions of

BU and TD signals.

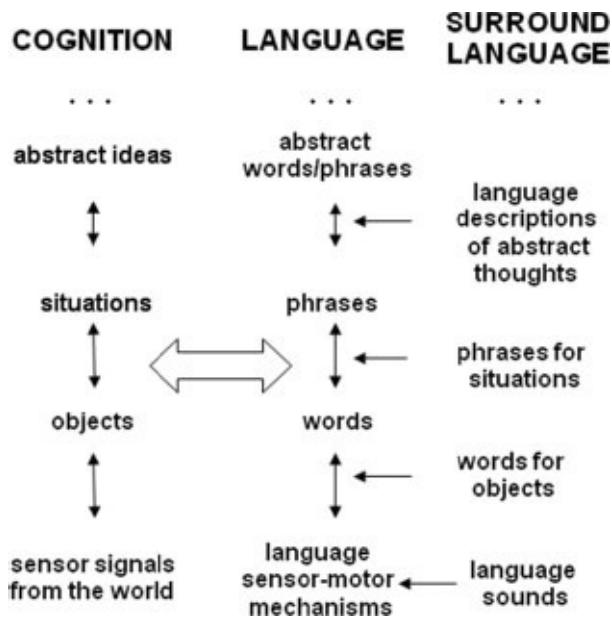


Figure 8.5: The dual hierarchy. Language and cognition are organized into approximate dual hierarchy. Learning language is grounded in the surrounding language throughout the hierarchy (indicated by thin horizontal arrows). Learning the cognitive hierarchy is grounded in experience only at the very “bottom.” The rest of the cognitive hierarchy is mostly grounded in language. Vertical arrows indicate interactions of BU and TD signals. A wide horizontal arrow indicates interactions between language and cognition; for abstract concepts these are mostly directed from language to cognition.

(1) The dual model explains functions of language and cognition in thinking: cognitive representations model surrounding world, relations between objects, events, and abstract concepts. Language stores culturally accumulated knowledge about the world, yet language is not directly connected to objects, events, and situations in the world. Language guides acquisition of cognitive representations from random percepts and experiences, according to what is considered worth learning and understanding in culture. Events that are not described in language are likely not even noticed or perceived in cognition.

(2) Whereas language is acquired early in life, acquiring cognition takes a lifetime. The reason is that language representations exist in surrounding language “ready-made,” acquisition of language requires only interaction with language speakers, but does not require much life experience. Cognition on the opposite requires life experience.

(3) This is the reason why abstract words excite only language regions of brain, whereas concrete words excite also cognitive regions (Binder *et al.*, 2005). The dual model predicts that abstract concepts are often understood as word descriptions, but not in terms of objects, events, and relations among them.

(4) In this way, the dual model explains why children can acquire the entire hierarchy of language including abstract words without experience necessary for understanding them.

(5) DL is the basic mechanism for learning language and cognitive representations.

The dual model suggests that language representations become crisp after language is learned (5–7 years of age), this corresponds to language representations being crisp and near logical. However, cognitive representations remain vague for two reasons. First, as we have discussed, this vagueness is necessary for the ability to perceive objects and events in their variations. Second, this vagueness is a consequence of limited experience; ability to identify cognitive events and abstract concepts in correspondence with language improves with experience; the vagueness is also the meaning of “continuing learning”, this takes longer for more abstract and less used concepts. How do these two different aspects of vagueness co-exist? Possibly, various concrete representations are acquired with experience; vague representations are still retained for perception of novelty. At lower levels, this occurs automatically, at higher levels individual efforts are necessary to maintain both concrete and vague representations (we know that some minds get “closed” with experience).

(6) The dual model gives mathematical description of the recursion mechanism (Perlovsky and Ilin, 2012). Whereas Hauser *et al.* (2002) postulate that recursion is a fundamental mechanism in cognition and language, the dual model suggests that recursion is not fundamental, hierarchy is a mechanism of recursion.

(7) Another mystery of human-cognition, not addressed by cognitive or language theories, is basic human irrationality. This has been widely discussed and experimentally demonstrated following discoveries of Tversky and Kahneman (1974), leading to the 2002 Nobel Prize. According to the dual hierarchy model, the “irrationality” originates from the dichotomy between cognition and language. Language is crisp and conscious while cognition might be vague and ignored when making decisions. Yet, collective wisdom accumulated in language may not be properly adapted to one’s personal circumstances, and therefore be irrational in a concrete situation. In the 12th century, Maimonides wrote that Adam was expelled from paradise because he refused original thinking using his own cognitive models, but ate from the tree of knowledge and acquired collective wisdom of language (Levine and Perlovsky, 2008).

The same Equations (4) or (3) and (5) that we have used to model the cognitive hierarchy are used for the mathematical modeling of the dual language-cognition model. Modeling the dual hierarchy differs from modeling cognition in that every observation, situation-collection of lower level signals, now includes both language and cognitive representations from lower levels. In the DL processes [Equations (4) and (5)], language hierarchy is acquired first (during early years of life); it is learned according to abstract words, phrases, and higher-level language representations existing in the surrounding language. Most cognitive representations remain vague for a while. As more experience is accumulated, cognitive representations are acquired corresponding to already existing language representations. This joint acquisition of both language and cognition corresponds to the wide horizontal arrow in [Figure 8.5](#). In this way, cognitive representations are acquired from experience guided by language.

As some cognitive representations become crisper and more conscious, this establishes more reliable connections between representation and events in the world. This provides learning feedback, grounding for learning of both cognition and language. Thus, both cognition and language are grounded not only in language, but also in the surrounding world. Language representations correspond to some extent to the world. At lower levels of the hierarchy, levels of objects and situations, these processes are supported by the fact that objects can be directly observed; situations can be observed to some extent after acquisition of the corresponding cognitive ability. At more abstract levels, these connections between language and the world are more sporadic, some connections of abstract language and cognitive representations could be more or less conscious, correspondingly more or less concrete could be understanding of abstract events and relations in the world.

Higher up in the hierarchy, less of cognitive representations ever become fully conscious, if they maintain an adequate level of generality and abstractness. Two opposing mechanisms are at work here (as already mentioned). First, vagueness of representations is a necessary condition for being able to perceive novel contents (we have seen it in the “open–close” eyes experiment). Second, vague contents evolve toward concrete contents with experience. Therefore, more concrete representations evolve with experience, while vague representations remain. This depends on experience and extent to which cultural wisdom contained in language is acquired by an individual. People differ in extents and aspects of culture they become conscious of at the higher levels. Majority of cognitive contents of the higher abstract concepts remain unconscious. This is why we can discuss in detail the meaning of life and the beautiful using language, but significant part of higher cognitive contents forever remain inaccessible to consciousness.

Interactions among language, cognition, and the world require motivation. The inborn motivation is provided by the knowledge instinct. Language acquisition is driven by its aspect related to the language instinct (Pinker, 1994). Certain mechanisms of mind may participate in both language and knowledge instinct, and division between language and knowledge instinct is not completely clear-cut. This area remains “grey” in our contemporary knowledge. More experimental data are needed to differentiate the two. I would emphasize that the language instinct drives language acquisition, its mechanisms “stop” at the border with cognition, and the language instinct does not concern improvement of cognitive representations and connecting language representations with the world.

Mechanisms of the knowledge instinct connecting language and cognition are experienced as emotions. These are special aesthetic emotions related to language. As we have discussed, these interactions are mostly directed from language to cognition, and correspondingly these emotions–motivations necessary for understanding language cognitively (beyond “just” words) are “more” on the language side. For cognitive mechanisms to be “interested” in language, to be motivated to acquire directions from

language, there has to be an emotional mechanism in language accomplishing this motivation. These emotions of course must be of ancient origin, they must have been supporting the very origin of language and originating pre-linguistically. These emotions are emotional prosody of language.

In the pre-linguistic past, animal vocalizations did not differentiate emotion-evaluation contents from conceptual-semantic contents. Animals' vocal tract muscles are controlled from the ancient emotional center (Deacon, 1989; Lieberman, 2000). Sounds of animal cries engage the entire psyche, rather than concepts and emotions separately. Emergence of language required emancipation of voicing from uncontrollable emotions (Perlovsky, 2004, 2009a, 2009b). Evolution of languages proceeded toward reducing emotional content of language voice. Yet, complete disappearance of emotions from language would make language irrelevant for life, meaningless. Connections of language with cognition and life require that language utterances remain emotional. This emotionality motivates connecting language with cognition. Some of these prosodial emotions correspond to basic emotions and bodily needs; yet there is a wealth of subtle variations of prosodial emotions related to the knowledge instinct and unrelated to bodily needs. I would add that the majority of everyday conversations may not relate to exchange of semantic information as in scientific presentations; majority of information in everyday conversations involve emotional information, e.g., information about mutual compatibility among people. This wealth of emotions we hear in poetry and in songs.

Emotionality of language prosody differs among languages; this impacts entire cultures. It is interesting to note that during the recent 500 years, during the transition from Middle English to Modern English, significant part of emotional connections between words and their meanings has been lost. I repeat, these emotions came from millennial past and subconsciously control the meanings of utterances and thoughts; disappearance (to significant extent) of these emotions makes English a powerful tool of science and engineering, including social engineering. There are differences between these areas of fast changes in contemporary life. Results of scientific and engineering changes, such as new drugs, new materials, and new transportation methods usually are transparent for society. Positive changes are adopted, when in doubt (such as genetic engineering) society can take evaluative measures and precautions. Social engineering is different, changes in values and attitudes usually are considered to be results of contemporary people to be smarter in their thinking than our predecessors. Identifying which part of these changes is due to autonomous changes in language, which are not under anybody's conscious control, should be an important part of social sciences.

The dual hierarchy model explains how language influences cognition. A more detailed development of this model can lead to explaining and predicting cultural differences related to language, so called Sapir-Whorf Hypothesis (SWH) (Whorf, 1956; Boroditsky, 2001). Prosodial emotions are influenced by grammar, leading to Emotional SWH (Perlovsky, 2009b; Czerwon, *et al.*, 2013). It follows that languages influence

emotionalities of cultures, in particular, the strength of emotional connections between words and their meanings. This strength of individual “belief” in the meaning of words can significantly influence cultures and their evolution paths.

Understanding prosodial emotions and their functions in cognition, developing appropriate mathematical models is important not only for psychology and social science but also for artificial intelligence and ANNs, for developing human– computer interface and future robots with human level intelligence.

8.10. Music Functions in Cognition

2400 years ago Aristotle (1995) asked “why music, being just sounds, reminds states of the soul?” Why an ability to perceive sounds emotionally and to create such sounds could emerge in evolution? Darwin (1871) called music “the greatest mystery”. Does computational intelligence community need to know the answer? Can we contribute to understanding of this mystery? The explanation of the mystery of music has been obtained from the dual model considered in the previous section. Music turns out to perform cognitive functions of utmost importance, it enables accumulation of knowledge, it unifies human psyche split by language, and it makes the entire evolution of human culture possible. Enjoying music is not just good for spending time free from job; enjoying music is fundamental for cognition. Humanoid robots need to enjoy music, otherwise cognition is not possible. To understand cognitive functions of music, music origin and evolution, we first examine an important cognitive mechanism counteracting the knowledge instinct, the mechanism of cognitive dissonances.

Cognitive dissonances (CD) are discomforts caused by holding conflicting cognitions. Whereas it might seem to scientists and engineers that conflicts in knowledge are welcome as they inspire new thinking, in fact CD are particularly evident when a new scientific theory is developed; new ideas are usually rejected. It is not just because of opposition of envious colleagues, it is also because of genetically inherited mechanism of CD. CD is among “the most influential and extensively studied theories in social psychology” (e.g., Alfnes *et al.*, 2010). CDs are powerful anti-knowledge mechanisms. It is well known that CD discomforts are usually resolved by devaluing and discarding a conflicting piece of knowledge (Festinger, 1957; Cooper, 2007; Harmon-Jones *et al.*, 2009); we discuss it in detail later. It is also known that awareness of CD is not necessary for actions to reduce the conflict (discarding conflicting knowledge); these actions are often fast and act without reaching conscious mechanisms (Jarcho *et al.*, 2011).

I would emphasize that every mundane element of knowledge to be useful must differ from innate knowledge supplied by evolution or from existing knowledge acquired through experience. Otherwise the new knowledge would not be needed. For new knowledge to be useful it must to some extent contradict existing knowledge. Can new knowledge be complementary rather than contradictory? Since new knowledge emerges by modifying previous knowledge (Simonton, 2000; Novak, 2010), there must always be conflict between the two. Because of this conflict between new and previous knowledge CD theory suggests that new knowledge should be discarded. This process of resolving CD by discarding contradictions is usually fast, and according to CD theory new knowledge is discarded before its usefulness is established. But accumulating knowledge is the essence of cultural evolution, so how human cultures could evolve? A powerful cognitive-emotional mechanism evolved to overcome CD, discomforts of contradictory knowledge, so that human evolution became possible.

A language phrase containing a new piece of knowledge, in order to be listened to without an immediate rejection, should come with a sweetener, a positive emotion sounding in the voice itself. In the previous section, we discussed that this is the purpose of language prosody. However, emotions of language have been reduced in human evolution, whereas knowledge has been accumulated and stronger emotions have been needed to sustain the evolution. The human mind has been rewired for this purpose. Whereas animal vocalization is controlled from the ancient emotional center, the human mind evolved secondary emotional centers in the cortex, partially under voluntary control, governing language vocalization. In the process of language evolution human vocalization split into highly semantic and low emotional language, and another type of vocalization, highly emotional and less semantic, connected to the primordial emotional center governing the entire psyche; this highly emotional vocalization unifying psyche gradually evolved into music. The powerful emotional mechanism of music overcame cognitive dissonances, it enables us and our predecessors to maintain contradictory cognitions and unify psyche split by language (Perlovsky, 2006a, 2010a, 2012a, 2012b, 2013a).

Pre-language animals' mind is unified. A monkey seeing an approaching tiger understands the situation conceptually (danger), is afraid emotionally (fear), behaves appropriately (jumps on a tree) and cries in monkey language for itself and for the rest of the pack ("tiger", in monkey language). However, all of these are experienced as a unified state of the mind, a monkey cannot experience emotions and concepts separately. A monkey does not contemplate the meaning of his life. Humans, in contrast, possess a remarkable degree of differentiation of their mental states. Emotions in humans have separated from concepts and from behavior. This differentiation destroyed the primordial unity of the psyche. With the evolution of language the human psyche lost its unity—the inborn connectedness of knowledge, emotions, and behavior. The meaning of our life, the highest purpose requiring the utmost effort is not obvious and not defined for us instinctively; the very existence of the meaning is doubtful. However, the unity of psyche is paramount for concentrating the will, and for achieving the highest goal. While part of the human voice evolved into language, acquired concrete semantics, lost much of its emotionality, and split our mental life into many disunited goals, another part of the voice evolved into a less concretely semantic but powerfully emotional ability—music—helping to unify the mental life (Perlovsky, 2006a, 2012a, 2012b, 2013a).

8.11. Theoretical Predictions and Experimental Tests

Mathematical models of the mind have been used for designing cognitive algorithms; these algorithms based on DL solved several classes of engineering problems considered unsolvable, reaching and sometimes exceeding power of the mind. A few examples have been discussed in this chapter, many others can be found in the referenced publications (e.g., Perlovsky *et al.*, 2011). These engineering successes suggest that DL possibly captures some essential mechanisms of the mind making it more powerful than past algorithms. In addition to solving engineering problems, mathematical models of the mind have explained mechanisms that could not have been understood and made predictions that could be tested experimentally. Some of these predictions have been tested and confirmed, none has been disproved. Experimental validations of DL predictions are summarized below.

The first and most fundamental prediction of dynamic logic is vagueness of mental representations and the DL process “from vague to crisp”. This prediction has been confirmed in brain imaging experiments for perception of objects (Bar *et al.*, 2006), and for recognition of contexts and situations (Kveraga *et al.*, 2007). DL is a mechanism of the knowledge instinct. The knowledge instinct theory predicts the existence of special emotions related to knowledge, aesthetic emotions. Their existence has been demonstrated in (Perlovsky *et al.*, 2010). This also confirms of existence of the knowledge instinct.

The dual hierarchy theory of language-cognition interaction predicts that language and cognition are different mechanisms. Existence of separate neural mechanisms for language and cognition has been confirmed in (Price, 2012). The dual hierarchy theory predicts that perception and understanding of concrete objects involves cognitive and language mechanisms, whereas cognition of abstract concepts is mostly due to language mechanisms. This has been confirmed in (Binder *et al.*, 2005).

The dual hierarchy emphasizes fundamental cognitive function of language emotionality, language prosody influences strength of emotional connections between sounds and meanings; these connections are the foundations of both language and cognition. The strengths of these emotional connections could differ among languages, affecting cultures and their evolution, leading to Emotional SWH. Various aspects of this hypothesis have been confirmed experimentally. Gutfreund (1990) demonstrated that Spanish is more emotional than English; Harris *et al.* (2003) demonstrated that emotional connections between sounds and meanings in Turkish are stronger than in English; these have been predicted based on grammatical structures of the corresponding languages (Perlovsky, 2009b). Czerwon *et al.* (2013) demonstrated that this emotional strength depends on grammar as predicted in Perlovsky (2009b).

A theory of musical emotions following from the dual model predicts that music helps unifying mental life, overcoming CD and keeping contradictory knowledge. This prediction has been confirmed in (Masataka and Perlovsky, 2012a, 2012b), which

describes a modified classical CD experiment (Aronson and Carlsmith, 1963); in the original experiment children devalued a toy if they were told not to play with it. The desire ‘to have’ contradicts the inability ‘to attain’; this CD is resolved by discarding the value of a toy. Masataka and Perlovsky modified this experiment by adding music played in the background. With music, the toy has not been devalued, contradictory knowledge could be retained. Another experiment explained the so-called Mozart effect: student’s academic test performance improved after listening to Mozart (Rauscher *et al.*, 1993). These results started a controversy resolved in (Perlovsky *et al.*, 2013). This publication demonstrated that the Mozart effect is the predicted overcoming of CD: as expected from CD theory students allocate *less* time to more difficult and stressful tests; with music in the background students *can* tolerate stress, allocate *more* time to stressful tests, and improve grades. These results have been further confirmed in (Cabanac *et al.*, 2013): students selecting music classes outperformed other students in all subjects. Another experiment demonstrated that music helps overcoming cognitive interference. A classical approach to creating cognitive interference is Stroop effect (Stroop, 1935). Masataka and Perlovsky (2013) demonstrated that music played in the background reduced the interference.

8.12. Future Research

Future engineering research should develop the large-scale implementation of the dual hierarchy model and demonstrate joint learning of language and cognition in robotics and human-computer integrated systems. Scientifically oriented research should use agent systems with each agent possessing a mind with language and cognitive abilities (the dual hierarchy). Such systems could be used for studying evolution of languages including grammar along with evolution of cultures. This will open directions toward studying evolution of cognitive dissonances and aesthetic emotions. Future research should address brain mechanisms of the knowledge instinct. A particularly understudied area is functions of emotions in cognition. Several directions for research in this area are outlined below.

Emotions of language prosody have only been studied in cases of intentionally strongly-emotional speech. These emotions are recognized without language in all societies, and unify us with non-human animals. Non-intentional prosodic emotions that are inherent functions of languages and connect language sounds to meanings should be studied. The dual model predicts fundamental importance of these emotions in language-cognition interaction.

Emotional SWH made a number of predictions about prosodic emotions and their connections to language grammar. In particular, this theory predicts that emotional links between sounds and meanings (words and objects-events they designate) are different in different languages, and the strengths of these links depend on language grammar in a predictable way (Perlovsky, 2009b): languages with more inflections have stronger emotional links between sounds and meanings. These predictions have been confirmed for English, Spanish, and Turkish (Guttfreund, 1990; Harris *et al.*, 2003; Czerwon, *et al.*, 2013). More experiments are needed to study connections between language structure, its emotionality, and cultural effects.

Among the greatest unsolved problems in experimental studies of the mind is how to measure the wealth of aesthetic emotions. Bonniot-Cabanac *et al.* (2012) initiated experimental studying of CD emotions. These authors demonstrated existence of emotions related to CD, their fundamental difference from basic emotions, and outlined steps toward demonstrating a very large number of these emotions. Possibly, a most direct attempt to “fingerprint” aesthetic emotions is to measure neural networks corresponding to them (Wilkins *et al.*, 2012). Three specific difficulties are faced when attempting to instrumentalize (measure) a variety of aesthetic emotions. The first is a limitation by words. In the majority of psychological experiments measuring emotions, they are described using words. But English words designate a limited number of different emotions, about 150 words designate between 6 and 20 different emotions (depending on the author; e.g., Petrov *et al.*, 2012). Musical emotions that evolved with a special purpose not to be limited by low emotionality of language cannot be measured in all their wealth by emotional words. Second difficulty is “non-rigidity” of aesthetic emotions. They are

recent in evolutionary time. Whereas basic emotions evolved hundreds of millions of years ago, aesthetic emotions evolved no more than two million years ago, and possibly more recently. This might be related to the third difficulty, subjectivity. Aesthetic emotions depend not only on stimuli, but also on subjective states of experiment participants. All experimental techniques today rely on averaging over multiple measurements and participants. Such averaging likely eliminates the wealth of aesthetic emotions, such as musical emotions, and only most “rigid” emotions remain.

This chapter suggests that mental representations near the top of the hierarchy of the mind are related to the meaning of life. Emotions related to improvement of the contents of these representations are related to the emotions of the beautiful. Can this conjecture be experimentally confirmed? Experimental steps in this direction have been made in (Biederman and Vessel, 2006; Zeki *et al.*, 2014).

References

- Alfnæs, F., Yue, C. and Jensen, H. H. (2010). Cognitive dissonance as a means of reducing hypothetical bias. *Eur. Rev. Agric. Econ.*, 37(2), pp. 147–163.
- Aronson, E. and Carlsmith, J. M. (1963). Effect of the severity of threat on the devaluation of forbidden behavior. *J. Abnor. Soc. Psych.*, 66, 584–588.
- Aristotle. (1995). *The Complete Works: The Revised Oxford Translation*, Barnes, J. (ed.). Princeton, NJ, USA: Princeton University Press.
- Bar, M., Kassam, K. S., Ghuman, A. S., Boshyan, J., Schmid, A. M., Dale, A. M., Hämäläinen, M. S., Marinkovic, K., Schacter, D. L., Rosen, B. R. and Halgren, E. (2006). Top-down facilitation of visual recognition. *Proc. Natl. Acad. Sci. USA*, 103, pp. 449–54.
- Barsalou, L. W. (1999). Perceptual symbol systems. *Behav. Brain Sci.*, 22, pp. 577–660.
- Bellman, R. E. (1961). *Adaptive Control Processes*. Princeton, NJ: Princeton University Press.
- Biederman, I. and Vessel, E. (2006). Perceptual pleasure and the brain. *Am. Sci.*, 94(3), p. 247. doi: 10.1511/2006.3.247.
- Binder, J. R., Westbury, C. F., McKiernan, K. A., Possing, E. T. and Medler, D. A. (2005). Distinct brain systems for processing concrete and abstract concepts. *J. Cogn. Neurosci.*, 17(6), pp. 1–13.
- Bonniot-Cabanac, M.-C., Cabanac, M., Fontanari, F. and Perlovsky, L. I. (2012). Instrumentalizing cognitive dissonance emotions. *Psychol.*, 3(12), pp. 1018–1026. <http://www.scirp.org/journal/psych>.
- Boroditsky, L. (2001). Does language shape thought? Mandarin and English speakers' conceptions of time. *Cogn. Psychol.*, 43(1), pp. 1–22.
- Bryson, A. E. and Ho, Y.-C. (1969). *Applied Optimal Control: Optimization, Estimation, and Control*. Lexington, MA: Xerox College Publishing, p. 481.
- Cabanac, A., Perlovsky, L. I., Bonniot-Cabanac, M.-C. and Cabanac, M. (2013). Music and academic performance. *Behav. Brain Res.*, 256, pp. 257–260.
- Carpenter, G. A. and Grossberg, S. (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Comput. Vis., Graph., Image Process.*, 37, pp. 54–115.
- Cooper, J. (2007). *Cognitive Dissonance: 50 Years of a Classic Theory*. Los Angeles, CA: Sage.
- Czerwon, B., Hohlfeld, A., Wiese, H. and Werheid, K. (2013). Syntactic structural parallels in?uence processing of positive stimuli: Evidence from cross-modal ERP priming. *Int. J. Psychophysiol.*, 87, pp. 28–34.
- Darwin, C. R. (1871). *The Descent of Man, and Selection in Relation to Sex*. London: John Murray Publishing House.
- Deacon, T. W. (1989). The neural circuitry underlying primate calls and human language. *Human Evol.*, 4(5), pp. 367–401.
- Dua, S. and Du, X. (2011). *Data Mining and Machine Learning in Cybersecurity*. Boca Raton, FL: Taylor & Francis.
- Duda, R. O., Hart, P. E. and Stork, D. G. (2000). *Pattern Classification, Second Edition*. New York: Wiley-Interscience.
- Festinger, L. (1957). *A Theory of Cognitive Dissonance*. Stanford CA: Stanford University Press.
- Friedman, J., Hastie, T. and Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors). *Ann. Statist.*, 28(2), pp. 337–655.
- Gödel, K. (2001). *Collected Works, Volume I, “Publications 1929–1936”*, Feferman, S., Dawson, Jr., J. W., and Kleene, S. C. (eds.). New York, USA: Oxford University Press.
- Grossberg, S. (1988). *Neural Networks and Natural Intelligence*. Cambridge: MIT Press.
- Grossberg, S. and Levine, D. S. (1987). Neural dynamics of attentionally modulated Pavlovian conditioning: blocking, inter-stimulus interval, and secondary reinforcement. *Psychobiol.*, 15(3), pp. 195–240.
- Gutfreund D. G. (1990). Effects of language usage on the emotional experience of Spanish–English and English–Spanish bilinguals. *J. Consult. Clin. Psychol.*, 58, pp. 604–607.
- Harmon-Jones, E., Amodio, D. M. and Harmon-Jones, C. (2009). Action-based model of dissonance: a review, integration, and expansion of conceptions of cognitive conflict. In M. P. Zanna (ed.), *Adv. Exp. Soc. Psychol.*,

- Burlington: Academic Press, 41, 119–166.
- Harris, C. L., Ayçiçegi, A. and Gleason, J. B. (2003). Taboo words and reprimands elicit greater autonomic reactivity in a first language than in a second language. *Appl. Psycholinguist.*, 24, pp. 561–579.
- Hauser, M. D., Chomsky, N. and Fitch, W. T. (2002). The faculty of language: what is it, who has it, and how did it evolve? *Science*, 298, pp. 1569–1570. doi: 10.1126/science.298.5598.156.
- Hilbert, D. (1928). The foundations of mathematics. In van Heijenoort, J. (ed.), *From Frege to Gödel*. Cambridge, MA, USA: Harvard University Press, p. 475.
- Hinton, G., Deng, L., Yu, D., Dahl, G. E. and Mohamed, A. (2012). Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. *IEEE Signal Process. Mag.*, 29(6).
- Ilin, R. and Perlovsky, L. I. (2010). Cognitively inspired neural network for recognition of situations. *Int. J. Nat. Comput. Res.*, 1(1), pp. 36–55.
- Jarcho, J. M., Berkman, E. T. and Lieberman, M. D. (2011). The neural basis of rationalization: cognitive dissonance reduction during decision-making. *SCAN*, 6, pp. 460–467.
- Kant, I. (1790). *The Critique of Judgment*, Bernard, J. H. (Trans.). Amherst, NY: Prometheus Books.
- Kveraga, K., Boshyan, J. and Bar, M. (2007). Magnocellular projections as the trigger of top-down facilitation in recognition. *J. Neurosci.*, 27, pp. 13232–13240.
- Levine, D. S. and Perlovsky, L. I. (2008). Neuroscientific insights on biblical myths: simplifying heuristics versus careful thinking: scientific analysis of millennial spiritual issues. *Zygon, J. Sci. Religion*, 43(4), pp. 797–821.
- Levine, D. S. and Perlovsky, L. I. (2010). Emotion in the pursuit of understanding. *Int. J. Synth. Emotions*, 1(2), pp. 1–11.
- Lieberman, P. (2000). *Human Language and Our Reptilian Brain*. Cambridge: Harvard University Press.
- Masataka, N. and Perlovsky, L. I. (2012a). Music can reduce cognitive dissonance. *Nature Precedings*: hdl:10101/npre.2012.7080.1. <http://precedings.nature.com/documents/7080/version/1>.
- Masataka, N. and Perlovsky, L. I. (2012b). The efficacy of musical emotions provoked by Mozart's music for the reconciliation of cognitive dissonance. *Scientific Reports* 2, Article number 694. doi:10.1038/srep00694. <http://www.nature.com/srep/2013/130619/srep02028/full/srep02028.html>.
- Masataka, N. and Perlovsky, L. I. (2013). Cognitive interference can be mitigated by consonant music and facilitated by dissonant music. *Scientific Reports* 3, Article number 2028. doi:10.1038/srep02028. <http://www.nature.com/srep/2013/130619/srep02028/full/srep02028.html>.
- Minsky, M. (1968). *Semantic Information Processing*. Cambridge, MA: MIT Press.
- Meier, U. and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 16–21 June, Providence, R1, pp. 3642–3649.
- Mengersen, K., Robert, C. and Titterington, M. (2011). *Mixtures: Estimation and Applications*. New York: Wiley.
- Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization, Second Edition*. Berlin, Heidelberg: Springer.
- Novak, J. D. (2010). Learning, creating, and using knowledge: concept maps as facilitative tools in schools and corporations. *J. e-Learn. Knowl. Soc.*, 6(3), pp. 21–30.
- Perlovsky, L. I. (1998). Conundrum of combinatorial complexity. *IEEE Trans. PAMI*, 20(6), pp. 666–670.
- Perlovsky, L. I. (2000). Beauty and mathematical intellect. *Zvezda*, 9, pp. 190–201 (Russian).
- Perlovsky, L. I. (2001). *Neural Networks and Intellect: Using Model-Based Concepts*. New York: Oxford University Press.
- Perlovsky, L. I. (2002a). Physical theory of information processing in the mind: concepts and emotions. *SEED On Line J.*, 2(2), pp. 36–54.
- Perlovsky, L. I. (2002b). Aesthetics and mathematical theories of intellect. *Iskusstvoznanie*, 2(2), pp. 558–594.
- Perlovsky, L. I. (2004). Integrating language and cognition. *IEEE Connect.*, 2(2), pp. 8–12.
- Perlovsky, L. I. (2005). Evolving agents: communication and cognition. In Gorodetsky, V., Liu, J. and Skormin, V. A. (eds.), *Autonomous Intelligent Systems: Agents and Data Mining*. Berlin, Heidelberg, Germany: Springer-Verlag,

pp. 37–49.

- Perlovsky, L. I. (2006a). Toward physics of the mind: concepts, emotions, consciousness, and symbols. *Phys. Life Rev.*, 3(1), pp. 22–55.
- Perlovsky, L. I. (2006b). Fuzzy dynamic logic. *New Math. Nat. Comput.*, 2(1), pp. 43–55.
- Perlovsky, L. I. (2007a). Symbols: integrated cognition and language. In Gudwin, R. and Queiroz, J. (eds.), *Semiotics and Intelligent Systems Development*. Hershey, PA: Idea Group, pp. 121–151.
- Perlovsky, L. I. (2007b). Modeling field theory of higher cognitive functions. In Loula, A., Gudwin, R. and Queiroz, J. (eds.), *Artificial Cognition Systems*. Hershey, PA: Idea Group, pp. 64–105.
- Perlovsky, L. I. (2007c). The mind vs. logic: Aristotle and Zadeh. *Soc. Math. Uncertain. Crit. Rev.*, 1(1), pp. 30–33.
- Perlovsky, L. I. (2007d). Neural dynamic logic of consciousness: the knowledge instinct. In Perlovsky, L. I. and Kozma, R. (eds.), *Neurodynamics of Higher-Level Cognition and Consciousness*. Heidelberg, Germany: Springer Verlag, pp. 73–108.
- Perlovsky, L. I. (2009a). Language and cognition. *Neural Netw.*, 22(3), pp. 247–257. doi:10.1016/j.neunet.2009.03.007.
- Perlovsky, L. I. (2009b). Language and emotions: emotional Sapir–Whorf hypothesis. *Neural Netw.*, 22(5–6), pp. 518–526. doi:10.1016/j.neunet.2009.06.034.
- Perlovsky, L. I. (2009c). ‘Vague-to-crisp’ neural mechanism of perception. *IEEE Trans. Neural Netw.*, 20(8), pp. 1363–1367.
- Perlovsky, L. I. (2010a). Musical emotions: functions, origin, evolution. *Phys. Life Rev.*, 7(1), pp. 2–27. doi:10.1016/j.plrev.2009.11.001.
- Perlovsky, L. I. (2010b). Intersections of mathematical, cognitive, and aesthetic theories of mind, *Psychol. Aesthetics, Creat., Arts*, 4(1), pp. 11–17. doi:10.1037/a0018147.
- Perlovsky, L. I. (2010c). Neural mechanisms of the mind, Aristotle, Zadeh, & fMRI. *IEEE Trans. Neural Netw.*, 21(5), pp. 718–33.
- Perlovsky, L. I. (2010d). The mind is not a kludge. *Skeptic*, 15(3), pp. 51–55.
- Perlovsky L. I. (2010e). Physics of the mind: concepts, emotions, language, cognition, consciousness, beauty, music, and symbolic culture. *WebmedCentral Psychol.*, 1(12), p. WMC001374. <http://arxiv.org/abs/1012.3803>.
- Perlovsky, L. I. (2010f). Beauty and art. cognitive function, evolution, and mathematical models of the mind. *WebmedCentral Psychol.*, 1(12), p. WMC001322. <http://arxiv.org/abs/1012.3801>.
- Perlovsky L. I. (2011). Language and cognition interaction neural mechanisms. *Comput. Intell. Neurosci.*, Article ID 454587. Open Journal. doi:10.1155/2011/454587. <http://www.hindawi.com/journals/cin/contents/>.
- Perlovsky, L. I. (2012a). Emotions of “higher” cognition. *Behav. Brain Sci.*, 35(3), pp. 157–158.
- Perlovsky, L. I. (2012b). Cognitive function, origin, and evolution of musical emotions. *Musicae Scientiae*, 16(2), pp. 185–199. doi: 10.1177/1029864912448327.
- Perlovsky, L. I. (2012c). Fundamental principles of neural organization of cognition. *Nat. Precedings*: hdl:10101/npre.2012.7098.1. <http://precedings.nature.com/documents/7098/version/1>.
- Perlovsky, L. I. (2013a). A challenge to human evolution—cognitive dissonance. *Front. Psychol.*, 4, p. 179. doi: 10.3389/fpsyg.2013.00179.
- Perlovsky, L. I. (2013b). Language and cognition—joint acquisition, dual hierarchy, and emotional prosody. *Front. Behav. Neurosci.*, 7, p. 123. doi:10.3389/fnbeh.2013.00123. http://www.frontiersin.org/Behavioral_Neuroscience/10.3389/fnbeh.2013.00123/full.
- Perlovsky, L. I. (2013c). Learning in brain and machine—complexity, Gödel, Aristotle. *Front. Neurorobot.* doi: 10.3389/fnbot.2013.00023. <http://www.frontiersin.org/Neurorobotics/10.3389/fnbot.2013.00023/full>.
- Perlovsky, L. I. (2014a). Aesthetic emotions, what are their cognitive functions? *Front. Psychol.*, 5, p. 98. <http://www.frontiersin.org/Journal/10.3389/fpsyg.2014.00098/full>; doi:10.3389/fpsyg.2014.0009.
- Perlovsky, L. I. and Ilin, R. (2010). Neurally and mathematically motivated architecture for language and thought. Brain and language architectures: where we are now? *Open Neuroimaging J.*, Spec. issue, 4, pp. 70–80.

<http://www.bentham.org/open/tonij/openaccess2.htm>.

- Perlovsky, L. I. and Ilin, R. (2012). Mathematical model of grounded symbols: perceptual symbol system. *J. Behav. Brain Sci.*, 2, pp. 195–220. doi:10.4236/jbbs.2012.22024. <http://www.scirp.org/journal/jbbs/>.
- Perlovsky, L. I. and Ilin, R. (2013). CWW, language, and thinking. *New Math. Nat. Comput.*, 9(2), pp. 183–205. doi:10.1142/S1793005713400036. <http://dx.doi.org/10.1142/S1793005713400036>.
- Perlovsky, L. I. and Levine, D. (2012). The Drive for creativity and the escape from creativity: neurocognitive mechanisms. *Cognit. Comput.* doi:10.1007/s12559-012-9154-3. <http://www.springerlink.com/content/517un26h46803055/>. <http://arxiv.org/abs/1103.2373>.
- Perlovsky, L. I. and McManus, M. M. (1991). Maximum likelihood neural networks for sensor fusion and adaptive classification. *Neural Netw.*, 4(1), pp. 89–102.
- Perlovsky, L. I. and Shevchenko, O. (2014). Dynamic Logic Machine Learning for Cybersecurity. In Pino, R. E., Kott, A. and Shevenell, M. J. (eds.), *Cybersecurity Systems for Human Cognition Augmentation*. Zug, Switzerland: Springer.
- Perlovsky, L. I., Bonniot-Cabanac, M.-C. and Cabanac, M. (2010). Curiosity and pleasure. *WebmedCentral Psychol.*, 1(12), p. WMC001275. http://www.webmedcentral.com/article_view/1275. <http://arxiv.org/ftp/arxiv/papers/1010/1010.3009.pdf>.
- Perlovsky, L. I., Deming, R. W. and Ilin, R. (2011). *Emotional Cognitive Neural Algorithms with Engineering Applications. Dynamic Logic: From Vague to Crisp*. Heidelberg, Germany: Springer.
- Perlovsky, L. I., Cabanac, A., Bonniot-Cabanac, M.-C. and Cabanac, M. (2013). Mozart effect, cognitive dissonance, and the pleasure of music. arxiv 1209.4017; *Behavioural Brain Research*, 244, 9–14.
- Petrov, S., Fontanari, F. and Perlovsky, L. I. (2012). Subjective emotions vs. verbalizable emotions in web texts. *Int. J. Psychol. Behav. Sci.*, 2(5), pp. 173–184. <http://arxiv.org/abs/1203.2293>.
- Pinker, S. (1994). *The Language Instinct: How the Mind Creates Language*. New York: William Morrow.
- Price, C. J. (2012). A review and synthesis of the first 20 years of PET and fMRI studies of heard speech, spoken language and reading. *Neuroimage*, 62, pp. 816–847.
- Rauscher, F. H., Shaw, L. and Ky, K. N. (1993). Music and spatial task performance. *Nature*, 365, p. 611.
- Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), pp. 533–536.
- Schiller, F. (1895). In Hinderer, W. and Dahlstrom, D. (eds.), *Essays, German Library, Volume 17*. GB, London: Continuum Pub Group.
- Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61, pp. 85–117, doi: 10.1016/j.neu-net.2014.09.003.
- Setiono, R. (1997). A penalty-function approach for pruning feedforward neural networks. *Neural Comput.*, 9(1), pp. 185–204.
- Simonton, D. K. (2000). Creativity. Cognitive, personal, developmental, and social aspects. *Am. Psychol.*, 55(1), pp. 151–158.
- Singer, R. A., Sea, R. G. and Housewright, R. B. (1974). Derivation and evaluation of improved tracking filters for use in dense multitarget environments. *IEEE Trans. Inf. Theory*, IT-20, pp. 423–432.
- Stroop, J. R. (1935). Studies of interference in serial verbal reactions. *J. Exp. Psych.*, 18, pp. 643–682.
- Tikhonoff, V., Fontanari, J. F., Cangelosi, A. and Perlovsky, L. I. (2006). Language and cognition integration through modeling field theory: category formation for symbol grounding. In *Book Series in Computer Science, Volume 4131*. Heidelberg: Springer.
- Titterington, D. M., Smith, A. F. M. and Makov, U. E. (1985). *Statistical Analysis of Finite Mixture Distributions*. New York: John Wiley & Sons.
- Tversky, A. and Kahneman, D. (1974). Judgment under uncertainty: heuristics and biases. *Science*, 185, pp. 1124–1131.
- Vapnik, V. (1999). *The Nature of Statistical Learning Theory, Second Edition*. New York: Springer-Verlag.

- Vityaev, E. E., Perlovsky, L. I., Kovalerchuk, B. Y. and Speransky, S. O. (2011). Probabilistic dynamic logic of the mind and cognition. *Neuroinformatics*, 5(1), pp. 1–20.
- Vityaev, E. E., Perlovsky, L. I., Kovalerchuk, B. Y. and Speransky, S. O. (2013). Pro invited article. *Biologically Inspired Cognitive Architectures*, 6, pp. 159–168.
- Werbos, P. J. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis. Boston, USA: Harvard University.
- Whorf, B. (1956). *Language, Thought, and Reality: Selected Writings of Benjamin Lee Whorf*. Carroll, J. B. (ed.). Cambridge: MIT Press.
- Wilkins, R. W., Hodges, D. A., Laurienti, P. J., Steen, M. R. and Burdette, J. D. (2012). Network science: a new method for investigating the complexity of musical experiences in the brain. *Leonardo*, 45(3), pp. 282–283.
- Zadeh, L. A. (1965). Fuzzy sets. *Inf. Control*, 8, pp. 338–352.
- Zeki, S., Romaya, J. P., Benincasa, D. M. T. and Atiyah, M. F. (2014). The experience of mathematical beauty and its neural correlates. *Human Neurosci.*, 8, Article 68. www.frontiersin.org.

Chapter 9

Introduction to Cognitive Systems

Péter Érdi and Mihály Bányai

The chapter reviews some historical and recent trends in understanding natural and developing artificial cognitive systems. One of the fundamental concepts of cognitive science, i.e., mental representation, is discussed. The two main directions, symbolic and connectionist (and their combination: hybrid) architectures are analyzed. Two main cognitive functions, memory and language, are specifically reviewed. While the pioneers of cognitive science neglected neural level studies, modern cognitive neuroscience contributes to the understanding of neural codes, neural representations. In addition, cognitive robotics builds autonomous systems to realize intelligent sensory-motor integration.

9.1. Representation and Computation

9.1.1. *Representations*

Cognitive science (CS), the interdisciplinary study of the mind, deals on the one hand with the understanding of the human mind and intelligence and on the other hand with the construction of an artificial mind and artificial cognitive systems. Its birth was strongly motivated by the information processing paradigm, thus CS aims to explain thinking as a computational procedure acting on representational structures. Historically, Kenneth Craik (Craik, 1943) argued that the mind does not operate directly on external reality, but on **internal models**, i.e., on representations. CS predominantly assumes that the mind has **mental representations** and computational manipulations on these representations are used to understand and simulate thinking. “... He emphasizes the three processes of translation, inference, and retranslation: “the translation of external events into some kind of neural patterns by stimulation of the sense organs, the interaction and stimulation of other neural patterns as in ‘association’, and the excitation by these effectors or motor patterns.” Here, Craik’s paradigm of stimulus-association-response allows the response to be affected by association with the person’s current model but does not sufficiently invoke the active control of its stimuli by the organism ...” (Arbib *et al.*, 1997, p. 38). Different versions of representations, such as logic, production rules (Newell and Simon, 1972), semantic networks of concepts (Collins and Quillian, 1969; Quillian, 1968), frames (Minsky, 1974), schemata (Bobrow and Norman, 1975), scripts (Schank and Abelson, 1977), and mental models,¹ analogies, images (Shepard, 1980) have been suggested and analyzed in terms of their representation and computational power, neural plausibility, etc. (Thagard, 2005). It is interesting to see that while behaviorism famously ignored the study of mental events, cognitive science, although it was born by attacking behaviorism in the celebrated paper of Chomsky on Skinner (Chomsky, 1959), was also intentionally ignorant: neural mechanisms were not really in the focus of the emerging interdisciplinary field. Concerning the mechanisms of neural-level and mental level representations, Churchland and Sejnowski (Churchland and Sejnowski, 1990) argued that representations and computations in the brain seem to be very different from the ones offered by the traditional symbol-based cognitive science.

9.1.2. *Symbolic Representation*

9.1.2.1. *The physical symbol system hypothesis*

The physical symbol system hypothesis served as the general theoretical framework for processing information by serial mechanism, and local symbols. It stated that physical symbol system has the necessary and sufficient means for general intelligent action (Newell and Simon, 1976) and serves as the general theoretical framework for processing information by serial mechanism, and local symbols. It is necessary, since anything

capable of intelligent action is a physical symbol system. It is sufficient, since any relevant physical symbol system is capable of intelligent action. The hypothesis is based on four ideas

- Symbols are physical patterns.
- Symbols can be combined to form more complex structures of symbols.
- Systems contain processes for manipulating complex symbol structures.
- The processes for representing complex symbol structures can themselves be symbolically represented within the system.

Thinking and intelligence was considered as problem solving. For well-defined problems, the problem space (i.e., branching tree of achievable situations) was searched by algorithms. Since problem spaces proved to be too large to be searched by brute-force algorithms, selective search algorithms were used by defining **heuristic** search rules.

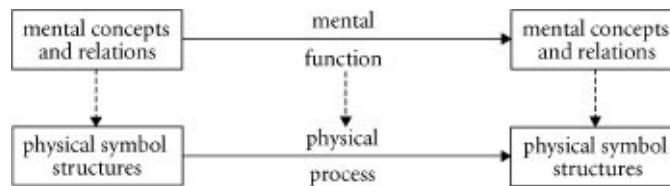


Figure 9.1: The assumed relationship between mental concepts and symbol structures.

9.1.3. Connectionism

While the emergence of connectionism is generally considered as the reaction for the not sufficiently rapid development of the symbolic approach, it had already appeared during the golden age of cybernetics related to the McCulloch–Pitts (MCP) models.

9.1.3.1. MCP model

In 1943, McCulloch, one of the two founding fathers of cybernetics (the other was Norbert Wiener), and the prodigy Walter Pitts published a paper with the title “A Logical Calculus of the Ideas Immanent in Nervous System”, which was probably the first experiment to describe the operation of the brain in terms of interacting neurons (McCulloch and Pitts, 1943), for historical analysis see (Abraham, 2002; Arbib, 2000; Piccinini, 2004).

The MCP model was basically established to capture the logical structure of the nervous system. Therefore, cellular physiological facts known even that time were intentionally neglected.

The MCP networks are composed by multi-input ($x_i, i = 1, \dots, n$) single output (y) threshold elements. The state of one element (neuron) of a network is determined by the following rule: $y = 1$, if the weighted sum of the inputs is larger than a threshold, and $y = 0$, in any other case:

$$y = \begin{cases} 1, & \text{if } \sum_i w_i x_i > \Theta \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

Such a rule describes the operation of all neurons of the network. The state of the network is characterized at a fixed time point by a series of zeros and ones, i.e., by a binary vector, where the dimension of the vector is equal with the number of neurons of the network. The updating rule contains an arbitrary factor: during one time step either the state of *one* single neuron or of the *all* neurons can be modified. The former materializes asynchronous or serial processing, the latter materializes synchronous or parallel processing.

Obviously, the model contains neurobiological simplifications. The state is binary, the time is discrete, the threshold and the wiring are fixed. Chemical and electrical interactions are neglected, glia cells are also not taken into consideration. McCulloch and Pitts showed that a large enough number of synchronously updated neurons connected by appropriate weights could perform many possible computations.

Since all Boolean functions can be calculated by loop-free (or feed-forward) neuronal networks, and all finite automata can be simulated by neuronal networks (loops are permitted, i.e., recurrent networks), von Neumann adapted the MCP model to the logical design of the computers. The problem of the brain–computer analogy/disanalogy was a central issue of early cybernetics, in a sense revived by the neurocomputing boom from the mid-eighties. More precisely, the metaphor has two sides (“computational brain” versus “neural computer”). There are several different roots of the early optimism related to the power of the brain–computer analogy. We will review two of them. First, both elementary computing units and neurons were characterized as digital input–output devices, suggesting an analogy at even the elementary hardware level. Second, the equivalence (more or less) had been demonstrated between the mathematical model of the “control box” of a computer as represented by the state-transition rules for a Turing machine, and of the nervous system as represented by the. Binary vectors of “0”s and “1”s represented the state of the computer and of the brain, and their temporal behavior was described by the updating rule of these vectors. In his posthumously published book *The Computer and the Brain*, John von Neumann (von Neumann, 1958) emphasized the particular character of “neural mathematics”: “... The logics and mathematics in the central nervous system, when viewed as languages, must structurally be essentially different from those languages to which our common experience refers ...”.

The MCP model (i) introduced a formalism whose refinement and generalization led to the notion of finite automata (an important concept in computability theory); (ii) is a technique that inspired the notion of logic design of computers; (iii) was used to build neural network models to connect neural structures and functions by dynamic models; (iv) offered the first modern computational theory of brain and mind.

One possible generalization of the MCP model is that the threshold activation function

is substituted by a g activation function.

9.1.3.2. Hebbian learning rules

Hebb marked a new era by introducing his learning rule and resulted in the sprouting of many new branches of theories and models on the mechanisms and algorithms of learning and related areas.

Two characteristics of the original postulate (Hebb, 1949) played key role in the development of post-Hebbian learning rules. First, in spite of being biologically motivated, it was a verbally described, *phenomenological* rule, without having view on detailed physiological mechanisms. Second, the idea seemed to be extremely convincing, therefore it became a widespread theoretical framework and a generally applied formal tool in the field of neural networks. Based on these two properties, the development of Hebb's idea followed two main directions. First, the postulate inspired an intensive and long lasting search for finding the *molecular* and *cellular* basis of the learning phenomena — which have been assumed to be Hebbian — thus this movement has been absorbed by neurobiology. Second, because of its computational usefulness, many variations evolved from the *biologically inspired* learning rules, and were applied to huge number of very different problems of artificial neural networks, without claiming any relation to biological foundation.

The simplest Hebbian learning rule can be formalized as:

$$\frac{d}{dt} w_{ij}(t) = k a_i(t) a_j(t), \quad k > 0. \quad (2)$$

This rule expresses the conjunction among pre- and post-synaptic elements (using neurobiological terminology) or associative conditioning (in psychological terms), by a simple product of the actual states of pre- and post-synaptic elements, $a_i(t)$ and $a_j(t)$. A characteristic and unfortunate property of the simplest Hebbian rule is that the synaptic strengths are ever increasing.

Long-term potentiation (LTP) was first discovered in the hippocampus and is very prominent there. LTP is an increase in synaptic strength that can be rapidly induced by brief periods of synaptic stimulation and which has been reported to last for hours *in vitro*, and for days and weeks *in vivo*.

The LTP (and later the LTD) after their discovery, have been regarded as the physiological basis of Hebbian learning. Subsequently, the properties of the LTP and LTD became more clear, and the question arises, whether the LTP and LTD could really be considered as the microscopical basis of the phenomenological Hebb type learning. Formally, the question is that how to specify the general functional F to serve as a learning rule with the known properties of LTP and LTD. Recognizing the existence of this gap between biological mechanisms and the long-used Hebbian learning rule, there have been

many attempts to derive the corresponding phenomenological rule based on more or less detailed neurochemical mechanisms.

Spike Timing Dependent Plasticity (STDP), a temporally asymmetric form of Hebbian learning induced by temporal correlations between the spikes of pre- and post-synaptic neurons, has been discovered (Bi and Poo, 1998) and extensively studied (Sjoestrom and Gerstner, 2010). For reviews of post-Hebbian learning algorithms and models of synaptic plasticity, see e.g., (Érdi and Somogyvári, 2002; Shouval, 2007).

9.1.3.3. Learning in artificial neural networks

The MCP model supplemented with Hebb's concept about the continuously changing connectivities led to the pattern recognizing algorithm, famously called as the Perceptron (Rosenblatt, 1962). Actually, by using the modern terminology, the Hebbian rules belong to the class of unsupervised learning rule, while the Perceptron implements supervised learning

The Perceptron is a classifier defined as

$$f(x) = w^T x + b. \quad (3)$$

The classification rule is $\text{sign}(f(x))$. The learning rule is to perform the following updates if the classifier makes an error:

$$w^{t+1} = w^t + yx, \quad (4)$$

$$b^{t+1} = b^t + y. \quad (5)$$

A version of the single layer perceptron uses the **delta** learning rule. It is a gradient descent method, and adapts the weights to minimize the deviation between the target value and the actual value. The delta rule for the modification of the synaptic weights w_{ji} is given as

$$\delta w_{ji} = \alpha(t_j - y_j)g'(h_j)x_i. \quad (6)$$

Here, α is the learning rate, t_j and y_j are the target and actual output values of neuron j , h_j is the weighted sum of the individual input x_i s.

Famously, the Perceptron can solve linearly separable problems (Minsky and Papert, 1969) only, and such kinds of networks are not able to learn, e.g., the XOR function.

Multilayer perceptrons supplemented with another learning algorithm (Werbos, 1974), i.e., with the backpropagation (BP) algorithm overcame the limitations of the single-layer Perceptron. BP is a generalization of the delta rule to multilayered feedforward networks. It is based on using the chain rule to iteratively compute gradients for each layer, and it

proved to be a useful algorithm.

BP algorithm has two parts. First, there is a forward activity propagation followed by the backpropagation of the output to calculate the delta quantities for each neuron. Second, the gradient is calculated and the update of the weight is determined.

Connectionist (what else?) networks consisting of simple nodes and links are very useful for understanding psychological processes that involve parallel constraint satisfaction. (Neo!) connectionism has been revitalized and became a popular alternative of the symbolistic approach from the mid-eighties, when the two volumes of *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* were published (Rumelhart and McClelland, 1986). An early successful application was an (artificial) neural network to predict the past tense of English verbs.

9.1.3.4. *Newer developments*

After the initial success, multilayer perceptrons with BP started to show their limitations with more complex classifications tasks. Multiple branches of machine learning techniques originated from generalization attempts of neural architectures.

One such advancement was the introduction of support vector machines (SVM) (Cortes and Vapnik, 1995) by Vapnik and Cortes. The mathematical structure is formally equivalent to a perceptron with one hidden layer of which the size may be potentially infinite. The goal of the optimization algorithm used to tune the network's parameters is to find a classification boundary that maximizes distance from data points in separated classes. Decision boundaries may be nonlinear if data is transformed into the feature space using a kernel function of appropriate choice. SVMs produce reproducible, optimal (in a certain sense) classifications based on a sound theory. They may be successfully applied in problems where data is not overly high-dimensional with relatively simple underlying structure, but might be very noisy.

Another direction of development of neural networks is deep learning. The basic idea is to stack multiple processing layers of neurons on top of each other forming a hierarchical model structure. Some of the most successful applications use the stochastic neuron instead of the deterministic one, where the probability of assuming one of the binary states is determined by a sigmoid function of the input.

$$\Pr(s_i = 1) = \frac{1}{1 + e^{-\sum_{j=1}^N w_{ij}x_i + b_i}}. \quad (7)$$

Stochastic networks may represent probability distributions over latent variables by the samples generated as the network is updated according to Equation (7). Such neurons may be connected in an undirected (that is, symmetric) manner, forming Boltzmann machines (Salakhutdinov and Hinton, 2009), which can be trained in an unsupervised

fashion on unlabeled datasets by the contrastive divergence algorithm (Hinton, 2002). Directed versions of similar networks may also be trained unsupervisedly by the wake-sleep algorithm (Hinton *et al.*, 1995).

To solve supervised classification problems, the first deep architectures were convolutional networks that solved the translation invariance of learned features by binding together of weights in processing layers. Boltzmann machines may be transformed into classification engines by refining the unsupervisedly pre-trained weights by back propagation of labeled data. Deep networks may be successfully applied to very high-dimensional data with complicated structure if the high signal-to-noise ratio is sufficiently high.

9.1.4. Engineering Perspectives

In terms of philosophical approaches, debates became less sharp, since the physical symbol system hypothesis which stated that [a] physical symbol system has the necessary and sufficient means for general intelligent action. In the technical application symbol, manipulation was the only game in town for many years.

9.1.4.1. General problem solver

General Problem Solver (GPS) was a computer program created in 1959 by Herbert A. Simon, J.C. Shaw, and Allen Newell and was intended to work as a universal problem solver machine. Formal symbolic problems were supposed to be solved by GPS. Intelligent behavior, as automatic theorem proving, and chess playing were paradigmatic examples of the ambitious goals. GPS, however, solved simple problems such as the Towers of Hanoi, that could be sufficiently formalized, it could not solve any real-world problems due to the combinatorial explosion. Decomposition of the task into subtasks and goals into subgoals somewhat helped to increase the efficiency of the algorithms.

9.1.4.2. Expert systems

Expert systems (also called as **knowledge-based** systems) were one of the most widely used applications of classical artificial intelligence. Their success was due to restricted use for specific fields of applications. The general goal has been to convert human knowledge to formal electronic consulting service. Generally it has two parts, the knowledge base and the inference machine. The central core of the inference machines is the rule-base, i.e., set of rules of inference that are used in reasoning. Generally, these systems use IF-THEN rules to represent knowledge. Typically, systems had from a few hundred to a few thousand rules.

The whole process resembles to medical diagnosis, and actually the first applications were towards medicine. For an introduction to expert systems, see e.g., (Jackson, 1998).

9.1.4.3. Knowledge representation and reasoning

Knowledge representation (KR) is a field of artificial intelligence aiming to represent certain aspects of the world to solve complex problems by using formal methods, such as automatic reasoning.

As it was suggested (Davis *et al.*, 1993).

- “...
- A KR is most fundamentally a surrogate, a substitute for the thing itself, used to enable an entity to determine consequences by thinking rather than acting, i.e., by reasoning about the world rather than taking action in it.
 - It is a set of ontological commitments, i.e., an answer to the question: In what terms should I think about the world?
 - It is a fragmentary theory of intelligent reasoning, expressed in terms of three components: (i) the representation’s fundamental conception of intelligent reasoning; (ii) the set of inferences the representation sanctions; and (iii) the set of inferences it recommends.
 - It is a medium for pragmatically efficient computation, i.e., the computational environment in which thinking is accomplished. One contribution to this pragmatic efficiency is supplied by the guidance a representation provides for organizing information so as to facilitate making the recommended inferences.
 - It is a medium of human expression, i.e., a language in which we say things about the world.

...”

In recent years, KR and reasoning has also derived challenges from new and emerging fields including the semantic web, computational biology, and the development of software agents and of ontology-based data management.

9.1.5. Philosophical Perspectives

9.1.5.1. Methodological solipsism

Jerry Fodor suggested methodological solipsism (Fodor, 1980) as a research strategy in cognitive science. He adopts an extreme **internalist** approach: the content of beliefs is determined by what is in the agent’s head, and nothing to do with what is in the world. Mental representations are internally structured much like sentences in a natural language, in that they have both syntactic structure and a compositional semantics.

There are two lines of opinions, while classical cognitivism is based on the representational hypothesis supplemented by the internal world assumption, other

approaches have other categories in their focus. Two of them are briefly mentioned here: **intentionality** and **embodied cognition**.

9.1.5.2. Intentionality

Searle (Searle, 1983, 1992) rejected the assumption undisputed from Craik to Simon that the representational mind/brain operates on formal internal models detached from the world and argued instead that its main feature is intentionality, a term which has been variously viewed as synonymous with connectedness, aboutness, meaningfulness, semantics or straightforwardly consciousness. Searle argued that the representational and computational structures that have typically been theorized in cognitive science lack any acceptable ontology. He argued that they are not being observable or understandable, so these structures just cannot exist.

9.1.5.3. Situated or embodied cognition

A seemingly different attempt to overcome the difficulties of methodological solipsism is to work with agents so simple as to not need a knowledge base at all, and basically don't need representations. The central hypothesis of embodied cognitive science is that cognition emerges from the interaction of brain, the whole body, and of its environment. What does it mean to understand a phenomenon? A pragmatic answer is to synthesize the behavior from elements. Many scientists believe if they are able to build a mathematical model based on the knowledge of the mechanism to reproduce a phenomenon and predict some other phenomena by using the same model framework, they understand what is happening in their system. Alternatively, instead of building a mathematical model one may wish to construct a robot. Rodney Brooks at MIT is an emblematic figure with the goal of building humanoid robots (Brooks, 2002). Embodied cognitive science now seems to be an interface between neuroscience and robotics: the features of embodied cognitive systems should be built both into neural models, and robots, and the goal is to integrate sensory, cognitive and motor processes. (Or even more, traditionally, emotions were neglected, as factors which reduce the cognitive performance. It is far from being true.)

9.2. Architectures: Symbolic, Connectionist, Hybrid

9.2.1. Cognitive Architectures: What? Why? How?

9.2.1.1. Unified theory of cognition

Alan Newell (Newell, 1990) spoke about the unified theory of cognition (UTC). Accordingly, there is single set of mechanisms that account for all of cognition (using the term broadly to include perception and motor control). UTC should be a theory to explain (i) the adaptive response of an intelligent system to environmental changes; (ii) the mechanisms of goal seeking and goal-driven behavior²; (iii) how to use symbols and (iv) how to learn from experience. Newell's general approach inspired his students and others to establish large software systems, cognitive architectures, to implement cognitions.

“Cognitive architecture is the overall, essential structure and process of a domain-generic computational cognitive model, used for a broad, multiple-level, multiple-domain analysis of cognition and behavior ...” (Sun, 2004). They help to achieve two different big goals: (i) to have a computational framework to model and simulate real cognitive phenomena: (ii) to offer methods to solve real-world problems.

Two key design properties that underlie the development of any cognitive architecture are memory and learning (Duch *et al.*, 2007). For a simplified taxonomy of cognitive architectures, see [Section 9.2.1.1](#).

Symbolic architectures focus on information processing using high-level symbols or declarative knowledge, as in the classical AI approach. Emergent (connectionist) architectures use low-level activation signals flowing through a network consisting of relatively simple processing units, a bottom-up process relaying on the emergent self-organizing and associative properties. Hybrid architectures result from combining the symbolic and emergent (connectionist) paradigms.

The essential features of cognitive architectures have been summarized (Sun, 2004). It should show (i) ecological realism, (ii) bio-evolutionary realism, (iii) cognitive realism and (iv) eclecticism of methodologies and techniques. More specifically, it cannot be neglected that (i) cognitive systems are situated in sensory-motor system, (ii) the understanding in human cognitive systems can be seen from an evolutionary perspective, (iii) artificial cognitive systems should capture some significant features of human cognition, (iv) at least for the time being multiple perspectives and approaches should be integrated.

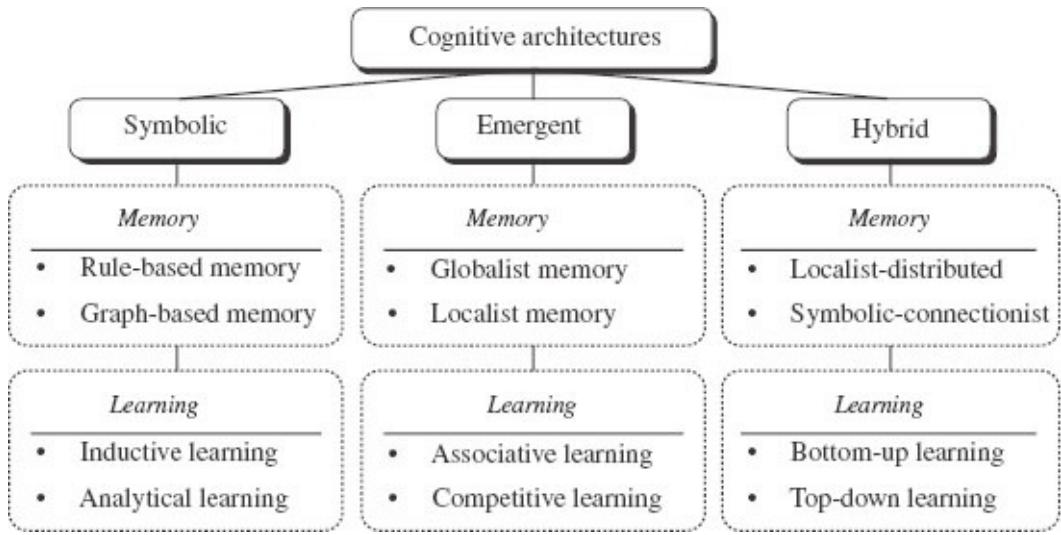


Figure 9.2: Simplified taxonomy of cognitive architectures. From Duch *et al.* (2007).

9.2.2. *The State, Operator and Result (SOAR) Cognitive Architecture*

SOAR is a classic example of expert rule-based cognitive architecture designed to model general intelligence (Laird, 2012; Milnes *et al.*, 1992).

SOAR is a general cognitive architecture that integrates knowledge-intensive reasoning, reactive execution, hierarchical reasoning, planning, and learning from experience, with the goal of creating a general computational system that has the same cognitive abilities as humans. In contrast, most AI systems are designed to solve only one type of problem, such as playing chess, searching the Internet, or scheduling aircraft departures. SOAR is both a software system for agent development and a theory of what computational structures are necessary to support human-level agents.

Based on theoretical framework of knowledge-based systems seen as an approximation to physical symbol systems, SOAR stores its knowledge in the form of production rules, arranged in terms of operators that act in the problem space, that is the set of states that represent the task at hand. The primary learning mechanism in SOAR is termed chunking, a type of analytical technique for formulating rules and macro-operations from problem solving traces. In recent years, many extensions of the SOAR architecture have been proposed: reinforcement learning to adjust the preference values for operators, episodic learning to retain history of system evolution, semantic learning to describe more abstract, declarative knowledge, visual imagery, emotions, moods and feelings used to speed up reinforcement learning and direct reasoning. SOAR architecture has demonstrated a variety of high-level cognitive functions, processing large and complex rule sets in planning, problem solving and natural language comprehension.

9.2.3. *Adaptive Control of Thought-Rational (ACT-R)*

We follow here the analysis of (Duch *et al.*, 2007). ACT-R is a cognitive architecture: a theory about how human cognition works. It is both a hybrid cognitive architecture and

theoretical framework for understanding and emulating human cognition (Anderson, 2007; Anderson and Bower, 1973). Its intention is to construct a software system that can perform the full range of human cognitive functions. The algorithm is realistic at the cognitive level, and weakly realistic in terms of neural mechanisms. The central components of ACT-R comprise a set of modules of perceptual-motor schemas, memory system, a buffer and a pattern matcher. The perceptual-motor modules basically serve as an interface between the system and the external world. There are two types of memory modules in ACT-R: declarative memory (DM) and procedural memory (PM). Both are realized by symbolic-connectionist structures, where the symbolic level consists of productions (for PM) or chunks (for DM), and the sub-symbolic level of a massively parallel connectionist structure. Each symbolic construct (i.e., production or chunk) has a set of sub-symbolic parameters that reflect its past usage and control its operations, thus enabling an analytic characterization of connectionist computations using numeric parameters (associative activation) that measure the general usefulness of a chunk or production in the past and current context. The pattern matcher is used to find an appropriate production.

ACT-R implements a top-down learning approach to adapt to the structure of the environment. In particular, symbolic constructs (i.e., chunks or productions) are first created to describe the results of a complex operation, so that the solution may be available without recomputing the next time a similar task occurs. When a goal, declarative memory activation or perceptual information appears it becomes a chunk in the memory buffer, and the production system guided by subsymbolic processes finds a single rule that responds to the current pattern. Sub-symbolic parameters are then tuned using Bayesian formulae to make the existing symbolic constructs that are useful more prominent. In this way, chunks that are often used become more active and can thus be retrieved faster and more reliably. Similarly, productions that more likely led to a solution at a lower cost will have higher expected utility, and thus be more likely chosen during conflict resolution (i.e., selecting one production among many that qualify to fire).

9.3. Cognitive Functions

9.3.1. General Remarks

Cognitive functions are related to mental processes, such as attention, learning, memory, language comprehension and production, reasoning, problem solving, planning, decision-making, etc. The mental processes can be realized by conscious or unconscious mechanisms. As an illustration, two topics, **memory** and **language** are briefly reviewed here.

9.3.2. Multiple Memory Systems

As also detailed in [Section 9.4.4.2](#), knowledge stored in the human brain can be classified into sparable memory systems. An important division can be made in terms of duration of recallability. Short-term memories serve as a temporary storage, that helps the execution of everyday tasks, and pre-store certain information that can later be solidified into long-term memories.

Long-terms memory can be divided into three subsystems according to function. The first is procedural memory, encoding how to swim or draw a flower. The second is episodic memory, that can store past events, similar to the scenes of a movie. And the third, semantic memory, is everything that we know about the world in a more or less context-invariant manner.

Different memory systems clearly interact, as sensory information needs to be interpreted according to semantic knowledge in order to be efficiently stored in short-term or episodic memory, which is in turn built into the semantic web of knowledge. However, there is evidence that different systems may operate separately from each other, as illustrated by the case of H.M., a patient with severe epilepsy who had to have hippocampal lobotomy. He retained his semantic knowledge about the world and his procedural skills, together with the ability to acquire new procedural knowledge, but he completely lost the ability to form new episodic memory patterns (he had anterograde amnesia).

A model for the operation of multiple memory systems on a cognitive level was proposed by Tulving (1985).

9.3.3. Language Acquisition, Evolution and Processing

9.3.3.1. What is language?

Language is a system of symbols used to communicate ideas among two or more individuals. Normatively, it must be learnable by children, spoken and understood by adults, and capable of expressing ideas that people normally communicate in a social and cultural context.

9.3.3.2. Cognitive approach to linguistics

As Paul Thagard reviews, the cognitive approach to linguistics raises a set of fundamental questions:

- How does the mind turn sounds into words (phonology)?
- How does the mind turn words into sentences (syntax)?
- How does the mind understand words and sentences (semantics)?
- How does the mind understand discourse (semantics, pragmatics)?
- How does the mind generate discourse?
- How does the mind translate between languages?
- How does the mind acquire the capacities just described?
- To what extent is knowledge of language innate?

Hypotheses about how the mind uses language should be tested:

- Symbolic
 - Linguistic knowledge consists largely of rules that govern phonological and syntactic processing.
 - The computational procedures involved in understanding and generating language are largely rule-based.
 - Language learning is learning of rules.
 - Many of these rules are innate.
 - The leading proponent of this general view has been Noam Chomsky.
 - Rule-based models of language comprehension and generation have been developed e.g., in the SOAR system and within other frameworks.
- Connectionist
 - Linguistic knowledge consists largely of statistical constraints that are less general than rules and are encoded in neural networks.
 - The computational procedures involved in understanding and generating language are largely parallel constraint satisfaction.

9.3.3.3. Language acquisition

As it is well-known, behaviorists psychology considered language as a learned habit, and famously one of the starting points of the cognitive science was Chomsky's attack on Skinner's concepts (Chomsky, 1959). Chomsky's theory of generative grammar,

approaches to children's acquisition of syntax (Chomsky, 1965) led to the suggestion of having a **universal grammar**. In a somewhat different context, it is identified with **language faculty** based on the *modularity of the mind* (Fodor, 1983) or the **language instinct** (Pinker, 2007). Language acquisition seems to be now a cognitive process that emerges from the interaction of biological and environmental components.

9.3.3.4. Language evolution

Is language mediated by a sophisticated and highly specialized “language organ” that is unique to humans and emerged completely out of the blue as suggested by Chomsky? Or was there a more primitive gestural communication system already in place that provided a scaffolding for the emergence of vocal language?

Steven Pinker and Paul Bloom (1990) argued for an adaptationist approach to language origins. Rizzolatti's (2008) discovery of the mirror neurons offered a new perspective of language evolution. A mirror neuron is a neuron that fires both when an animal acts and when the animal observes the same action performed by another. The mirror neuron hypothesis leads to a neural theory of language evolution reflected in [Figure 9.3](#).

9.3.4. Language processing

Early AI has strong interest in (natural) language processing (NLP). One of the pioneers of AI, Terry Winograd created a software (SHRDLU) to understand a language about a “toy world” (Winograd, 1972). SHRDLU was instructed to move various objects around in the “blocks world” containing various basic objects: blocks, cones, balls, etc. The system also had some memory to store the names of the object. Its success generated some optimism, but the application of the adopted strategy for real world problems remained restricted.

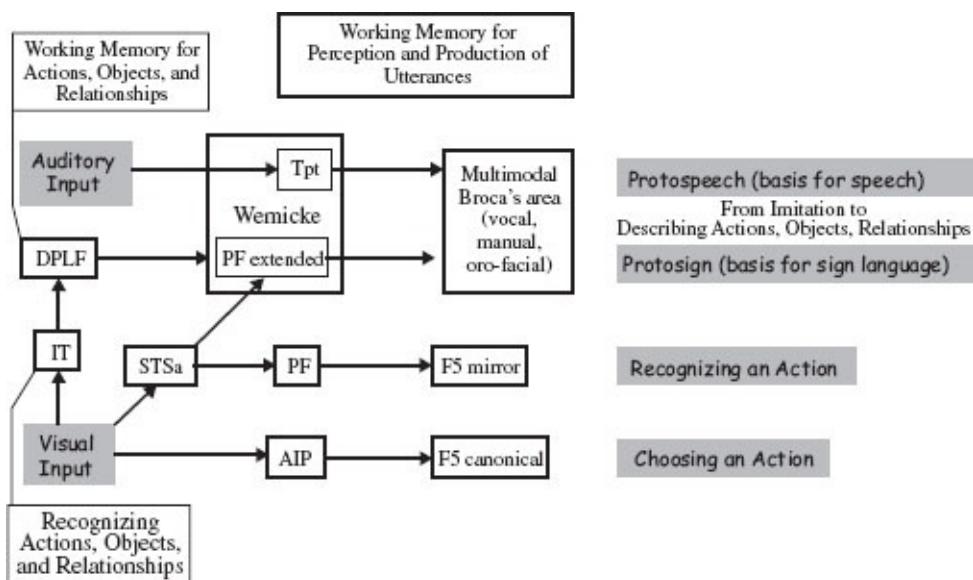


Figure 9.3: Model of the influence of protosign upon the mirror system and its impact on the evolution of language (Arbib, 2005).

The new NPL systems are mostly based on machine learning techniques, often by using statistical inference. Initially the big goal was to make “machine translation”. Nowadays, there are many tasks of NPL, some of them listed here: speech recognition (including speech segmentation), information extraction/retrieval are related more to syntactic analysis, while sentiment analysis and automatic summarization needs semantic/pragmatic analysis (see e.g., Jurafsky and Marti, 2008).

9.4. Neural Aspects

9.4.1. Biological Overview

9.4.1.1. Hierarchical organization

Cognitive functions are realized by the nervous system of animals and humans. The central computing element of these systems is the cortex, which connects to the outside world and the body through sensors and actuators. The cortex can be regarded as a hierarchy of networks operating at different scales. The basic building block of the cortex is the neuron (Ramón y Cajal, 1909), a spatially extended cell that connects to other such cells by synapses. These are special regions of the cell membrane, where electrical changes can trigger the release of certain molecules in the intercellular space. These molecules, the neurotransmitters, may bind to the receptor proteins of the other cell of the synapse, changing its membrane potential (the difference between the electric potential of intracellular and extracellular space, maintained by chemical concentration gradients). Additionally, the membrane potential dynamics of the neurons may produce action potentials (also called spikes or firing), sudden changes that propagate along the elongated axon of the cell to the synapses, triggering transmitter release.

9.4.1.2. Chemical systems of cellular operation

Protein interaction networks support the life cycle of every living cell, but neurons are equipped with additional machinery enabling them to transmit electrical impulses to each other, possibly over large distances as described above. The transmitters in each synapse may be excitatory or inhibitory, depending on the direction of their effect on the postsynaptic membrane potential. A single cell may only form either excitatory or inhibitory synapses. The principal excitatory cells in the cortex are called pyramidal cells. Synaptic signalling and connected cellular machineries are operated by the interaction of hundreds of proteins and other molecules and ions (Ma'ayan *et al.*, 2005).

9.4.1.3. The network of cells

According to our current knowledge, the human brain consists of 10^{11} neurons, each receiving and giving about 10^4 synapses. Most of these cells are formed by the time the child is born, but neurogenesis occurs during adulthood as well, to a lesser extent (Zhao *et al.*, 2008). Synapses can be formed, eliminated and changed in transmission efficacy by sensory experiences.

The cortical network of neurons is not uniformly connected, but organized according to architectural principles. On the surface of the cortical tissue we find the grey matter, which consists of cell bodies, and below the white matter, which is mostly axons and axonal bundles connecting distant neurons. Grey matter is organized into horizontal

layers, which differ in the relative number of the cells of different types in them. Most of the cortex has six layers, some areas have five or less.

The horizontal organization of cortical tissue is much less well understood than the vertical in terms of layers. A candidate for such a structure is the concept of cortical column as proposed by Mountcastle (1997). Micro-columns consist of about 80 pyramidal neurons developed from the same stem cell, and macro-columns consist about 8,000 pyramidal cells. Columnar organization is expressed to a variable degree in different areas of the cortex.

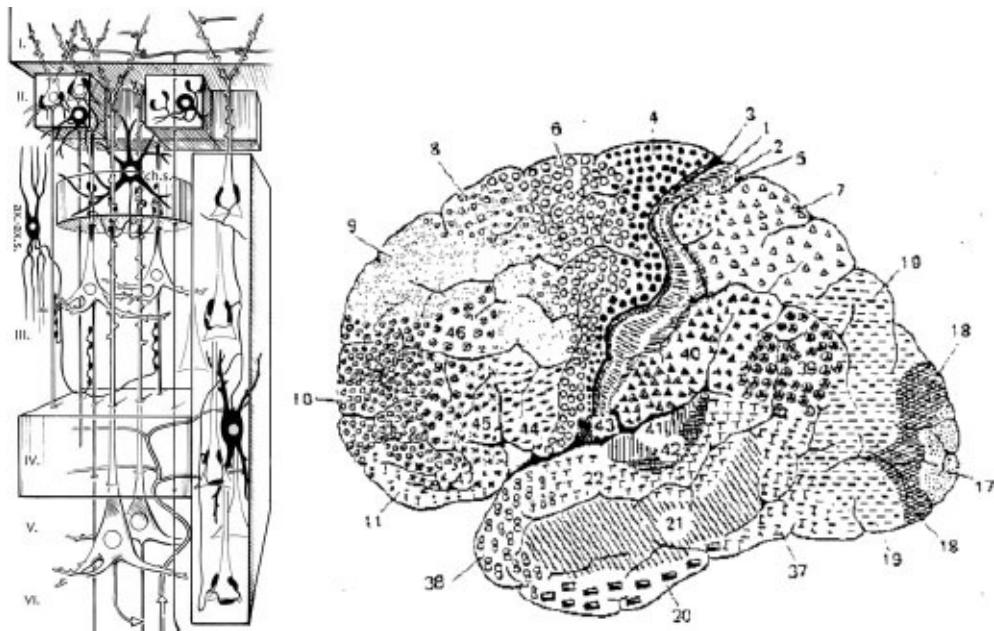


Figure 9.4: Anatomy of neural networks. Left: cellular connectivity in the cortex by Szentágothai. Right: cortical areas by Brodmann.

A central question about cortical networks is whether a repeated local connectivity pattern between cells exists over all or most of the cortex. Such a hypothesized module is called the *canonical microcircuit*. Connectivity between layers exhibits a certain level of constancy as noted by Szentágothai (1983), thus it may serve as a candidate for the canonical microcircuit (depicted in [Figure 9.4](#)). Mapping of such connectivity patterns is possible by axonal tracing (Lanciego and Wouterlood, 2011).

9.4.1.4. Areas by anatomy and function

Due to the huge number of neurons in the brain, it is useful to define macroscopic units in their network. As the cortex is indeed clustered in terms of connectivity and consists of highly specialized regions in terms of involvement in different behavioral tasks, this partition can be constructed according to anatomical and functional measures. The classical specification of 52 cortical areas was given by Brodmann (depicted in [Figure 9.4](#)), and was refined by generations of anatomists and physiologists since then (Strotzer, 2009).

We can often associate certain functions to areas by measuring their population

activity during the execution of different tasks. Sensory areas deal with the primary processing of the external input of the nervous system. Further processing and combination of information happens in the associative areas. The hippocampus is a widely studied area that is involved in the formation and recall of memory patterns. Higher level cognitive functions such as decision-making, attention, control over other areas and planning are associated with the prefrontal areas. The output of the nervous system is produced in the motor areas. Highly specialized areas exist for some specific tasks like face recognition or processing and production of speech.

Cortical areas are connected by axons and axonal bundles in both local and nonlocal manners. Mapping of this network is possible by Diffusion Tensor Imaging (Iturria-Medina *et al.*, 2008) together with axonal tracing.

9.4.2. *The Neural Code*

9.4.2.1. *Membrane potential, spikes and cortical state*

A central question of neuroscience is which physiological phenomena of the neurons are responsible for transmitting information and exactly how. Current consensus is that the membrane potential and the spikes are the most important. There are other signaling pathways in the nervous system that may have modulatory effects on information transfer between cells (Vizi *et al.*, 2004).

If we measure the electric field generated by a large population of neurons (the local field potential, LFP), we can observe periodicity with different frequencies. Based on such spectra and other characteristics of the LFP, one can define cortical states in various ways (Harris and Thiele, 2011). REM and non-REM sleep and wakefulness may be defined as macroscopical states of the cortex, one may observe regular transitions to more synchronous and asynchronous global spiking behavior of neurons, there may be local up- and downstates in sensory areas on the ~100 ms scale, and the phase of different spectral components of the LFP may define a point in a state space of a dynamical system too. The importance and function of all of these definitions of cortical states is actively debated and far from being settled.

A detailed mathematical model of membrane potential dynamics of a neuron was given by Hodgkin and Huxley (Hodgkin and Huxley, 1952). Different hypotheses about the neural code can be expressed as models abstracting away certain aspects of this physiologically realistic description.

9.4.2.2. *Temporal and phase codes*

Temporal code hypotheses assume that the exact timing of the spikes of a neuron are relevant regarding information transfer, discarding the shape of the spike. The mathematical model that formalizes the temporal coding hypothesis is the leaky-integrate-

and-fire neuron (LIF), looking back to a long history in science (Abbott, 2005). The dynamics of this model is given by the following equation:

$$\tau \frac{dV}{dt} = -V(t) + RI(t), \quad (8)$$

where V is the membrane potential of the cell, I is any input current from synapses and R is synaptic resistance. We have to define a firing threshold θ , an equilibrium potential V_0 , a refractory period τ_r and a refractory potential V_r to make the model complete. In a simulation, we would compute the trajectory of the system according to Equation (8), and if V reaches θ , we record a spike, and set V to V_r for time τ_r , and then to V_0 . By defining a synaptic model that describes $I(t)$ depending on presynaptic spike times, we can build networks of LIF neurons to simulate population behavior.

Phase coding hypotheses assume that spike timing is relevant in relation to the phase of some periodic component of the averaged population activity. These oscillations are hypothesized to implement different cognitive functions. Waking state of the cortex is characterized by beta (15–30 Hz) and gamma (30–80 Hz) oscillations (Bressler, 1990). Whether they are the way neural systems implement information processing (Buzsáki, 2004) or an epiphenomenon of computational algorithms and stimulus properties (Bastos *et al.*, 2012) is still to be settled.

9.4.2.3. Rate codes

Rate coding hypotheses assume that the exact timing of the spikes is not important, only the frequency with which they occur. The model formalizing such hypotheses is the rate neuron, defined at a discrete time step as follows:

$$r = \text{sig}\left(\sum_{i=1}^D w_i u_i - \theta\right) \quad (9)$$

with r being the firing rate of a neuron with D input synapses, w_i the efficacy of synapse i , u_i the firing rate of the presynaptic neuron of the same synapse, θ the firing threshold parameter, and $\text{sig}()$ a sigmoid nonlinear function. These model neurons can be used to construct networks without additional elements. Such a network corresponds to the artificial neural network (ANN) paradigm used in machine learning.

Double codes combine the rate and temporal coding aspects, for an example see Lengyel *et al.* (2003).

9.4.2.4. Neural computation

Computation denotes the mathematical transformation that neurons implement on their inputs to create their outputs. In the tradition of cybernetics, neurons and the brain were

regarded as logical devices, carrying out the computation of logical functions. This idea gave rise to the ANN approach to artificial intelligence problems, and a simplified version of the rate model, the McCulloch–Pitts neuron (McCulloch and Pitts, 1943). In this model, instead of the sigmoid nonlinearity, we use a step function, thus the output of the neuron becomes binary, making the networks of such units equivalent to a logical function.

The rate and logical neurons implement the mathematical operation of linear separation of inputs (networks of them combine several decision boundaries solving nonlinear classification too). This operation can be regarded as one of the computational primitives found by physiologists and proposed by computationalists, which also includes linear filtering, exponentiation, divisive normalization (Carandini and Heeger, 2012) and coordinate basis transformation enabling classification (DiCarlo, 2012).

Similar to the quest for the canonical microcircuit, a central question in neuroscience is whether there is a basic computational unit in terms of microcircuitry. Attempts have been made to find an universal computational role for the structural microcircuit (Bastos *et al.*, 2012) and the cortical column (George and Hawkins, 2009).

9.4.3. Neural Representations

Representations associate with neural code the specific quantities it should carry information about, let that be some descriptor of the environment, expectation of the outcome an action, knowledge about a concept or a property of another neural population's activity. These quantities can be described with numerical variables, that serve as building blocks of representations.

If we add formal rules describing relationships between variables, we get a structure equivalent to a formal language, that the natural systems use to store, process and transmit information. This is the internal model that the brain builds from sensory experiences, and what we call neural representation. There may be multiple mathematical frameworks in which one can formulate such models, and a principal goal of neuroscience is to discover which one or more of these is actually related to neural activity. Such a discovery heavily relies on the ability to predict not only behavioral, but physiological quantities, which is no easy task to do at the current state of our knowledge, as even the measurement of these values can be rather challenging.

9.4.3.1. Perception as inference

The idea that sensory perception can be regarded as inference about unobserved quantities describing the environment based on observed sensory activations has deep roots in the history of science (see Helmholtz (Von Helmholtz, 1866) and Alhacen (Smith, 2001)). Such inference requires the building of models of the environment and the world that store previous experiences in a parsimonious manner, allowing the animal or human to calculate

estimates of directly unobserved quantities.

9.4.3.2. *Dynamic models*

Dynamical systems may represent knowledge in the structure of their phase space (Strogatz, 2006) and autonomous dynamics. Through learning (which is to adjust the parameters of the system, as described in [Section 9.4.4.3](#)), several attractors of different nature can be formed in this state space, which may correspond to values of latent quantities describing the environment.

In dynamical systems, the trajectory itself can be regarded as the process of inference. Its starting point (or boundary condition, depending of the modeling framework) is the input to the system, and the attractor where the trajectory ends, be it a point or loop, is the inferred quantity.

Artificial neural networks are a special class of dynamical systems with a close structural correspondence to biological neural systems. Feed-forward networks are universal function approximators, and as such, can be regarded as black box models of knowledge, where a set of adjustable basis functions (hidden units of the network) are used to store a mapping from inputs to outputs without interpreting the activations of the bases.

Chaotic behavior expressed by some dynamical systems can serve as a candidate for modeling spontaneous associations. Chaotic itinerary is one such theory, where trajectories jump from one attractor-like region to another (Tsuda, 1991).

9.4.3.3. *Probabilistic models*

There is ample evidence that the brain calculates with the uncertainty of the represented quantities. The source of this uncertainty may be sensory noise or unobserved properties of the environment. Probability theory offers a consistent way to model these phenomena. From the different interpretations of probabilities, cognitive models use the one that regards the framework as an extension of logic (Jaynes, 2003), and probabilities as measures of available information about quantities. Probabilistic cognitive modeling has been interlocked with the normative modeling approach, advocated by Barlow (Barlow, 1990) and Wiener (Rosenblueth *et al.*, 1943).

Probabilistic models define a joint probability distribution over variables defining the environment of the observer, including quantities directly observable by its sensors, and latent variables, such as the identity of objects in a scene. These models describe the process of generating the observables from the latents, so one can create synthetic stimuli with statistics similar to measured stimuli. The inversion of such a model allows the observer to infer the distribution of latent values given some observations, and to learn the regularities of the environment as parameters.

Physical causality is not represented in probabilistic generative models *per se*, but one may argue that human reasoning is inherently causal instead of probabilistic, and augment the framework of probabilities with concepts directly describing causal relationships, as described by Pearl (2000).

Probabilistic models may be defined to incorporate the temporal evolution of variables or distributions, thus describing dynamic phenomena. Dynamic models with latent variables can describe cognitive processes, such as the inference of invariant features in sensory data by slow feature analysis (Wiskott and Sejnowski, 2002).

9.4.3.4. *How do neurons correspond to variables?*

One of the central questions of neuroscience is how do neurons represent variables of internal models. Many different proposals exist, and the question is far from being settled. If we assume that a specific neuron is responsible for a given variable of the internal model, we have to choose a physiological quantity that will represent the value of that variable at any given point of time. One option is to use the membrane potential of the cell, but one can define a time bin and a spike count, or an instantaneous firing rate without binning. Whether a single variable is represented by only one neuron or more is also unknown. A certain amount of redundancy exists in neural circuitry as representations are robust to minor injuries, but this can be realized by all different variables with heavy dependence too.

The visual system is one of the better understood regions of the cortex in terms of representation. Neurons in the primary visual cortex have receptive fields (that is, the set of inputs that they respond to) resembling wavelet functions, sensitive to edges with a preferred orientation and spatial frequency (Hubel and Wiesel, 1968). Neurons with similar receptive fields are closer to each other, forming a columnar structure, thus connecting modular anatomical architecture to function. In higher-order visual areas one can find neurons responsive to specific concepts, such as people (Quiroga, 2005). Intermediary levels of representation are less well understood, there are competing proposals about the quantities that are represented by neurons of these areas (Ziemba *et al.*, 2013).

If we assume that the probabilistic brain hypothesis is correct, we need the neurons to represent probability distributions of model variables instead of single values. This can also be realized in multiple ways. One is probabilistic population coding (PPC), which aims to represent distributions in a parametric manner (Ma *et al.*, 2006). Another concept is the sampling hypothesis, where the abovementioned neuronal properties encode a sample from the probability distribution of the corresponding variable (Fiser *et al.*, 2010). The sampling hypothesis also creates a connection to a class of practical machine learning algorithms (Neal, 1992).

9.4.4. Learning Internal Models

9.4.4.1. Plasticity in the brain

Storage of information in the brain is realized by the changing strength of coupling between neurons either through the variation of synaptic strengths or the number of synapses. Multiple biological mechanisms exist that implement changes in synaptic connectivity depending on neural activity.

Short-term plasticity (STP) lasts on the timescale of hundreds of milliseconds or seconds. The biological mechanism behind the facilitatory type of STP is the increased probability of transmitter release in connection with spike-dependent presynaptic calcium dynamics. Conversely, depressive effect is exerted by the depletion of transmitters due to frequent spiking. Whether a synapse shows facilitatory or depressive behavior depends on the specific cell type to which it belongs.

Long-term plasticity has effects potentially for decades. It is implemented by a complicated interaction between synaptic proteins and gene regulatory systems.

9.4.4.2. Memory systems of the brain

Similar to plasticity mechanisms, memory systems can be categorized into short and long term storages. Long term memory has multiple distinct subsystems with different goals, separability of which is proven by lesion studies. Motor memory stores sequences of movements that may be executed in certain situations (such as riding a bicycle).

Episodic memory stores autobiographical scenes or short sequences of events that we observed once. Conversely, semantic memory stores the context-independent knowledge about the world including concepts, connections, mechanisms or rules. It is the internal model that describes our best ideas about how the world works. There is a performance advantage of having an episodic memory instead of just the semantic model of the environment as reliable estimates of model parameters are costly in terms of data (Lengyel and Dayan, 2007).

Episodic memories contain snapshots of collections of context-dependent information. To construct a generally valid model of the environment, one has to extract invariant concepts from these snapshots, and organize them into a structure that balances good predictions and compact representation. Such extraction may be described as the construction of concept-trees from scenes (Battaglia and Pennartz, 2011). The *consolidation* of episodic memories into semantic knowledge is hypothesized to be one of the functions of sleeping. The interplay between episodic hypothesized and semantic memories is bidirectional, as we need semantic knowledge to construct parsimonious descriptions of episodes and scenes.

The most widely studied type of short term memory is the working memory. This is a

temporary storage operating on a time scale of order of a second or hundred milliseconds. The brain uses this functionality to coordinate everyday tasks. Experiments show that the persistent activity (that is, increased population firing) of certain prefrontal areas correlate with working memory functionality. Separate working memory systems exist for different sensory systems to process signals of that modality. Capacity of the human working memory is around seven objects (anything that is stored in the semantic memory), leading to hypotheses that this functionality may be implemented by cortical oscillations described in [Section 9.4.2.2](#), the capacity resulting from the ratio of their frequencies.

9.4.4.3. *Learning problems and rules*

Learning as a mathematical problem involves adjusting the parameters of a model so that it minimizes some error function. Learning problems may be categorized into three families based on the nature of the available information and the goal.

In unsupervised or representational settings, we have a set of observations. The goal of the learner is to create a model that captures the regularities in the data. The learned model may be used for inference of latent variables or predicting future observations. A simple learning rule for adjusting weight parameters in neural models for unsupervised problems was proposed by Hebb (1949). This spike-timing dependent plasticity rule states that if a neuron's firing elicits firing in another with great reliability, the coupling between the two should be strengthened (in discrete time models this means simultaneous activation). This rule can be modified in many different ways to incorporate additional assumptions about the system to be modeled (Dayan and Abbott, 2001).

In supervised settings, we have a set of input–output pairs and the goal is to learn the mapping between the two sets of variables. The learned model may be used to predict outputs for new inputs. Here we can construct an explicit error function to minimize from the desired and the produced output for each input. A learning rule using gradient descent to find a local minimum in such a function is called the delta rule, or error backpropagation for feedforward neural networks (Rumelhart *et al.*, 1985), or Rosenblatt's algorithm for a single McCulloch–Pitts neuron.

In reinforcement learning (RL) settings, the learner is acting in an environment described by a state space. At each step, we observe the state of the environment and a possibly sparse reward signal, based on which we have to choose an action. The goal is to maximize reward in a long run, for which we need to learn values of visited states and generalize them to unvisited ones. Pavlovian conditioning and the Rescorla–Wagner rule can be regarded as simple examples of an RL paradigm. A general RL problem may be solved with the temporal difference learning rule (Sutton and Barto, 1998) that adjusts estimated state values based on the prediction error of the reward signal. A state space may be represented by neural networks, an idea that served as the basis of successful proof-of-concept applications (Mnih *et al.*, 2013). In the brain, neural systems using

dopamine as neurotransmitter, population activity is thought to encode reward prediction (Waelti *et al.*, 2001).

9.4.4.4. *Storage of information in neural systems*

Mapping between different variables can be stored in a feedforward network of rate neurons trained in a supervised manner. These mappings can be generalized to inputs never seen, with varying efficiency. Another classical way to store patterns in a neurons is to use a Hopfield network (Hopfield, 1982), which is a recursive system of McCulloch–Pitts neurons with symmetric weights. Elements of the learning set can be stored in the weight matrix of the network using a Hebbian learning rule. This creates point attractors in the phase space of the system at the patterns to store (and unfortunately at some other patterns), so when applying a new activation to the network, it will converge to the closest stored vector.

A probabilistic extension of the Hopfield network is the Boltzmann machine (Ackley *et al.*, 1985), which is able to store probability distributions of binary (and with some extensions, other) variables. The MCP neurons are replaced with probabilistic ones, making the neuron draw a sample from a Bernoulli distribution at each time step. A network may have unobserved variables too, for which the learning rule sets the weights so that on the observed units the activation statistics matches that of the training set up to pairwise correlations.

9.4.5. *Measuring Cognitive Systems*

To assess the validity of different modeling approaches to cognitive systems, one needs to collect data from the brain. In general, this is a very challenging task due to the enormous number of computing units and potentially relevant physiological quantities, and to the invasive nature of most direct measurement techniques in existence today.

9.4.5.1. *Electrophysiology*

As we believe that electric properties of the neurons are central to their information processing, direct measurement of membrane potentials is a primary goal. Single cells can be measured with high accuracy with the patch-clamp technique, which is limited in the number of possible parallel recordings. If one would like to measure population activity, extracellular electrodes or electrode arrays can record the LFP. However, such recordings pose rather challenging mathematical problems when one tries to obtain more direct information of cellular activity either by current source density reconstruction (Somogyvári *et al.*, 2012), or spike train reconstruction by waveform clustering (Buzsáki, 2004).

Such invasive techniques can only be used in animal experiments, and in humans only

in the case when a patient needs electrophysiological localization of damaged tissue to be removed by surgery. A non-invasive electrophysiological technique is the electroencephalogram (EEG), which records the neural electric field outside of the skull, providing a temporally detailed picture of macroscopic changes in population activity of neurons. However, EEG is limited in the sense of localization of the source of specific patterns. The magnetoencephalogram (MEG) is somewhat better in this regard in return for a vastly higher price compared to the low-cost EEG.

9.4.5.2. *Other imaging methods*

Indirect measurements of cellular activity record a physical quantity that can be regarded a proxy to the electric activity of the cells. Methods based on magnetic resonance (MR) are very popular due to their noninvasive nature and high spatial resolution over the whole brain. When applied for functional imaging (fMRI), localization of activated brain areas is possible, however the temporal resolution is in the order of a second, thus one cannot see temporally detailed responses. Seeing the activity of single cells is generally not possible with fMRI today, however, there are some advancements in this direction (Radecki *et al.*, 2014).

Imaging techniques are not only useful to localize task-specific activity, but it is also possible to infer functional connectivity from such data. Interaction networks of brain areas are also task-specific or related to a resting state. Connectivity analysis usually relies on a model of area and stimulus interactions (Friston *et al.*, 2003), and applicable to compare different groups of subjects. One direction of application is the characterization of neurological diseases, such as schizophrenia, which is thought to be a disconnection syndrome (Bányai *et al.*, 2011).

Some other functional imaging methods also promise better quality data about cellular activity. These include intrinsic optical imaging, voltage sensitive dyes and calcium imaging.

9.4.5.3. *Psychophysics*

Cognitive models can make many predictions about performance in different tasks. These can be checked without making invasive or otherwise expensive measurements of neural activity and provide indirect evidence about the plausibility of different models. Psychophysics was founded in the 19th century by Fechner (Fechner, 1907), and since then it is a very important tool for cognitive science. Sensory classification tasks measured in psychophysical experiments also shed light on the nature of representations in the cortex. For example, in the visual areas, the structure low-level representations in the visual cortex can be examined by texture stimuli (as introduced by Julesz (1962)). Higher-level representations of objects is phenomenologically described by Gestalt psychology (Koffka, 2013). These results can be combined with physiology to assess the relevance of

particular models on the neural level.

9.5. Cognitive Robotics

9.5.1. Challenges of Automation

9.5.1.1. Typical settings and problems

Nowadays, robotics is starting to leave behind the assembly line and the futuristic demos on tech shows and university press events. Everyday interaction with autonomous machines is a prospect of the near future, either in the form of a software running on a general-purpose computer or a custom-built hardware that moves around. Examples of autonomous machines in testing or production phase include self-driving cars, drone swarms and speech recognition and synthetization systems available in mobile computing devices.

As the problems such agents need to solve resemble the situations living organisms find themselves every day, we will find that a strong parallelism exists between the tools used for modeling neural cognitive systems and the ones used to produce adaptive behavior in machines. These methods, together with other effective algorithms, are undergoing rapid improvement in the fields of machine learning and artificial intelligence. When creating physically embodied agents, these methods need to be augmented by advancements from mechanical, electrical and software engineering, control theory and signal processing, and in terms of adaptive algorithms, a larger emphasis is placed on strategy learning and decision-making. Thus, cognitive robotics is formed as a cooperation between diverse fields of multiple scientific disciplines.

Applications of such robots go beyond civil use, as military or astronautical purposes create a major drive to these developments all around the globe.

9.5.1.2. Building models of the environment and the agent

The core problem of adaptive behavior is to represent the world and the robot itself in an efficient manner to support perception, decision and action. Formation of such representations is closely related to the phenomenon of internal models in the human brain, as detailed in [Section 9.4.3](#). However, when building a robot, different constraints are in effect regarding the cost of certain computations or storage. Thus, for every problem in cognitive neuroscience there is a parallel problem in machine learning, where instead of asking what is the way nature solves a given task, we seek the optimal solution with respect to the constraint of the computational architecture at hand.

The knowledge the agent needs to represent include the encoding of existing objects, their properties, relationships between them, concepts as correlation of objects in time or space and the agent itself, that is, the possible actions it can take and their effects on the environment. Unlike to the study of biological systems, the language of representation and code of communication are not restricted in artificial agents by anything but the

capabilities of their hardware.

In realistic settings, we typically cannot define a good representation by hand. Thus, automatic formation of representation is often defined as an unsupervised learning problem, where the optimal representation is sought in such a way that it supports optimal decision (classification, action selection) the best. Unsupervised learning algorithms exist for many modeling frameworks (Ghahramani, 2004), and with the advent of large-scale datasets (typically at companies providing websites where users can store media content) they can be tested in problems with near-realistic complexity.

9.5.1.3. *Learning architectures*

One of the central concepts of cybernetics was the theory of feedback loops that can implement adaptive control for autonomous agents. The simplest problems may be solved without adaptation, as illustrated by the Braitenberg vehicles (Braitenberg, 1986) modeling primitive animal behavior. More complex problems require the use of adaptive dynamical systems that are modifying their parameters based on optimality criteria as a feedback mechanism.

In complex environments, generalization to unseen situations is of primary importance, as the agent has typically no chance of exploring all possible states or inputs, thus it has to extrapolate to unseen ones based on some metric of the state or input space. Artificial neural networks (see [Section 9.4.3.2](#)) are universal function approximators with good generalization properties, and they form the basis of many problem models in autonomous agents.

If the robot needs to represent the uncertainty of modelled quantities, probabilistic models can be employed (Bishop, 2006). Advantages of this approach are that probability is a framework unifying many different modeling and algorithmic approaches on a principled ground, and it may be regarded as a natural extension to binary logic representing belief in propositions (Jaynes, 2003). Moreover, the framework of graphical models provides an intuitive language of model creation and improvement.

Probabilistic neural networks combine the two approaches. They are the state-of-the-art solution to many benchmark and real-world problems, as they can be trained with effective algorithms in unsupervised and supervised settings as well.

9.5.2. *Sensory Processing*

9.5.2.1. *Learning invariances*

The basic task of processing a sensory input is to classify it according to different criteria, e.g., the presence of a specific object or object type in it. To perform such classifications, one has to differentiate regularities of the phenomenon to discover from regularities

arising due to finite data (for example, when deciding whether we see a lion, we have to take into account the fact that lions may appear in various positions in front of various backgrounds, but if we have seen only male lions, one without a mane might be misclassified).

Learning invariances is a potentially very challenging task, in which humans exhibit a performance which is hardly matched by any algorithm so far, especially in terms of robustness to change of context.

9.5.2.2. *Image processing*

Many robotic applications need to deal with some form of visual input in the form of images or video streams. Typically, one needs to segment the images to detect the presence of certain objects. Invariances that need to be handled during visual object recognition include rotation, translation, resizing (possibly due to perspective), change of background and possibly color, relative position of parts or stretching.

For the purpose of testing and development of algorithms, several benchmark databases exist, such as the MNIST database of handwritten images and the ImageNet database with lots of labels attached to the images by humans.

The most successful solutions for image classification are deep neural networks that capitalize on the hierarchical nature of image composition by fitting a layered architecture to observations. Unsupervised pre-training of layers can enhance the performance greatly, as it constructs a representation that is related to the inherent statistics of the input set (which may contain much more data than the set we have labels provided for), which can be refined by adding the information of the available labels in a supervised manner.

9.5.2.3. *Speech processing*

Effective interaction with humans is mostly possible through speech. It is different from visual input in the sense that it is typically lower dimensional, but temporally extended sequential data with high autocorrelations.

To model such a dataset, one needs to employ dynamical models that explicitly represent temporal dependence. The class of Hidden Markov Models (HMMs) offers the simplest solution to this problem. More complex, nonlinear relationships may be captured by recurrent neural networks, in which feedback loops are placed on processing layers representing time. Learning algorithms defined for feedforward networks may be extended for such loopy architectures relatively easily.

Speech processing solutions are possibly the branch of robotics that pervaded everyday life the most in the form of mobile phone software that not only processes, but also produces speech. This may be accomplished by constructing a world model that defines not only the grammar but topical relationships between words.

9.5.2.4. Understanding content

The hardest problems of signal processing can be described as semantic analysis, as we typically look for the meaning of the input by making sense of a visual scene or an uttered sentence. This cannot be accomplished solely on the basis of pattern mining, but one needs to construct a world model that effectively encodes relationships between concepts. Semantics can indeed be defined as a higher level in the hierarchy of syntaxes.

In visual processing, slow feature analysis (Wiskott and Sejnowski, 2002) aims to detect temporally stable objects, creating object hierarchies building up visual scenes. In natural language processing, document topic models, such as Latent Dirichlet Allocation (LDA) (Blei *et al.*, 2003) try to model word correlations by identifying document categories.

We can also attempt to discover the model structure that describes the data best instead of hand-picking it. Model selection or structure learning (Kemp and Tenenbaum, 2008) is the unsupervised discovery of the model form that can fit to the observations best.

9.5.3. Decision-Making

9.5.3.1. Utility and cost functions

When deciding about which action to take, it is not enough to determine what is the effect of the options according to the internal model, we also have to define how desirable those outcomes are. This can be described by a cost or utility function that encodes the undesirability or equivalently, the desirability of situations or actions.

As Kahneman and Tversky (Tversky and Kahneman, 1981) noted, cost functions of humans deviate from expected reward, reflecting priorities auxiliary to the task at hand (they also show that equivalent reformulation of problems lead to different decisions, which behavior is best kept at minimum in artificial agents).

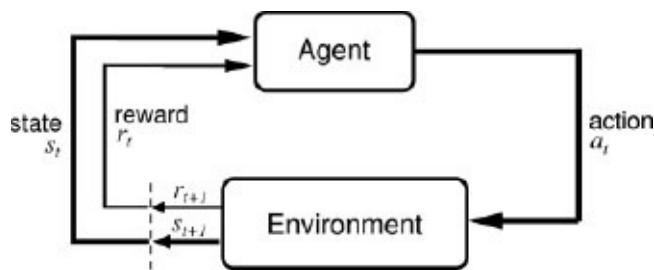


Figure 9.5: Agent-environment interplay in a MDP.

Simple decision-making protocols may be defined on the basis of predefined cost functions, but more complex problems require the adaptation of the estimation of such costs or values based on experience.

9.5.3.2. Strategy learning

For learning strategies, it is useful to model the environment and the agent as a set of states and associated rewards, a set of actions, and a transition probability function defined on triplets of two states and an action. Such a model is called a Markov Decision Process (MDP), and reinforcement learning methods operate on it to find the best strategy to collect reward.

Finding the optimal action to take at each state of the environment is a reinforcement learning (RL) problem as described in [Section 9.4.4.3](#). Well-performing RL solvers update their belief about values of states using some variant of the temporal difference (TD) learning rule:

$$V(s_t) \leftarrow V(s_t) + \alpha [r_{t+1} + \gamma V(s_{t+1}) - V(s_t)], \quad (10)$$

where s_t is the state the environment is in at time t , V is the belief about the value of the state, r is the reward signal, γ and α are parameters of the update rule. The TD rule makes use of the error in the prediction of subsequent reward by the estimated value of a state. Action selection may either be based on a separately learned state transition probability function, or the estimation of a state-action value function instead of a state-value.

Reinforcement learning using a neural representation was used to create proof-of-concept applications playing games against human players (Tesauro, 1995) and in virtual environments (Mnih *et al.*, 2013).

9.5.3.3. Active learning

In MDP environments, actions of the agent influence the learning process greatly, as data collection is dependent of the actual chain of states it visits. Thus, it is important to choose an action selection strategy that maximizes reward in the long run by finding the best strategy, and when it is reasonable to assume that the agent won't find any better, employ it. This is the classic exploration versus exploitation dilemma (Sutton and Barto, 1998), which is usually resolved in a way that initially the agent act random to explore the environment as much as possible, and as reward accumulates, thus state value estimations get more reliable, it chooses the best action with a proportionally greater probability. This way we can ensure that in theory (that is, with an arbitrary amount of learning epochs) the globally optimal strategy can be found.

9.5.3.4. Learning to move mechanical bodies

Moving machines can take many forms. One category is vehicle-like, such as cars, drones (Virág *et al.*, 2014) or submarines, which have simpler movement patterns and easier to control, in return they are limited in terms of terrain difficulty. Alternatively, robots may mimic living organisms not only in terms of behavior, but also physical form. These solutions include bipedals, four-legged robots mimicking dogs (Touretzky and Tira-

Thompson, 2005) or mules, and structures with more legs. A great advantage of such constructions is that they can manage more difficult terrains such as rocks or steps.

Complex movement of animal-like structures can be copied from actual animals. By motion capture techniques, one can record the movement of an animal or a human to time series of coordinates, and look for periodic patterns or other primitives in them that can be transcribed to the robot's program.

Optimal movement can be and often is obtained by reinforcement learning. Then the state is defined by some parametrization of the joints or motors of the robot, and the reward function is constructed in such a way that it encourages the reaching of a goal or keeping moving, and punishes loss of stability. This way complex movement patterns can be acquired without having (or being able at all) to describe them in terms of joint coordinates.

9.5.3.5. *Cooperative agents*

Agents aiming for optimal behavior in realistic settings need to take into account other goal-directed entities acting in the environment. Moreover, if the agents share the same goal, they can perform better (or solve otherwise intractable tasks) in a cooperative manner. A simple form of this behavior can be observed in insect colonies that inspired a class of optimization algorithms, Ant Colony Optimization (ACO). More generally, swarm intelligence is a field of research where complex behavior is assembled as a cooperation between simple individual agents (Bonabeau *et al.*, 1999).

A higher level of cooperation can be realized not only on the behavioral, but on the hardware level, where simple agents perform the self-assembly of a more complicated structure (Gro *et al.*, 2006).

Strategy learning in a cooperative setting may be approached by reinforcement learning techniques, however, such a situation can only be modelled by a partially observable MDP (POMDP) due to the fact that effects of actions of other agents may confound the state transitions of the environment. Optimal behavior in a POMDP is a heavily researched problem (Bucsoniu *et al.*, 2010).

Cooperative RL is a very ambitious goal in general, but advancements can be realized and tested in well-controlled game situations, such as the RoboCup annual football championship organized in different categories exclusively for teams of robotic agents.

9.6. Conclusion

In this chapter, we reviewed the philosophical and mathematical treatment of cognitive systems modeling, including directions that aim to understand biological systems, and to build artificial solutions to problems encountered by living organisms. We gave an overview of the basic approaches of cognitive modeling, symbolic, connectionist and hybrid, and the computational tools and architectures that can be used to model phenomena in either of those settings. Possibilities of connecting the computational picture to biological neurons were shown, and applications where cognitive principles were successfully adopted in artificial systems were overviewed.

The chapter is meant to be an introduction to the very diverse field of cognitive science, with the references serving as pointers to the branches of study mentioned. We hoped to convey a picture of the frontier in the great endeavor to understand the human brain as it stood in 2014. As the reader surely concluded based on the fragmentary nature of success stories, we are only at the beginning of the work. We are starting to grasp concepts in sufficient detail to predict behavior and sometimes even physiology, but assembling a composite picture will require much more effort. The central question of such an integrated model will be the nature of neural codes and representations, as advancing in those topics may allow cognitive science to be a proper natural science with predictive models and repeatable experiments. Nowadays, multiple large-scale programs are being initiated in neuroscience aiming for an integrative approach, and their results are expected to unfold in the next decade.

The most complex problems in cognitive science, such as social systems or the hard problem of consciousness, may also be addressed in a more satisfying manner when we have a more complete understanding of the formal treatment of thinking and problem solving systems.

References

- Abbott, L. F. (2005). Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain Res. Bull.*, pp. 1–2.
- Abraham, T. (2002). (Physio)logical circuits: The intellectual origins of the McCulloch–Pitts neural networks. *J. Hist. Behav. Sci.*, 38, pp. 3–25.
- Ackley, D. H., Hinton, G. E. and Sejnowski, T. J. (1985). A learning algorithm for boltzmann machines. *Cognit. Sci.*, 9(1), pp. 147–169.
- Anderson, J. R. (2007). *How Can the Human Mind Occur in the Physical Universe?* New York: Oxford University Press.
- Anderson, J. R. and Bower, G. H. (1973). *Human Associative Memory*. Washington: Winston and Sons.
- Arbib, M. (2000). Warren McCulloch's search for the logic of the nervous system. *Perspect. Biol. Med.*, 43, pp. 193–216.
- Arbib, M., Érdi, P. and Szentágothai, J. (1997). *Neural Organization: Structure, Function, and Dynamics*. Cambridge, MA: MIT Press.
- Arbib, M. A. (2005). From monkey-like action recognition to human language: An evolutionary framework for neurolinguistics. *Behav. Brain Sci.*, 28, pp. 105–168.
- Barlow, H. B. (1990). Conditions for versatile learning, Helmholtz's unconscious inference, and the task of perception. *Vis. Res.*, 30(11), pp. 1561–1571.
- Bastos, A. M., Usrey, W. M., Adams, R. A., Mangun, G. R., Fries, P. and Friston, K. J. (2012). Canonical microcircuits for predictive coding. *Neuron*, 76(4), pp. 695–711.
- Battaglia, F. P. and Pennartz, C. M. A. (2011). The construction of semantic memory: Grammar-based representations learned from relational episodic information. *Front. Comput. Neurosci.*, 5, p. 36.
- Bi, G. and Poo, M. (1998). Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *J. Neurosci.*, 18, pp. 10464–10472.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*, Vol. 1. New York: Springer.
- Blei, D. M., Ng, A. Y. and Jordan, M. I. (2003). Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3, pp. 993–1022.
- Bobrow, D. and Norman, D. (1975). Some principles of memory schemata. In Bobrow, D. and Collins, A. (eds.) *Representation and Understanding*. New York: Academic Press.
- Bonabeau, E., Dorigo, M. and Theraulaz, G. (1999). From natural to artificial swarm intelligence.
- Braitenberg, V. (1986). *Vehicles: Experiments in Synthetic Psychology*. Cambridge: MIT Press.
- Bressler, S. L. (1990). The gamma wave: A cortical information carrier? *Trends Neurosci.*, 13(5), pp. 161–162.
- Brooks, R. (2002). *Flesh and Machines: How Robots Will Change Us*. New York: Pantheon Books.
- Buzsáki, G. (2004). Large-scale recording of neuronal ensembles. *Nat. Neurosci.*, 7(5), pp. 446–451.
- Buzsáki, G. (2004). Neuronal oscillations in cortical networks. *Science*, 304(5679), pp. 1926–1929.
- Busoniu, L., Babuška, R. and De Schutter, B. (2010). Multi-agent reinforcement learning: An overview. In *Innovations in Multi-Agent Systems and Applications* — 1. Berlin, Heidelberg: Springer, pp. 183–221.
- Bányai, M., Diwadkar, V. A. and Érdi, P. (2011). Model-based dynamical analysis of functional disconnection in schizophrenia. *NeuroImage*, 58(3), pp. 870–877.
- Carandini, M. and Heeger, D. J. (2012). Normalization as a canonical neural computation. *Nat. Rev. Neurosci.*, 13(1), pp. 51–62.
- Chomsky, N. (1959). A review of B. F. Skinner's verbal behavior. *Lang.*, 35, pp. 26–58.
- Chomsky, N. (1965). *Aspects of the Theory of Syntax*. Cambridge: MIT Press.
- Churchland, P. and Sejnowski, T. (1990). Neural representation and neural computation. *Philos. Perspect.*, 4, pp. 343–382.
- Collins, A. and Quillian, M. (1969). Retrieval time for semantic memories. *J. Verbal Learn. Verbal Behav.*, 8, pp. 240–48.

- Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Mach. Lear.*, 20(3), pp. 273–297.
- Craik, K. (1943). *The Nature of Explanation*. Cambridge: Cambridge University Press.
- Davis, R., Shrobe, H., and Szolovits, P. (1993). What is a knowledge representation? *AI Mag.*, 14(1), pp. 17–33.
- Dayan, P. and Abbott, L. F. (2001). *Theoretical Neuroscience*, Vol. 31. Cambridge, MA: MIT press.
- DiCarlo, J. J., Zoccolan, D. and Rust, N. C. (2012). Perspective. *Neuron*, 73(3), pp. 415–434.
- Duch, W., Oentaryo, R. and Pasqueier, M. (2007). Cognitive architectures: Where do we go from here? In Wang, P., Goertzel, B. and Franklin, S. (eds.), *Front. Artif. Intell. Appl.*, 171, pp. 122–136.
- Érdi P. and Somogyvári, Z. (2002). Post-hebbian learning algorithms. In Arbib, M. (ed.), *The Handbook of Brain Theory and Neural Networks, Second Edition*. Cambridge, MA: MIT Press, pp. 533–539.
- Fechner, G. T. (1907). *Elemente der Psychophysik*, Vol. 2. Breitkopf & Härtel.
- Fiser, J., Berkes, P., Orbán, G. and Lengyel, M. (2010). Statistically optimal perception and learning: From behavior to neural representations. *Trends Cognit. Sci.*, 14(3), pp. 119–130.
- Fodor, J. (1980). Methodological solipsism considered as a research strategy in cognitive psychology. *Behav. Brain Sci.*, 3, pp. 63–109.
- Fodor, J. (1983). *The Modularity of Mind: An Essay on Faculty Psychology*. Cambridge: MIT Press.
- Friston, K. J., Harrison, L. and Penny, W. (2003). Dynamic causal modelling. *NeuroImage*, 19(4), pp. 1273–1302.
- George, D. and Hawkins, J. (2009). Towards a mathematical theory of cortical micro-circuits. *PLoS Comput. Biol.*, 5(10), p. e1000532.
- Ghahramani, Z. (2004). Unsupervised Learning. In *Advanced Lectures on Machine Learning*. Berlin, Heidelberg: Springer, pp. 72–112.
- Gro, R., Bonani, M., Mondada, F. and Dorigo, M. (2006). Autonomous self-assembly in swarm-bots. *IEEE Trans. Robot.*, 22(6), pp. 1115–1130.
- Harris, K. D. and Thiele, A. (2011). Cortical state and attention. *Nat. Rev. Neurosci.*, 12(9), pp. 509–523.
- Hebb, D. O. (1949). *The Organisation of Behavior, a Neuropsychological Theory*. New York: Wiley.
- Hebb, D. O. (1949). *The Organization of Behavior: A Neuropsychological Approach*. New York: John Wiley & Sons.
- Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8), pp. 1771–1800.
- Hinton, G. E., Dayan, P., Frey, B. J. and Neal, R. M. (1995). The “wake–sleep” algorithm for unsupervised neural networks. *Science*, 268(5214), pp. 1158–1161.
- Hodgkin, A. L. and Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.*, 117(4), pp. 500–544.
- Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. In *Proc. Nat. Acad. Sci.*, 79(8), pp. 2554–2558.
- Hubel, D. H. and Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *J. Physiol.*, 195(1), pp. 215–243.
- Iturria-Medina, Y., Sotero, R. C., Canales-Rodríguez, E. J., Alemán-Gómez, Y. and Melie García, L. (2008). Studying the human brain anatomical network via diffusion-weighted MRI and graph theory. *NeuroImage*, 40(3), pp. 1064–1076.
- Jackson, P. (1998). *Introduction To Expert Systems, Third Edition*. USA: Addison Wesley.
- Jaynes, E. T. (2003). *Probability Theory: The Logic of Science*. Cambridge: Cambridge University Press.
- Johnson-Laird, P. (1983). *Mental Models: Towards a Cognitive Science of Language, Inference, and Consciousness*. Cambridge: Cambridge University Press.
- Julesz, B. (1962). Visual pattern discrimination. *IRE Trans. Inf. Theory*, 8(2), pp. 84–92.
- Jurafsky, D. and Marti, J. (2008). *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics, Second Edition*. Englewood Cliffs, NJ: Prentice-Hall.

- Kemp, C. and Tenenbaum, J. B. (2008). The discovery of structural form. *Proc. Nation. Acad. Sci.*
- Koffka, K. (2013). *Principles of Gestalt Psychology*. Routledge.
- Laird, J. (2012). *The Soar Cognitive Architecture*. Cambridge: MIT Press.
- Lanciego, J. L. and Wouterlood, F. G. (2011). A half century of experimental neuroanatomical tracing. *J. Chem. Neuroanat.*, 42(3), pp. 157–183.
- Lengyel, M. and Dayan, P. (2007). Hippocampal contributions to control: The third way. *Adv. Neural Inf. Process. Syst.*
- Lengyel, M., Szatmáry, Z. and Érdi, P. (2003). Dynamically detuned oscillations account for the coupled rate and temporal code of place cell firing. *Hippocampus*, 13(6), pp. 700–714.
- Ma'ayan, A., Jenkins, S. L., Neves, S., Hasseldine, A., Grace, E., Dubin-Thaler, B., Eungdamrong, N. J., Weng, G., Ram, P. T., Rice, J. J., Kershenbaum, A., Stolovitzky, G. A., Blitzer, R. D. and Iyengar, R. (2005). Formation of regulatory patterns during signal propagation in a Mammalian cellular network. *Science*, 309(5737), pp. 1078–1083.
- Ma, W. J., Beck, J. M., Latham, P. E. and Pouget, A. (2006). Bayesian inference with probabilistic population codes. *Nat. Neurosci.*, 9(11), pp. 1432–1438.
- McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous system. *Bull. Math. Biophys.*, 5(4), pp. 115–133.
- Milnes, B., Pelton, G., Doorenbos, R., Laird, M., Rosenbloom, P. and Newell, A. (1992). A specification of the soar cognitive architecture in z. Technical Report. Pittsburgh, PA, USA: Carnegie Mellon Univ.
- Minsky, M. (1974). A framework for representing knowledge. *Technical Report*. Cambridge: MIT AI Laboratory Memo 306.
- Minsky, M. and Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA: MIT Press.
- Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M. (2013). Playing atari with deep reinforcement learning.
- Mountcastle, V. B. (1997). The columnar organization of the neocortex. *Brain*, 120(Pt 4), pp. 701–722.
- Neal, R. M. (1992). Connectionist learning of belief networks. *Artif. Intell.*, 56(1), pp. 71–113.
- Newell, A. (1990). *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.
- Newell, A. and Simon, H. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall.
- Newell, A. and Simon, H. (1976). Computer science as empirical enquiry: Symbols and search. *Commun. Assoc. Comput. Mach.*, 19, pp. 113–126.
- Pearl, J. (2000). *Causality: Models, Reasoning and Inference*, Vol. 29, Cambridge: Cambridge University Press.
- Piccinini, G. (2004). The first computational theory of mind and brain: A close look at Mcculloch and Pitts's 'logical calculus of ideas immanent in nervous activity'. *Synthese*, 141, pp. 175–215.
- Pinker, S. (2007). *The Language Instinct: How the Mind Creates Language*. New York: Harper Perennial Modern Classics.
- Pinker, S. and Bloom, P. (1990). Natural language and natural selection. *Behav. Brain Sci.*, 13, pp. 707–784.
- Quillian, M. (1968). Semantic memory. In Minsky, M. (ed.), *Semantic Information Processing*. Cambridge, MA: MIT Press.
- Quiroga, R. Q., Reddy, L., Kreiman, G., Koch, C. and Fried, I. (2005). Invariant visual representation by single neurons in the human brain. *Nature*, 435(7045), pp. 1102–1107.
- Radecki, G., Nargeot, R., Jelescu, I. O., Le Bihan, D. and Ciobanu, L. (2014). Functional magnetic resonance microscopy at single-cell resolution. *Aplysia Californica. Proc. Nation. Acad. Sci.*, 111(23), pp. 8667–8672.
- Ramón y Cajal, S. (1909). *Histologie du système nerveux de l'homme et des vertébrés*. Maloine.
- Rizzolatti, G., Sinigaglia, C. and Anderson, F. (2008). *Mirrors in the Brain: How Our Minds Share Actions and Emotions*. New York, US: Oxford University Press.

- Rosenblatt, F. (1962). *Principles of Neurodynamics*. Washington, DC: Spartan Books.
- Rosenblueth, A. and Bigelow, J. (1943). Behaviour, purpose and teleology. *Philos. Sci.*, 10, pp. 18–24.
- Rosenblueth, A., Wiener, N. and Bigelow, J. (1943). Behavior, purpose and teleology. *Philos. Sci.*, 10(1943), S. 18–24.
- Rumelhart, D. and McClelland, J. (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Cambridge, MA: MIT Press.
- Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1985). Learning internal representations by error propagation.
- Salakhutdinov, R. R. and Hinton, G. E. (2009). Deep boltzmann machines. *Proc. Int. Conf. Artif. Intell. Stat.*, pp. 448–455.
- Schank, R. and Abelson, R. (1977). *Scripts, Plans, Goals and Understanding*. Hillsdale, NJ: Erlbaum.
- Searle, J. (1983). *Intentionality: An Essay in the Philosophy of Mind*. Cambridge: Cambridge University Press.
- Searle, J. (1992). *The Rediscovery of the Mind*. Cambridge, MA: MIT Press.
- Shepard, R. (1980). *Internal Representations: Studies in Perception, Imagery, and Cognition*. Bradford: Montgomery, VT.
- Shouval, H. (2007). Models of synaptic plasticity. <http://www.scholarpedia.org/article/>.
- Sjoestrom, J. and Gerstner, W. (2010). Spike-timing dependent plasticity. <http://www.scholarpedia.org/article/>.
- Smith, A. M. (2001). *Alhacen's Theory of Visual Perception:A Critical Edition, with English Translation and Commentary, of the First Three Books of Alhacen's De Aspectibus, the Medieval Latin Version of Ibn Al-Haytham's Kitab al-Manazir*, Vol.1. USA: American Philosophical Society.
- Somogyvári, Z., Cserpán, D., Ulbert, I. and Érdi, P. (2012). Localization of single-cell current sources based on extracellular potential patterns: The spike CSD method. *Eur. J. Neurosci.*, 36(10), pp. 3299–3313.
- Strogatz, S. H. (2006). *Nonlinear Dynamics and Chaos (With Applications to Physics, Biology, Chemistry*. Perseus Publishing.
- Strotzer, M. (2009). One century of brain mapping using brodmann areas. *Clin. Neuroradiol.*
- Sun, R. (2004). Desiderata for cognitive architectures. *Philos. Psychol.*, 17, pp. 341–373.
- Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning*. Cambridge: MIT Press.
- Szentágothai, J. (1983). The modular architectonic principle of neural centers. In *Reviews of Physiology, Biochemistry and Pharmacology*, Vol. 98. Berlin, Heidelberg: Springer, pp. 11–61.
- Tesauro, G. (1995). Temporal difference learning and TD-Gammon. *Commun. ACM*.
- Thagard, P. (2005). *Mind: Introduction to Cognitive Science, Second Edition*. Cambridge, MA: MIT Press.
- Touretzky, D. S. and Tira-Thompson, E. J. (2005). Tekkotsu: A framework for AIBO cognitive robotics, *Proceedings of the National Conference on Artificial Intelligence*, Vol. 20(4), Menlo Park CA; Cambridge, MA; London: AAAI Press; MIT Press, p. 1741.
- Tsuda, I. (1991). Chaotic itinerancy as a dynamical basis of hermeneutics in brain and mind. *World Futures: J. Gen. Evol.*, 32(2–3), pp. 167–184.
- Tulving, E. (1985). How many memory systems are there? *Am. Psychol.*, 40(4), p. 385.
- Tversky, A. and Kahneman, D. (1981). The framing of decisions and the psychology of choice. *Science*, 211(4481), pp. 453–458.
- Virág, C., Vásárhelyi, G., Tarcai, N., Szörényi, T., Somorjai, G., Nepusz, T. and Vicsek, T. (2014). Flocking algorithm for autonomous flying robots. *Bioinspir. Biomim.*, 9(2), p. 025012.
- Vizi, E. S., Kiss, J. P. and Lendvai, B. (2004). Nonsynaptic communication in the central nervous system. *Neuroche. Int.*, 45(4), pp. 443–451.
- von Neumann, J. (1958). *The Computer and the Brain*. New Haven: Yale Univ. Press.
- Von Helmholtz, H. (1866). *Handbuch der physiologischen Optik: mit 213 in den Text eingedruckten Holzschnitten und 11 Tafeln*. Vol.9, Voss.

- Waelti, P., Dickinson, A. and Schultz, W. (2001). Dopamine responses comply with basic assumptions of formal learning theory. *Nature*, 412(6842), pp. 43–48.
- Werbos, P. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis. Cambridge: Harvard University.
- Winograd, T. (1972). *Understanding Natural Language*. Waltham, MA: Academic Press.
- Wiskott, L. and Sejnowski, T. J. (2002). Slow feature analysis: Unsupervised learning of invariances. *Neural Comput.*, 14(4), pp. 715–770.
- Zhao, C., Deng, W. and Gage, F. H. (2008). Mechanisms and functional implications of adult neurogenesis. *Cell*, 132(4), pp. 645–660.
- Ziemba, C. M., Heeger, D. J., Simoncelli, E. P., Movshon, J. A. and Freeman, J. (2013). A functional and perceptual signature of the second visual area in primates. *Nat. Neurosci.*, 16(7), pp. 1–12.

¹ The phrase mental models has a specific, more technical meaning in Johnson-Lairds work than in Craiks account (Johnson-Laird, 1983).

² One of the two founding papers of cybernetics addressed this problem (Rosenblueth and Bigelow, 1943).

Chapter 10

A New View on Economics with Recurrent Neural Networks

*Hans-Georg Zimmermann, Ralph Grothmann
and Christoph Tietz*

The identification of dynamical systems for forecasting and diagnosis plays an important role in technology as well as in economics. The special challenge in economics is that less *a priori* knowledge is available and the focus lies on the modeling of observed dynamical systems. We will explain our considerations in an economical framework, nevertheless, most of them can be applied to technical examples too.

System identification and forecasting in economics can be split. On the one hand, there are volume (e.g., demand- or load-) forecasting support marketing tasks, on the other hand there are price forecasts, which are essentially financial market forecasts. Market models are often not far away from being world models, presenting an extreme challenge.

Neural networks can not only meet this challenge but also provide unique insights into the nature and hidden structure of the identification problem.

10.1. Understanding Economics

This chapter shows the different challenges in marcoeconomics, business economics and finance which can be met with the help of neural networks (NN). The mathematics of NN allow insights into the market analysis which would not be possible without this approach.

10.1.1. Markets as Recurrent Dynamical Systems

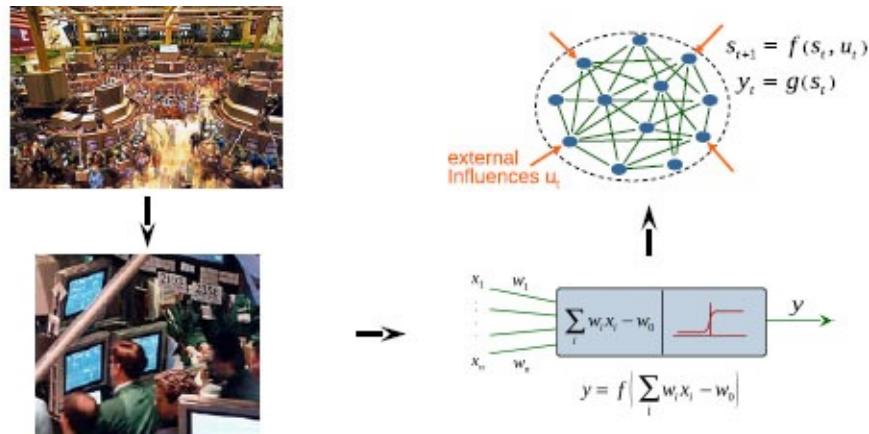
Markets are high-dimensional, nonlinear systems. The dimensionality stems from the worldwide interactions between multiple agents, individual decisions of the agents can be seen as step functions which cause a nonlinear dynamic. An argument which is often used to defend linear models is that a Tailor expansion can be used to build a model and it is legitimate to start with the first term of the expansion to analyze a preliminary linear model. Especially in a fast changing world such an argument is questionable because a Tailor expansion can only be accurate around the expansion point. This will never be good enough in a world with fast changing dynamics.

Instead of modeling a complete market at once, let us focus first on the behavior of a single decision-maker, a trader. She has thousands of numbers available to her, arriving continuously from the data providers. Somewhere in these numbers the worldwide activity of the relevant markets is described. No one can grasp such an amount of information, therefore information filtering is an essential part of the decision-making process. This information filtering differs between traders, sometimes even conceptually like in the different systems of chartists or fundamentalists. With this focus of attention, a trader analyzes price changes in the market. The outcome of this analysis is an opinion about the emergence of a future market pattern. If the trader is confident of her analysis, she will sign a business contract.

These three steps—filtering, aggregation of information and decision-making can be efficiently described with the mathematical model of a single neuron (see the lower right of [Figure 10.1](#)). The filtering is realized by some zero weights on the incoming connections, the aggregation is modelled as a weighted sum and the decision step can be seen as evaluation of a nonlinear transfer function.

One trader alone does not make a market. Only the interaction between traders in a complex sequence of decisions and actions constitutes the dynamics of a market. The decisions observed over time can be discretized in form of a recurrent dynamical system with state equations as given in the upper right of [Figure 10.1](#). A state s_t with observations u_t leads to the new state s_{t+1} . The state transformation equation defines the microeconomic picture of the interaction between all the microscopic decisions of individual traders. Unfortunately, we are not able to observe the complete microscopic dynamics. We can only observe the macroscopic result of all these decisions as the behavior of the market. The output equation of the dynamical system bridges the gap between micro- and macro-

economics with g as the bridging function and y_t as the observable subject of the market.



A neuron is a **decision model** combining **information filtering** & aggregation & **switching**.
A neural network can be seen as interaction of decisions and hence, as a market model.

Figure 10.1: Neural networks—from decision models to markets.

Is this simple mathematical model powerful enough to reproduce the different types of behavior in real markets? If individual decisions are uncorrelated, the overall market behavior might be very simple. On the other hand, if all the individual decisions correlate, then the behavior of real markets can become extremely chaotic. We will show that neural models are powerful enough to capture different types of markets. Model types do not have to be altered for different market conditions, neural models offer a reasonable microeconomic model and not only functional assumptions. Finally, we find at the core of neural models the adaptability of state transition and output functions to observed data.

10.1.2. Improving Business Economics with Recurrent Neural Networks

Business economics requires precise forecasts in order to enhance the quality of planning throughout the value chain. The complex planning and decision-making scenarios generally require the forecast to take account of a wide range of influencing factors with largely nonlinear cause and effect relationships.

An important application is demand forecasting for production control. One main challenge is the partial knowledge about the world, as competitors do not offer information about their decision-making. To overcome this lack of information, forecasting models can use the time-recursive structure of the observations to reconstruct information about the surrounding world which is not directly observable.

A second difficulty is that a demand forecast has to be applicable to new products which initially cannot provide training data over extended time intervals. Nevertheless, a reasonable demand forecast is necessary for production planning.

A third challenge arises where forecasts for a very large amount of variables have to be calculated. This is a typical situation in load forecasting (electricity, water or gas). If a load forecast has to be calculated for each quarter of an hour over one day, there are 96 variables to predict in form of a load curve. As will be shown in [Section 10.3](#), Recursive

Neural Networks (RNNs) which match the time recursive structure of the forecasting problem allow the simultaneous forecast of all 96 variables based on only one model.

10.1.3. Understanding Finance with Recurrent Neural Networks

The ability to forecast is a central requirement for rational decision-making, since the merit of any decision is always measured by its consequence in the future. This applies in particular to the financial sector, for example, in exchange rate trading or capital investment. The best known forecasting techniques in this context is chart analysis, which only evaluates data from the analyzed time series itself in the past. In contrast, fundamental analysis attempts to describe the actual dynamics of the market processes. The success of chart analyses is hampered by the low volume of input information, while that of a fundamental analysis is limited by the complexity of the market and the fact that it disregards the psychological factors influencing decision-making.

[Section 10.4.1](#) introduces the network architecture of Historically Consistent Neural Networks (HCNNs), which is especially well suited for financial market models. It breaks up the common distinction between input and output values of a network and closes the modeled dynamical system by regarding all input and output values external to the network commonly as uniform observables. Its capabilities are demonstrated with results from modeling copper markets at the European Energy Exchange (EEX) and the London Metal Exchange (LME). This network architecture is refined in [Section 10.4.2](#) to make better use of the available training information and speed up the training.

Markets can show different regimes, which can be characterized as based on (causal) historical developments, on (retro-causal) motivation and expectations or a mixture of both. We show in [Sections 10.4.4](#) and [10.4.5](#), how the prior knowledge about these regimes can be coded in neural network architectures which extend the basic HCNN to Causal-Retro-Causal Historical Consistent Neural Networks (CRCNNs) which are finally able to model the copper markets we introduced earlier.

As mentioned in [Section 10.1.1](#), Neural Networks can offer market models but for valuable decision support, a quantitative analysis of the underlying uncertainty has to be added. A typical analysis in finance is the tradeoff between expectation and risk of a position. Causal and retro-causal models offer a new view on uncertainty in forecasting, because for these models uncertainty is not seen as an essential part of the world. It is caused by the non-unique reconstruction of non-observable (hidden) variables in a model. [Section 10.5](#) shows how this concept of uncertainty based on hidden variables allows a quantitative analysis of uncertainty and risk which surpasses commonplace distribution assumptions. We apply this analysis to the forecast of the Dow Jones Industrial Index (DJX) which illustrates the immediate consequences for auction pricing.

10.2. Recurrent Neural Networks

10.2.1. Small Recurrent Neural Networks

Starting with the view of a market as a deterministic dynamical system with a microeconomic state transition and an output equation bridging the gap to the macroeconomic observables, we can formally assume a vector time series y_τ which is created by a dynamical system with discrete time τ using a state transition and output equation (Haykin, 1994), given as:

$$\begin{aligned} \text{state transition} \quad s_\tau &= f(s_{\tau-1}, u_\tau), \\ \text{output equation} \quad y_\tau &= g(s_\tau). \end{aligned} \tag{1}$$

The time-recurrent state transition equation $s_\tau = f(s_{\tau-1}, u_\tau)$ describes the current state s_τ by means of a function of the previous state $s_{\tau-1}$ and of the external influences u_τ . The system formulated in the state transition equation can therefore be interpreted as a partially autonomous and partially externally driven dynamic. We call this an *open dynamical system*.

The output equation, also called observer equation $y_\tau = g(s_\tau)$, uses the non-observable system state s_τ in every time step τ to calculate the output of the dynamical system y_τ . The data-driven system identification is based on the selected parameterized functions $f()$ and $g()$. We choose the parameters in $f()$ and $g()$ such that the quadratic error function [see Equation (2)] is minimized.

$$\frac{1}{T} \sum_{\tau=1}^T (y_\tau - y_\tau^d)^2 \rightarrow \min_{f,g}. \tag{2}$$

The two functions $f()$ and $g()$ are thus estimated such that the average distance between the observed data y_τ^d and the system outputs y_τ across a number of observations $\tau = 1, \dots, T$ is minimal.

So far, we have given a general description of the state transition and the output equation for open dynamical systems. Without loss of generality, we can specify the functions $f()$ and $g()$ by means of a Recurrent Neural Network (RNN) (Schäfer and Zimmermann, 2006; Zimmermann *et al.*, 2002), as:

$$\begin{aligned} \text{state transition} \quad s_\tau &= \tanh(As_{\tau-1} + Bu_\tau), \\ \text{output equation} \quad y_\tau &= Cs_\tau. \end{aligned} \tag{3}$$

Equation (3) specifies an RNN with weight matrices A , B , and C . It is designed as a nonlinear state-space model, which is able to approximate any continuous functions $f()$ and $g()$ on a compact domain (Hornik *et al.*, 1989), over a finite time horizon. We choose the hyperbolic tangent \tanh as the transfer function for the state transitions $s_{\tau-1} \rightarrow s_\tau$. The output equation is specified as a linear function. The RNN output is generated by a

superposition of two components: (i) the autonomous dynamics (coded in matrix A), which accumulates information in the state vectors over time (memory), and (ii) the influence of external factors (coded in matrix B).

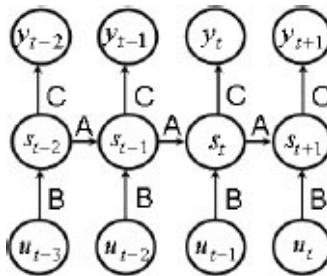


Figure 10.2: Basic RNN unfolded in time with shared weights A , B , and C .

Without loss of generality, we can assume a linear output equation in Equation (3). Any nonlinearity in the output equation could be merged with the state transition equation. For details see Schäfer and Zimmermann (2006).

The temporal system identification implies that we have to determine appropriate matrices A , B , and C , which minimize the error function Equation (2). We apply the technique of finite unfolding in time (Rumelhart *et al.*, 1986) to solve this task. The underlying idea of finite unfolding is that any RNN can be rewritten as an equivalent iterated Feedforward Neural Network, as long as the matrices A , B , and C are identical over all time steps (shared weights). Figure 10.2 depicts the resulting RNN in a simplified representation, where the transfer functions are located inside the nodes of a graph with the connections encapsulating the weight matrices.

One advantage of shared matrices is the moderate number of free parameters (weights), which reduces the risk of overfitting (Zimmermann and Neuneier, 2000). The actual training is conducted using error backpropagation through time (EBTT) together with a stochastic learning rule (Rumelhart *et al.*, 1986; Werbos, 1974). For algorithmic solutions, the reader is referred to the overview article by B. Pearlmutter (Pearlmutter, 2001).

10.2.2. Overshooting

The basic idea of the previous section defines a frame for all our modeling tasks. We can now add further architectural elements to the neural networks to code prior information in the network topology. This helps to control the training process and constrains the basic gradient descent of the backpropagation training algorithm to favor solutions which match the additional side conditions. Constraints can either be introduced from observations of past training attempts or from knowledge about the structure of the modeled dynamical system.

Predicting load curves (e.g., for electricity), a stable forecast often has to be generated over multiple time steps into the future. A problem often observed during training of RNNs for this application is that RNNs tend to focus on only the most recent external

influences in order to explain the system dynamics. This has the consequence that forecasts can be only generated for a small number of steps before the model becomes unstable. To balance the information flow and extend the stable forecast range, we use a technique called overshooting. Overshooting extends the autonomous part of the system dynamics (coded in matrix A) into the future (here $\tau + 2, \tau + 3, \tau + 4$) (Zimmermann and Neuneier, 2000) (see [Figure 10.3](#)). With the iterated application of the shared weight matrix A we can compute consistent multi-step forecasts. A corresponding RNN architecture is depicted in [Figure 10.3](#).

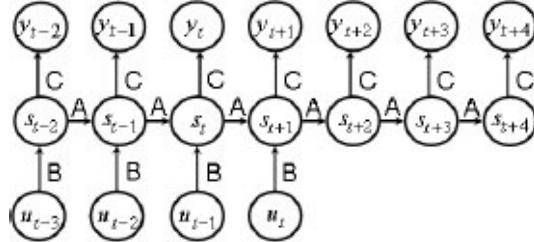


Figure 10.3: RNN incorporating overshooting.

For the RNN, we typically use an input preprocessing $u_t = x_t - x_{t-1}$ as the transformation for the raw data x . This avoids trends in the input or target variables of the RNN. The missing external inputs $u_{t>0}$ in the forecast part of the network can be interpreted as an assumption of a constant environment ($u_{t>0} \approx 0$). The effectiveness of overshooting depends on the strength of the autonomous dynamics. The stronger the autonomous flow is, the better the forecast accuracy for the overshooting time steps will be. A network with more time steps in the overshooting part will put higher emphasis on learning the autonomous part of the dynamic. It will accumulate a larger number of error gradients for the adaption of the state transition matrix A from the forecast part of the network during training. In this way the number of forecast steps implicitly determines the emphasis, which the training algorithm puts on the autonomous component of the system dynamic.

Summarizing, we can state that overshooting generates additional valuable forecast information about the identified dynamical systems and acts as a regularization method for the training.

10.3. Error Correction

Another problem, often encountered in demand forecast applications, is the incomplete observability of the market and a high noise level of the observed data. Under these conditions, system identification often fails and produces models that only have learned the input data by heart (overfitting) and do not show the desirable generalization capabilities.

To recapitulate the basic form of a dynamical system, we can use Equation (4) to identify the temporal relationships between states of a deterministic dynamical system by setting up a memory in form of the states in a state transition equation, as long as we have a complete description of all external forces influencing the system.

$$\begin{aligned}s_\tau &= f(s_{\tau-1}, u_\tau), \\ y_\tau &= g(s_\tau).\end{aligned}\tag{4}$$

If we are unable to identify the underlying system dynamics due to insufficient input information or unobserved influences, we can interpret the observed model error at a preceding time step as an indicator that our model will be misleading at a later point in time. Using this error as an additional input for the state transition, we extend Equation (4) to obtain Equation (5)

$$\begin{aligned}s_\tau &= f(s_{\tau-1}, u_\tau, y_{\tau-1} - y_{\tau-1}^d), \\ y_\tau &= g(s_\tau).\end{aligned}\tag{5}$$

Once a perfect description of the underlying dynamics has been learned, the extension of Equation (5) is no longer required, because the observed model error at $t - 1$ approaches zero. Hence, modeling the dynamical system, one could directly refer to Equation (4). In all other situations, the model uses its own error flow as an indicator for unexpected shocks. This is similar to the MA part of a linear ARIMA model [ARIMA stands for Autoregressive Integrated Moving Average model, which utilizes both linear autoregressive components and stochastically moving average-components derived from the observed model error to fit a time series, see (Wei, 1990)]. Since we are working with state space models, we do not need delayed error corrections.

Another resemblance bears to Kalman Filters, where the model error is used to improve the system identification. In contrast to the online adaptation in the Kalman approach, we try to identify a fixed nonlinear system which is able to handle external shocks. By using such a system, we cannot avoid an error when the external shock appears. Instead, our task is to adapt fast to the new situation. Such a strategy significantly decreases the rote learning of false causalities and will improve the generalization ability of our model.

10.3.1. Error Correction Neural Networks

A first neural network implementation of the error correction equations in Equation (5) can be written as:

$$\begin{aligned}s_\tau &= f(s_{\tau-1}, u_\tau, y_{\tau-1} - y_{\tau-1}^d), \\ y_\tau &= g(s_\tau).\end{aligned}\quad (6)$$

The term Cs_{t-1} recomputes the previous output y_{t-1} and compares it with the observed data y_{t-1}^d . The transformation matrix D is necessary to adjust different dimensionalities in the state transition equation. It is important to note that the identification of the above model framework is ambiguous, creating numerical instabilities. The main reason is the autoregressive structure between s_{t-1} and s_t which can either be coded in matrix A or in DC . We can disambiguate Equation (6) into the form in Equation (7) utilizing $\hat{A} = A + DC$.

$$\begin{aligned}s_\tau &= \tanh(\hat{A}s_{\tau-1} + Bu_\tau + Dy_{\tau-1}^d), \\ y_\tau &= Cs_\tau.\end{aligned}\quad (7)$$

The framework of Equation (7) is algebraically equivalent to Equation (6) and avoids numerical instabilities. Unfortunately, by using Equation (7), the explicit error information provided by external shocks is lost, since it is no longer measured as a deviation from zero. Neural networks that use tanh as the transfer function are, on the other hand, well known to be numerically most stable, when the model variables fluctuate around zero, which fits best to the finite state space $(-1; 1)^n$ spanned by the tanh nonlinearity.

Fortunately, we can rewrite Equations (6) and (7) again as shown in Equation (8) in an error correction form, measuring the difference between the expected value Cs_{t-1} and the observation y_{t-1}^d . The disambiguation is here a consequence of the additional nonlinearity.

$$\begin{aligned}s_\tau &= \tanh(As_{\tau-1} + Bu_\tau + D \tanh(Cs_{\tau-1} - y_{\tau-1}^d)), \\ y_\tau &= Cs_\tau.\end{aligned}\quad (8)$$

The system identification, Equation (9), is now a parameter optimization task adjusting the weights of the *four* matrices A, B, C, D .

$$\frac{1}{T} \cdot \sum_{\tau=1}^T (y_\tau - y_\tau^d)^2 \rightarrow \min_{A, B, C, D} . \quad (9)$$

10.3.2. Unfolding in Time for Error Correction Neural Networks

Next, we translate the formal description of Equation (8) into a network architecture named Error Correction Neural Network (ECNN).

The ECNN architecture (Figure 10.4) is best understood, if one analyzes the dependency between the elements in one unfolding segment of the network with $s_{t-1}, u_t, z_{t-1} = Cs_{t-1} - y_{t-1}^d$ and s_t .

We can identify two types of inputs to the model: The external inputs u_t directly influence the state transition $s_{t-1} \rightarrow s_t$. The second input type are the targets y_{t-p}^d , where only the difference between the internally expected output y_{t-1} and the observation y_{t-1}^d has an impact on the state calculation for s_t . Note that $-Id$ is the negative of an identity matrix which is kept constant during the network training.

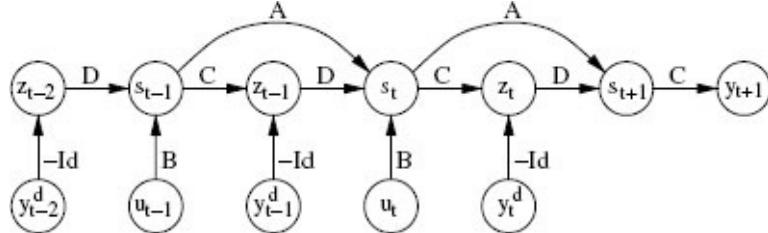


Figure 10.4: Error correction neural network.

This design allows an elegant handling of missing values in the series of target vectors: If there is no compensation of the internal expected value $y_{t-1} = Cs_{t-1}$, the system automatically generates a replacement. A special case occurs at the time $t + 1$. At this point in time, we have no compensation of the internal expected value and thus the system computes a forecast.

The outputs of the ECNN, which generate error information during the learning phase, are the $z_{t-\tau}$. Note that the target values for the outputs $z_{t-\tau}$ are zero, because we want to optimize the compensation mechanism between the expected values $y_{t-\tau}$ and their observations $y_{t-\tau}^d$.

The ECNN avoids another problem of the basic RNN networks (see [Figure 10.2](#) in [Section 10.2.1](#)): In the *finite* unfolding of a RNN, no information about a valid initial state vector s_0 is available at the beginning of the training. Usually, an initial state of zero is used to initialize the state sequence. Only later in the training, a consistent initial state value can be learned. From the initial state, the network can only gradually adapt to a valid initial state vector. This burdens the system identification process, because the network not only has to identify the dynamical system but also needs to adapt, to correct the assumed initial state vector. The ECNN on the other hand uses the error correction mechanism to compensate for the initial state misrepresentation already in the first state transition. This can stabilize and speed up the training process significantly.

10.3.3. Combining Overshooting & ECNN

A combination of the basic ECNN presented in [Section 10.3.1](#) and the overshooting technique of [Section 10.2.2](#) is shown in [Figure 10.5](#).

In addition to the advantages of overshooting described in [Section 10.2.2](#), it has a further effect on the learning of the ECNN. A forecast provided by the ECNN is in general based on the modeling of the recursive structure of a dynamical system (coded in the matrix A) and the error correction mechanism, which acts as external input (coded in C

and D). Overshooting puts emphasis on the training of the autoregressive substructure in the forecast part of the network, stabilizing it for long term forecasts. Of course, we have to supply the additional target values $y_{t+1}, y_{t+2}, y_{t+3}, \dots$, for the overshooting part of the network. Nevertheless does the number of parameters (weights) of the network not increase, compared to the standard ECNN in [Figure 10.4](#). Overshooting only adds network structure which moderates the training process to favor solutions with desirable qualities (here the stability of long term forecasts).

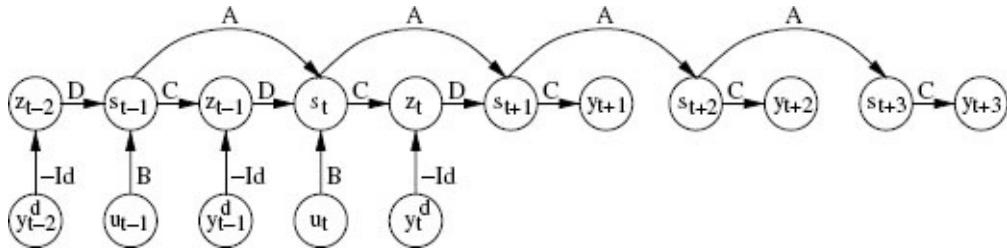


Figure 10.5: Combining overshooting and error correction neural networks.

10.3.4. Variant–Invariant Separation

One characteristic of load forecast applications is the high number of input variables. In this section we will show, how a technique called variant–invariant separation can be used to reduce the complexity of the modeling task. The basic idea is that of a coordinate transformation. We will introduce the technique using the example of a pendulum like the one shown in [Figure 10.6](#).

One naïve way to describe the pendulum dynamic could state the coordinates of all the points along the pendulum axis (an infinite number of values) and their temporal shifts. Of course, nobody would describe a pendulum in this way. Instead, the description can be simplified, if we separate it into one variant (the angle ϕ , which changes over time) and the invariants (all points of the pendulum always lie on its axis).

We learn from this exercise that an appropriate coordinate transformation can lead to tremendous simplifications in the modeling of dynamics if we separate variants and invariants. Our goal is to describe a dynamical system with a minimum number of variants.

A typical example for this technique is the forecasting of yield curves in economy, where we can compress ten or more maturities of interest rates to a small number of variants (for the German economy, the standard yield curves can be reduced to only three variants). From the forecasts of only these variants we can predict the complete yield curve. Another example is tactical portfolio management. Instead of forecasting every individual stock, we can compress the task by e.g., only predicting market sectors.

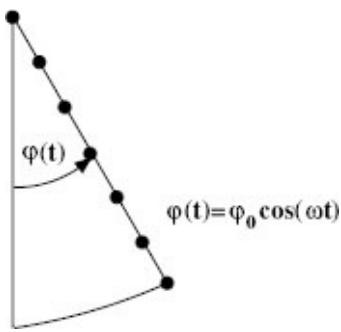


Figure 10.6: The dynamic of a pendulum can be separated in only one variant, the angle ϕ , and an infinite number of invariants.

10.3.5. Variant–Invariant Separation with Neural Networks

The translation of the variant–invariant separation to a neural network architecture is shown in [Figure 10.7](#).

In this network, the internal state variables s_t, s_{t+1} play the role of the variants, which develop over time with the help of the forecast transition function $f()$. The variants are separated from the invariants by the compressor function $g()$. The decompressor function $h()$ combines the forecast of the variants with the invariants to reconstruct the complete dynamic.

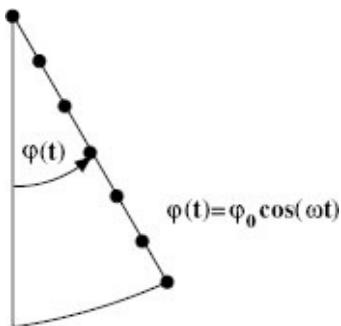


Figure 10.7: Variant–invariant separation by neural networks.

Although the general description of the variant–invariant separation can easily be stated without referring to neural networks at all, the realization is a typical neural network approach in the way that it uses concatenations of functional systems. Training the subsystems of compression/decompression and forecast is then adequately implemented using the EBTT algorithm.

10.3.6. Combining Variant–Invariant Separation with ECNN

In this section, we merge the concept of dimension reduction with the ECNN architecture. One might expect that merging both architectures leads to a very large interconnected structure. In [Figure 10.8](#), we show a different approach. The compressor-decompressor network (also called *bottleneck network*) on the left-hand side is only seemingly disconnected from the ECNN network on the right-hand side. The weight matrices E and F are shared between both networks, combining the compression/decompression functionality of the bottleneck network with the forecast functionality of the ECNN.

Training both networks together combines the search for an optimal coordinate transformation with the system identification.

In addition to the ECNN presented in [Section 10.3.3](#), we introduced two new matrices E and F which parameterize compression and decompression respectively. The ECNN calculates the state transitions of the dynamical system on a state vector which has been compressed to the minimal dimension of vector x_t instead of the high dimensional vector y_t . It is important to note that the ECNN requires $-y_t^d$ as inputs in order to generate $-x_t^d$ in the z_t layer. In this way, the internal forecasts $x_t = Cs_{t-1}$ can be compensated by the transformed target data $-x_t^d = F(-y_t^d)$.

Applying this architecture to a number of real world modeling tasks, we found that the training of the combined neural network ([Figure 10.8](#)) is very robust, i.e., the coordinate transformation and the forecast can be trained simultaneously. For models with very high dimensionality it is also feasible, to first train the bottleneck network, then freeze the compressor and decompressor weights, train the ECNN and finally train both components together for a final optimization. Furthermore, node pruning algorithms are applicable to the x_t layer to support the dimension reduction.

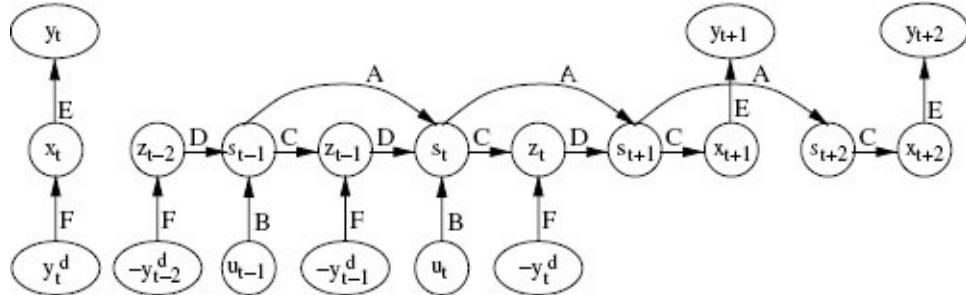


Figure 10.8: Combining variant–invariant separation and forecasting.

10.4. High Dimensional Dynamical Systems

10.4.1. Historically Consistent Neural Networks (HCNN)

Many real-world technical and economic applications can be seen in the context of large systems in which various (nonlinear) dynamics interact over time. Unfortunately, only a small subset of variables can be observed. From the subset of observed variables and their dynamics we have to reconstruct the hidden variables of the large system in order to understand the overall system dynamics. Here the term *observables* includes the input and output variables of conventional modeling approaches (i.e., $y_\tau := (y_\tau, u_\tau)$).

With the concept of observables, we can identify a consistency problem in the overshooting extensions of the RNN ([Figure 10.3](#)) and ECNN ([Figure 10.5](#)) architectures. On the output side, these networks provide forecasts for the observables y_τ , whereas they assume on the input side that the same observables y_τ can be kept constant and their influence on the (autonomous) dynamic can be neglected. This contradiction can be remedied, if we give up the distinction between input and output variables for a network but only regard observables. They describe the external state of the dynamical system for which we can compare measured observables with our forecast for the same observables. In this way, we can close the dynamical system.

The consequence of this concept is an architecture which we call HCNN. This model class follows our design principles for the modeling of large dynamical systems and overcomes the conceptual weakness described above. Equation (10) formulates the HCNN.

$$\begin{aligned} \text{state transition} \quad s_\tau &= A \tanh(s_{\tau-1}), \\ \text{output equation} \quad y_\tau &= [Id, 0]s_\tau. \end{aligned} \tag{10}$$

The joint dynamic for all observables is characterized in the HCNN by the sequence of states s_τ . For convenience, the observables ($i = 1, \dots, N$) are arranged on the first elements of the state vector s_τ , followed by the non-observable (hidden) variables. The matrix $[Id, 0]$ is constant and extracts the observables from the state vector. The initial state s_0 is described as a bias vector. The bias s_0 and the matrix A are the only free parameters of the HCNN.

Like standard RNNs ([Figure 10.2](#)), HCNNs are also universal approximators. The proof for the RNN can be found in Schäfer and Zimmermann (2006). [Figure 10.9](#) outlines the proof for HCNNs. It can be divided into six steps:

1. The output equation is shifted one-time step into the future and the resulting $s_{\tau+1}$ is substituted by the state transition equation.
2. By combining outputs and state variables into an extended state we derive an extended state transition equation. The output of the system is derived from the first components

of the extended internal state.

3. For the extended state transition equation, we apply the feedforward universal approximation theorem. At least for a finite time horizon this guarantees a small approximation error. Note that in RNNs at least one large component of the state vector together with the tanh transfer function can mimic a bias vector. Thus, we have omitted the explicit notation of a bias vector in the equations.
4. In this step, we remove one of the two matrices within the state transition equation. We apply a state transformation $r_\tau = As_\tau$. This results in two state transition equations.
5. The two-state transition equations can be reorganized into one-state transition, which combines the linear parts of the equations in one matrix.
6. Rewriting this matrix results in the claimed formulation for closed systems.

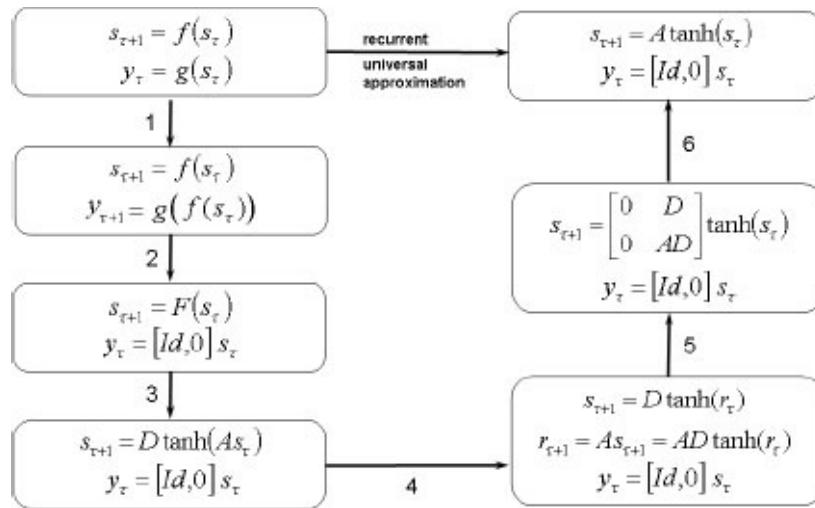


Figure 10.9: Proof of the universal approximation capabilities for HCNNs.

In contrast to the RNN and ECNN formulations in Equations (3) and (8), we shift in Equation (10) the application of matrix A in front of the tanh transfer function. This has the advantage that the system states — and thus also the system outputs — are not limited to the finite state space $(-1; 1)^n$, which is usually created by the tanh nonlinearity. The output equation has a simple and application independent form. Note that only the leading elements of the state vector are observable.

[Figure 10.10](#) depicts the HCNN architecture. In this network architecture, the HCNN states s_τ are realized as hidden layers with identity transfer functions. The intermediate states r_τ use the usual tanh squashing. The forecasts are supplied by the output layers y_τ . There are no target values available for the future time steps. The forecast values $y_{\tau>t}$ can be read out at the corresponding future time steps of the network.

Since the HCNN model has no inputs, we have to unfold the neural network along the complete data history. This is different from small recurrent neural networks (see e.g., [Figure 10.3](#)), where we construct training data patterns in form of sliding windows. The HCNN learns the large dynamic from a single history of observations (i.e., a single

training data pattern). For our commodity price forecasts, we unfold the HCNN over 440 trading days in the past to predict the next 20 days.

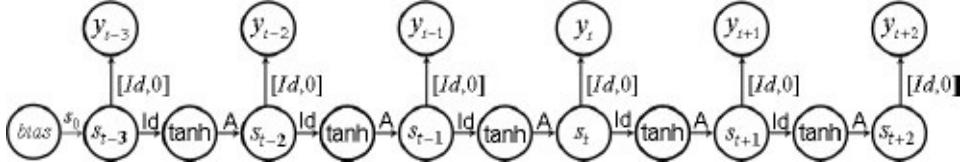


Figure 10.10: HCNN.

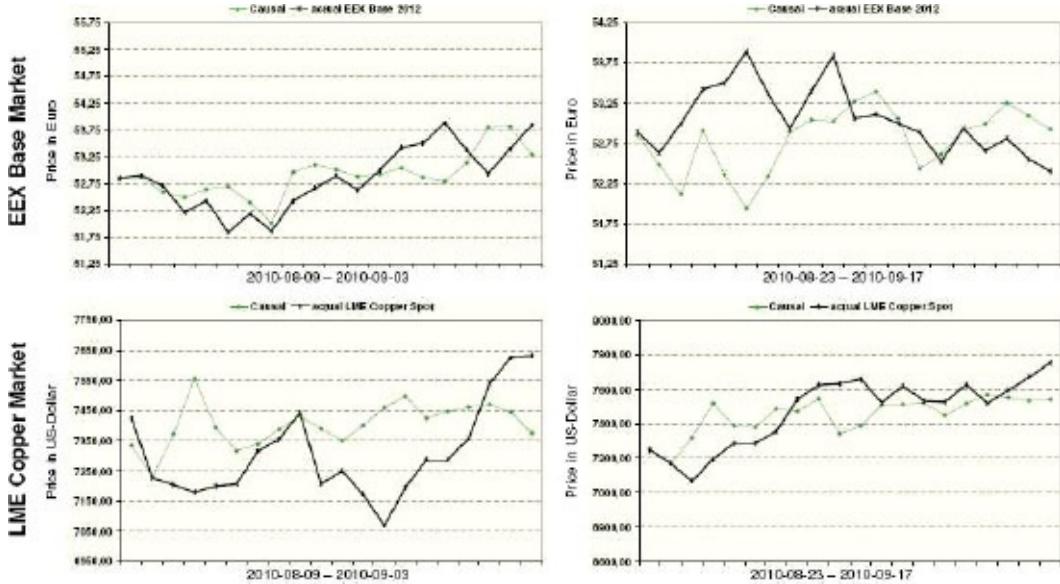


Figure 10.11: Forecasting EEX and LME future contracts over 20 days with HCNNs.

We applied the HCNN to forecast the market price development over the next 20 trading days of electricity and copper future contracts traded on the European Energy Exchange (EEX) and the London Metal Exchange (LME) respectively. Figure 10.11 compares the HCNN forecasts with the actual future prices for two forecast horizons. The HCNN model produces good forecasts for the first forecast horizon of the EEX market and the second horizon of the LME market. It fails in the other cases. We will analyze these failures in depth in Section 10.4.4.

10.4.2. Architectural Teacher Forcing (ATF) for HCNNs

In practice, we observe that HCNNs are difficult to train since the models do not use any input signals and are unfolded across the complete dataset. ATF makes the best possible use of the data from the observables and accelerates the training of the HCNN (Pearlmutter, 2001; Williams and Zipser, 1989; Zimmermann *et al.*, 2012). It was originally introduced as an extension to the algorithmic training procedures of RNNs. Here we integrate the idea into the network architecture, so we can continue to use the unmodified EBTT training algorithm (Haykin, 1994). The HCNN with integrated teacher forcing is shown in Figure 10.12.

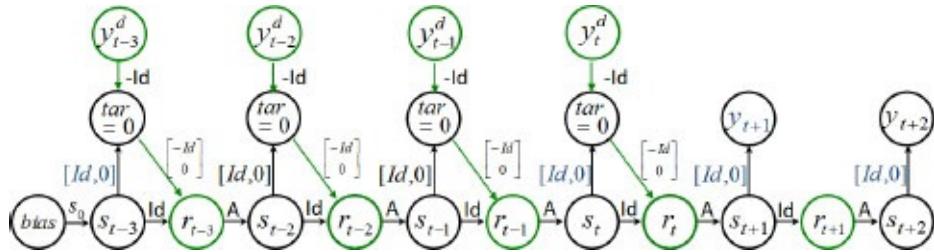


Figure 10.12: HCNN with ATF.

In this network, the calculated values for all observables up to time $\tau = t$ are replaced with the actual observations in the next unfolding step. For this, we adapt the network architecture: As a first change, the output layers of the HCNN are given fixed zero target values. The negative observed values $-y_t^d$ for the observables are then added to the output, forcing the HCNN to produce the expected values y_t to compensate.

As we constructed the HCNN in its plain form, the intermediate state vectors r_τ receive a copy of the state vectors s_τ which contain in their first elements the network output for the observables. If we now feed back the difference between the network outputs and the expected values to the intermediate state vectors, we can correct the calculated values to the observed values as shown in [Figure 10.12](#), providing the next unfolding step with the correct values

$$y_\tau^d = y_\tau - (y_\tau - y_\tau^d). \quad (11)$$

For later reference let us call Equation (11), the teacher forcing identity. All connection matrices of the teacher forcing mechanism are kept constant. Only the transition matrix A is adapted to transform the corrected intermediate state to the state of the next unfolding step.

By definition, we cannot provide observables for the future time steps. Here, the system is iterated exclusively on the expected values. This turns an open dynamical system into a closed one. The HCNN architecture in [Figure 10.12](#) is equivalent to the architecture in [Figure 10.10](#), if it has converged to zero error in the training. In this case we have found a solution for the original problem.

10.4.3. Sparsity, Dimensionality versus Connectivity and Memory

HCNNs may have to model many observables in parallel over time depending on the modeled system. The result is high dimensional dynamical systems (e.g., in our commodity price models we use $\text{dim}(s) = 300$) which are unfolded over many time steps. Backpropagation training with a fully connected state transition matrix A of such dimensions is numerically unstable: In the forward pass the nonlinear squashing function of the network reliably prevents the uncontrolled growth of propagated values. Nevertheless in the backward pass, the derivative of the squashing function may fail to prevent the uncontrolled growth of the propagated error values leading to unstable training.

One way to avoid this problem is to use a sparse matrix A .

We have to answer the question, which dimensionality and which sparsity to choose. In Zimmermann *et al.* (2006), we showed that dimensionality and sparsity are related to another pair of meta-parameters: connectivity (con) and memory length (mem). Connectivity is defined as the number of nonzero elements in each row of matrix A . The memory length is the number of unfolding steps building the memory window from which we have to collect information in order to reach a Markovian state, i.e., the state vector could accumulate all required information from the past. We propose the following parameterization for the state dimension $\text{dim}(s)$ and sparsity (Zimmermann *et al.*, 2006):

$$\text{Dimension of } A: \text{dim}(s) = \text{con} \cdot \text{mem}, \quad (12)$$

$$\text{Sparsity of } A: \text{sparsity} = \text{rnd} \left(\frac{\text{con}}{\text{mem} \cdot \text{con}} \right) = \text{rnd} \left(\frac{1}{\text{mem}} \right). \quad (13)$$

Equation (13) represents the insight that a sparse system conserves information over more time steps before the information diffuses in the network. As an artificial example, compare a shift register with a fully connected matrix. The shift register requires only one connection from one time step to the next and acts as memory whereas a fully connected matrix masks the information passed from one time step to the next with the superposition of information.

Let us assume that we have initialized the state transition matrix as a sparse matrix A with uniform random sparsity. Following Equation (13), the denser parts of A can model faster sub-dynamics within the overall dynamic, while the highly sparse parts of A can focus on slow subsystems which require more memory. The sparse random initialization allows the combined modeling of systems on different time scales.

Unfortunately, Equation (12) favors very large dimensions. Our earlier work [see (Zimmermann *et al.*, 2006)] started with the setting of the system memory length mem, because for RNNs the memory length is equal to the number of the past unfolding steps. On the other hand, connectivity has to be chosen higher than the number of the observables. Working with HCNNs, the memory length becomes less important because we unfold the neural network along the complete data horizon. Here, the connectivity plays the major role. Empirical tests provided the rule of thumb that the EBTT algorithm remains stable with a connectivity less than or equal to 50 ($\text{con} \leq 50$). To balance the computational performance, we usually limit the state dimensionality to $\text{dim}(s) = 300$. This implies a sparsity of $50/300 \approx 17\%$. We leave the fine tuning of the parameters to the EBTT learning algorithm.

Matrix A is initialized with a randomly chosen sparsity grid. It is not optimized by e.g., pruning algorithms. We counter possible initialization biases towards inferior learned models with ensemble forecasts which average individual solutions learned with different

random initializations.

We conducted ensemble experiments with different sparsity grids versus ensembles based on a constant sparsity grid. We found that the average of the ensemble as well as the ensemble spread are unaffected by the initialization of the sparsity grid (for more details on ensemble forecasting, see [Section 10.5](#)). It should be noted that these considerations hold only for large systems.

10.4.4. Retro-Causal Historical Consistent Neural Networks

A market dynamic can often be explained by a causal analysis of the dynamical system, i.e., we explain the system behavior at present time exclusively with a superposition of the historical development. Unfortunately, a causal model might lack in performance when the dynamic of interest results from rational decision-making and planning, where the anticipation of the future together with a superordinated goal (e.g., profit or utility maximization) plays an important role. Rational planning implies an information flow from an expected future state of the system backwards in order to make an optimal decision at present time given a specific target resp. reward function. Therefore, we have to formulate a model in which the information flows from the future backwards to the past. Inspired by the standard (causal) HCNN (see [Figure 10.10](#)), we define a RCNN by

$$\text{state transition } s_\tau = \tanh(A's_{\tau+1}), \quad (14)$$

$$\text{output equation } y_\tau = [Id, 0]s_\tau, \quad (15)$$

where the internal state s_τ depends on the future state $s_{\tau+1}$ and the observables y_τ are derived from the internal state s_τ . In a causal HCNN the dynamic evolves in time from an initial state s_0 as a cause and effect model coded in the state transition equation. In a retro-causal HCNN, the dynamic is described by a final future state s_T together with a retro-causal state transition equation for effect and cause modeling. The retro-causal state transition is coded in matrix A' . Opposite to causal HCNNs, retro-causal HCNNs use error information up to the end of the future unfolding and forecasting is based on the estimation of the final state s_T .

Given the equations of the retro-causal HCNN [Equation (10)], we sketch the architecture of the RCNN in [Figure 10.13](#). Likewise to the causal HCNN, the RCNN is trained using the EBTT algorithm and the architectural realization of teacher forcing (Haykin, 1994; Pearlmatter, 2001; Williams and Zipser, 1989).

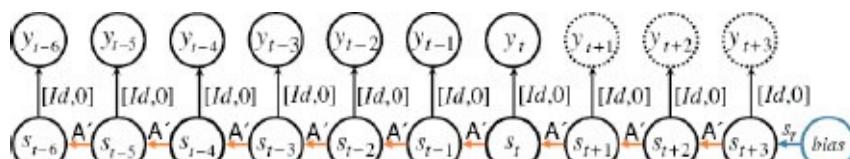


Figure 10.13: Architecture of the RCNN.

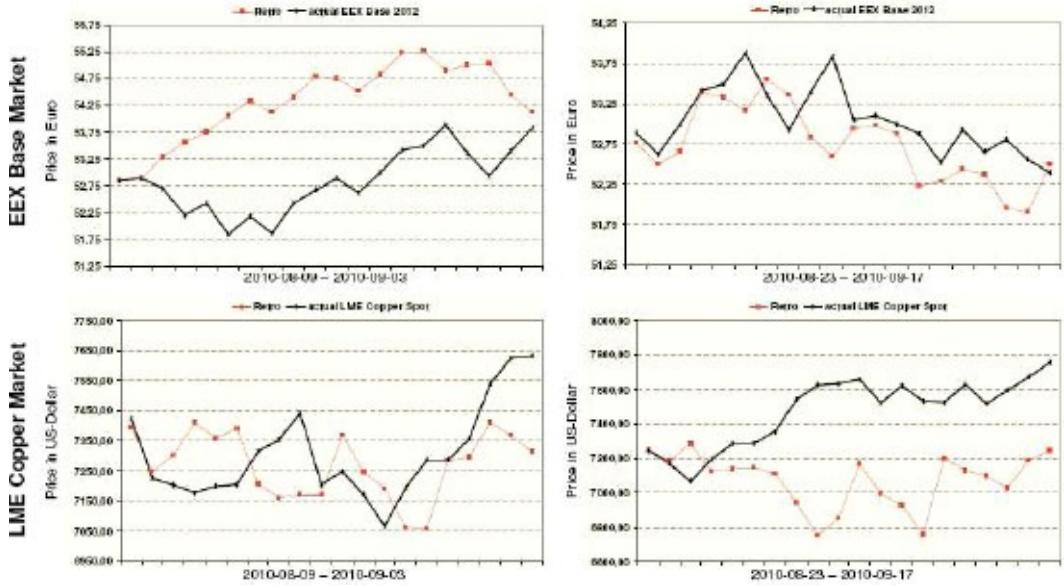


Figure 10.14: Forecasting EEX and LME future contracts over 20 days with retro-causal HCNNs (RCNNs).

To study the forecast accuracy for the EEX and LME future contracts, we applied RCNN to the selected forecast horizons. The results are depicted in Figure 10.14.

We can now see that in the cases where the causal HCNN fails, the RCNN is able to provide an accurate forecast of the market dynamic (see Figures 10.11 and 10.14). However, we do not know *ex ante* whether the market is under a causal or retro-causal regime. Next we want to combine both modeling approaches in an integrated model to handle both market regimes properly.

10.4.5. Causal-Retro-Causal Historical Consistent Neural Networks (CRCNN)

CRCNN combine a causal and a retro-causal information flow in an integrated neural model given by

$$\text{state transition } s_\tau = \tanh([As_{\tau-1}, A's'_{\tau+1}]), \quad (16)$$

$$\text{output equation } y_\tau = [Id, 0]s_\tau + [Id, 0]s'_\tau. \quad (17)$$

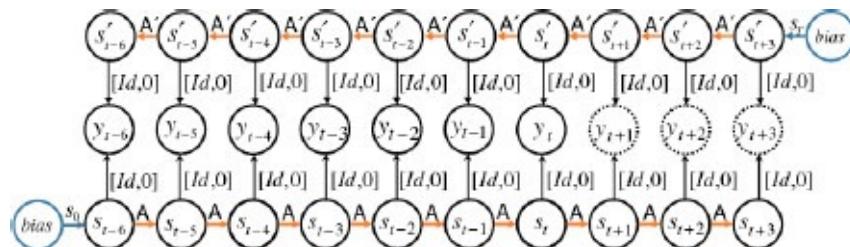


Figure 10.15: Architecture of the CRCNN.

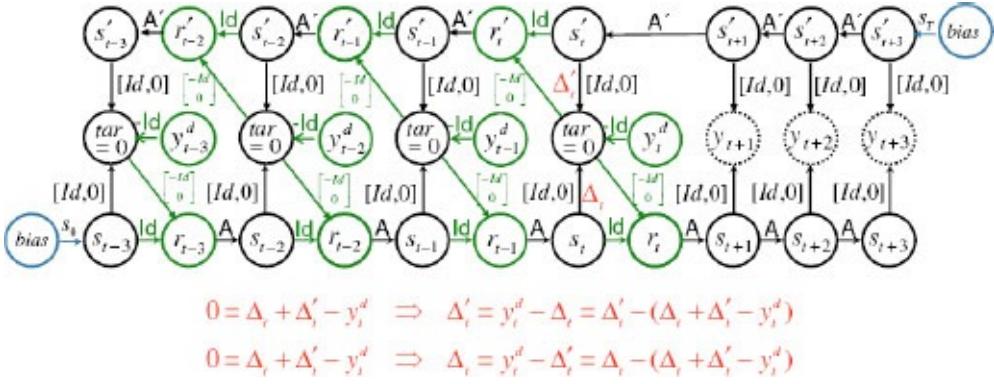


Figure 10.16: Extended CRCNN incorporating a Teacher Forcing (TF) mechanism.

The state transition equation s_t of the CRCNN, Equation (16) is the sum of causal and retro-causal influences. The dynamic of all observables is explained by a sequence of causal states s_t using the transition matrix A to explain the causal information flow and of retro-causal states s'_t using the transition matrix A' for the retro-causal information flow. The output equation [Equation (16)], computes the observables y_t by adding the causal and retro-causal information. Based on Equations (16) and (17), we can formulate the unfolded network architecture for the CRCNN shown in Figure 10.15.

Like for the basic HCNN, the CRCNN is unfolded over the entire time interval of available data, i.e., we learn the unique complete history of the system. As we did for the basic HCNN in Section 10.4.2, we also add architectural teacher forcing as shown in Figure 10.16.

The fundamental idea of the CRCNN architecture is that for the causal and the retro-causal part of the network, the error of one part is compensated by the opposite part. This has consequences for the teacher forcing mechanism, as the state correction for one part of the network should only correct the portion of the difference between calculated and observed values which is not explained by the opposite part of the network. If we denote the output contributions of the two network parts with $[Id, 0]s_t = \delta_t$ and $[Id, 0]s'_t = \delta'_t$, we can rewrite the teacher forcing identity from Equation (11) for the CRCNN is

$$\delta_t^d = \delta_t - (\delta_t + \delta'_t - y_t^d), \quad (18)$$

$$\delta'^d_t = \delta'_t - (\delta_t + \delta'_t - y_t^d). \quad (19)$$

It is important to note that over the training of the network, the ratio of δ_t to δ'_t will adapt to the market regime, so in contrast to the simple case of the HCNN, the teacher forcing of the CRCNN cannot produce the definite correction values but it still provides best estimates of the correction based on the current contribution of the network parts.

The CRCNN (as well as the HCNN and the RCNN) models the dynamics of all observables and hidden variables in parallel. Thus, high-dimensional state transition matrices A and A' are required. For numerical stability, we again recommend to initialize these matrices with a (random) sparse connectivity. The degree of sparsity is chosen with

respect to the metaparameters connectivity and memory length [see Equation (13) in [Section 10.4.3](#)].

Our CRCNN architecture relies on the assumption that the network can learn the balance between the causal and the retro-causal parts of the network from the presented data. This has the consequence that for all possible extensions of our CRCNN architecture, extreme care has to be applied, not to introduce asymmetries in the network architecture. Any asymmetry in the network would favor one of the network parts and introduce a serious modeling bias. Even in a perfect symmetric architecture, the modeling of the market regime may fail if the wrong part of the network arrives faster at a local minimum than the part which later could model the market regime better. Training is not guaranteed to provide the best possible model. Our solution for this problem is to train multiple models with random initializations and view the complete trained model ensemble as the modeling result. We will elaborate on this idea in the following section.

[Figure 10.17](#) depicts the impact of our modeling efforts to predict the EEX and LME future contracts for the selected time periods.

The combination of the causal and retro-causal information flow within an integrated CRCNN is able to provide an accurate prediction for all future contracts under consideration. The trained CRCNN is able to detect beforehand if the market is governed by a causal or retro-causal regime as well as a mixture thereof, since the causal and retro-causal information flows are dynamically combined during the training of the CRCNN.

10.5. Uncertainty and Risk

The experience gained during the latest financial crisis has triggered a far-reaching discussion on the limitations of quantitative forecasting models and made investors very conscious of risk (Föllmer, 2009). In order to understand risk distributions, traditional risk management uses diffusion models. Risk is understood as a random walk, in which the diffusion process is calibrated by the observed past error of the underlying model (McNeil *et al.*, 2005). In contrast, this section focuses on ensemble forecasts in order to provide important insights into complex risk relationships. These insights stem from the reconstruction of internal (unobserved) model variables which complement the observable description of the modeled system.

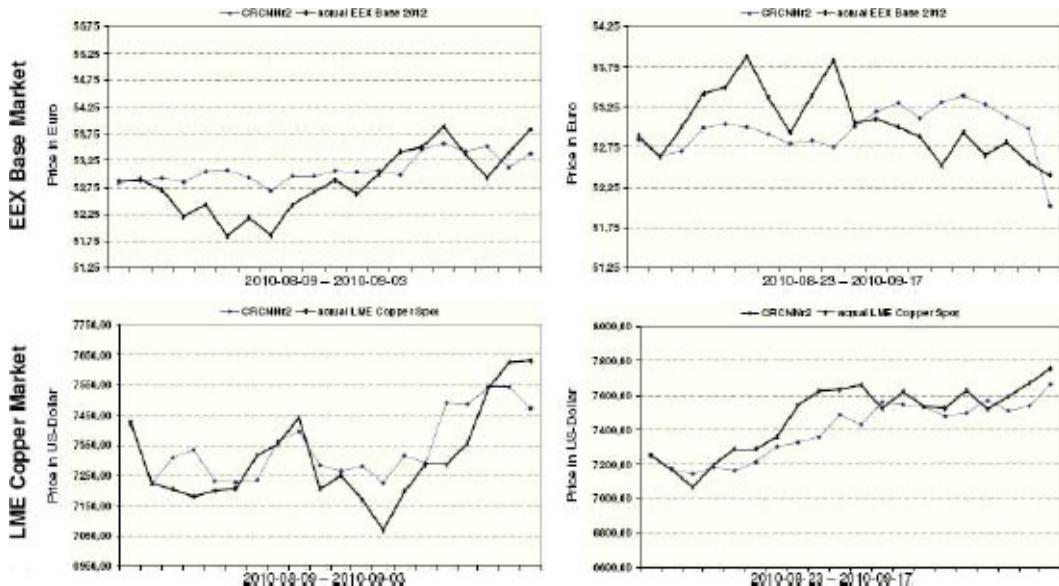


Figure 10.17: Forecasting EEX and LME future contracts over 20 days with CRCNN.

If the system identification is calculated repeatedly for HCNNs or CRCNNs with varying start conditions, an ensemble of solutions will be produced, which all have been trained to a forecast error of zero on the training data, but produce different predictions for the future. Since every HCNN or CRCNN model can produce a perfect description of the observed data, the complete ensemble is the true solution. A way to simplify the forecast is to take the arithmetical average of the individual ensemble members as the expected value, provided that the ensemble histogram is unimodal in every time step.

In addition to the expected forecast value, we consider the bandwidth of the ensemble, i.e., its distribution. The form of the ensemble is governed by differences in the reconstruction of the hidden system variables from the observables: for every finite number of observations there is an infinite number of explanation models which describe the data perfectly, but differ in their forecasts, since the observations make it possible to reconstruct the hidden variables in different forms during the training. In other words, our risk concept is based on the partial observability of the world, leading to different reconstructions of the hidden variables and thus, different future scenarios. Since all scenarios are perfectly consistent with the history, we do not know which of the scenarios

describes the future trend best and risk emerges.

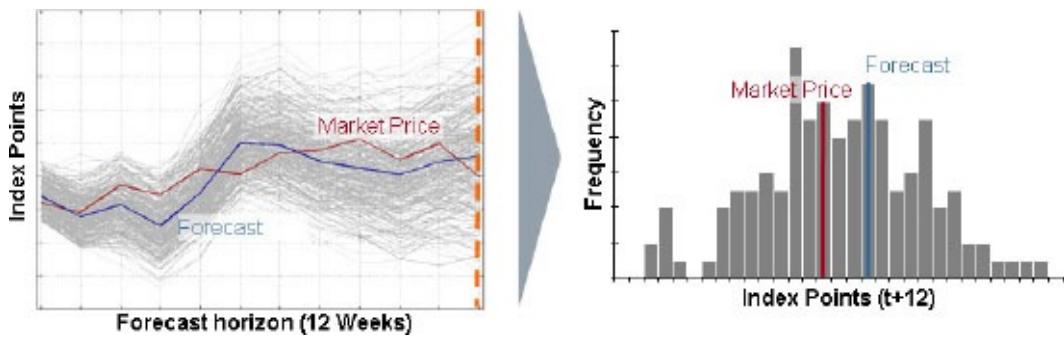


Figure 10.18: HCNN ensemble forecast for the DJX (12 weeks forecast horizon), left, and associated index point distribution for the ensemble in time step $t + 12$, right.

This approach directly addresses the model risk. For HCNN and CRCNN models, we claim that the model risk is equal to the forecast risk. The reasons can be summarized as follows: First, HCNNs are universal approximators which are therefore able to describe every market scenario. Second, the form of the ensemble distribution is caused by an underlying dynamic, which interprets the market dynamics as the result of interacting decisions (Zimmermann, 1994). Third, in experiments we have shown that the ensemble distribution is independent of the model configuration details, if we build large models and large ensembles.

Let us exemplify our risk concept: The diagram (Figure 10.18, left) shows our approach applied to the Dow Jones Industrial Index (DJX). For the ensemble, a HCNN was used to generate 250 individual forecasts for the DJX.

For every forecast date, all individual forecasts of the ensemble members together define the empirical density function, i.e., a probability distribution over many possible market prices at a single point in time [see Figure 10.18, right]. It is noticeable that the actual development of the DJX is found always within the ensemble channel [see gray lines, Figure 10.18, left]. The expected value for the forecast distribution is also an adequate point forecast for the DJX [see Figure 10.18, right]. For further discussion of ensembles and their visual analysis with heat maps, see also Köpp *et al.* (2014).

10.6. Conclusion

At the beginning of this chapter we claimed that the mathematics of neural networks allows unique insights into market analysis. With the modeling techniques presented in the previous sections we are now in the position to review this claim.

First of all we have shown that we can model economy as dynamical systems. The universal approximation property makes our neural networks a strong tool which is not limited by hidden *a priori* structures in the models as, e.g., linear models would be.

Our strongest model building tool is a collection of neural network architectures which lets us add *a priori* information about dynamical systems to our networks without demanding specialized training algorithms for each new problem. Let us review which architectural designs make neural networks well suited for the challenges of the three economic systems we covered:

Business Economics: One application here is demand forecasting. This problem is characterized by a small number of variables modeled as a state space model which has to cope with incomplete knowledge about external variables—the competitors will not reveal their decision process. Our network architecture for this problem is the ECNN ([Figure 10.5](#)). Its error correction mechanism handles unknown influences and shocks in the data.

Our second application is load forecasting. Here, a large number of observations is typically available and we use variant–invariant separation in bottleneck networks to reduce the dimensionality of the problem. ECNN networks can then be applied to the low dimensional variant state space ([Figure 10.8](#)). The inverse of the dimension reduction transforms the low dimensional forecasts back to the high dimensional external description.

Macro-Economics: To forecast commodity prices, we have to model worldwide dynamical systems. The decisions of a large number of actors superpose, which we model with state transitions in recurrent neural networks. The resulting price is an observable macro-economic variable.

We cannot hope to reduce the dimensionality of this problem and neither is it suitable to introduce external influences in the past which we would have to assume as constant beyond the present. Our HCNN architecture with teacher forcing ([Figure 10.12](#)) solves this problem initially for causal market regimes. The governing idea is, to close the modeled dynamical system and regard all input and output variables just as observables which we have to explain in our model.

Unfortunately, many real markets are not only driven by a purely causal information flow. Expectation and motivation driven markets demand an inversed, retro-causal flow. Additionally, markets can change their regime dynamically which calls for combined causal-retro-causal models which we realized in the form of the CRCNN ([Figure 10.16](#)).

Teacher forcing can be adapted to these models which completes our architectural toolbox.

To choose the right architecture for an economic modeling problem means to understand the basic mechanisms which characterize the modeled market. Each of our architectural variants emphasizes individual market dynamics we have observed and which helped us to gain a deeper understanding of our modeling tasks. Where we found elements missing from our toolbox we designed new architectures and analyzed their behavior.

Finance: The challenge in finance is to balance risk and expected return. For this, a deeper understanding of risk has to be acquired. The standard view on risk begins with the assumption that risk and uncertainty are external elements which behave like a random walk and have to be added to the model and its forecast.

In our approach, uncertainty is a consequence of the non-unique reconstruction of hidden (non-observable) variables in our large dynamical models. With a finite history of data we can train an ensemble of different models which all fit the data and are therefore valid models which differ in the reconstruction of the hidden variables and of course in their forecasts. We claim that uncertainty is not the uncertainty of building the one correct model but the distribution of the forecasts from our model ensembles. We argue that the universal approximation property of our models allows us to catch any complexity of the world on a finite time horizon. We can also show empirically that for large models the distribution of model forecasts is invariant beyond a minimum ensemble size and over many model architectures.

In our projects, we have studied several thousand ensemble distributions so far. Understanding more about their microstructure is an ongoing topic of our research.

References

- Casdagli, M., Eubank, S., Farmer, J. D. and Gibson, J. (1991). State space reconstruction in the presence of noise. *Physica D*, 51, pp. 52–98.
- Elton, E. J., Gruber, M. J., Brown, St. J. and Goetzmann, W. N. (2007). *Modern Portfolio Theory and Investment Analysis*, Seventh Edition. New York: John Wiley & Sons.
- Elman, J. L. (1990). Finding structure in time. *Cognit. Sci.*, 14, pp. 179–211.
- Föllmer, H. (2009). Alles richtig und trotzdem falsch?, Anmerkungen zur finanzkrise und finanzmathematik. MDMV 17/2009, pp. 148–154.
- Gibson, J. F., Farmer, D. J., Casdagli, M. and Eubank, S. (1992). An analytic approach to practical state space reconstruction. *Physica D*, 57.
- Haykin, S. (1994). *Neural Networks. A Comprehensive Foundation*. New York: Macmillan College Publishing.
- Haykin, S. (2008). *Neural Networks and Learning Machines, Third Edition*. Upper Saddle River, NJ: Prentice Hall.
- Hornik, K., Stinchcombe, M. and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2, pp. 359–366.
- Köpp, C., von Mettenheim, H.-J. and Breitner, M. H. (2014). Decision analytics mit heatmap-visualisierung von mehrschrittigen ensembleddaten. *Wirtschaftsinformatik*, 56(3), pp. 147–157.
- Medsker, L. R. and Jain, L. C. (1999). *Recurrent Neural Networks: Design and Application*. CRC Press Int. Series Comput. Intell., No. I.
- McNeil, A., Frey, R. and Embrechts, P. (2005). *Quantitative Risk Management: Concepts, Techniques and Tools*. Princeton, New Jersey: Princeton University Press.
- Neuneier, R. and Zimmermann, H. G. (1998). How to train neural networks. In *Neural Networks: Tricks of the Trade*. Berlin: Springer-Verlag, pp. 373–423.
- Ott, E., Sauer, T. and Yorke, J. A. (eds.) (1994). *Coping with Chaos*. New York: Wiley.
- Packard, N., Crutchfield, N. H., Farmer, J. D. and Shaw, R. S. (1980). Geometry from a time series. *Phys. Rev. Lett.*, 45, pp. 712–716.
- Pearlmutter, B. (1995). Gradient calculations for dynamic recurrent neural networks: A survey. In *IEEE Trans. Neural Netw.*, 6(5), pp. 1212–1228.
- Pearlmutter, B. (2001). Gradient calculations for dynamic recurrent neural networks. In *A Field Guide to Dynamical Recurrent Networks*. New York, USA: IEEE Press, pp. 179–206.
- Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E. and McClelland, J. L. (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume I: Foundations*. Cambridge, MA: MIT Press/B Bradford Books.
- Sauer, T., Yorke, J. A. and Casdagli, M. (1991). Embedology. *J. Stat. Phys.*, 65(3/4), pp. 579–616.
- Schäfer, A. M. and Zimmermann, H.-G. (2006). Recurrent neural networks are universal approximators. *ICANN*, 1, pp. 632–640.
- Takens, F. (1981). Detecting strange attractors in turbulence. In Rand, D. A. and Young, L. S. (eds.), *Dynamical Systems and Turbulence*, 898, *Lect. Notes Math.* Springer, pp. 366–381.
- Wei, W. W. S. (1990). *Time Series Analysis: Univariate and Multivariate Methods*. New York: Addison-Wesley Publishing Company.
- Werbos, P. J. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD Thesis. Cambridge, MA: Harvard University.
- Werbos, P. J. (1994). *The Roots of Backpropagation. From Ordered Derivatives to Neural Networks and Political Forecasting*. NY: Wiley.
- Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Comput.*, 1(2), pp. 270–280.

- Zimmermann, H. G., Grothmann, R. and Neuneier, R. (2002). Modeling of dynamical systems by error correction neural networks. In Soofi, A. and Cao, L. (eds.), *Modeling and Forecasting Financial Data, Techniques of Nonlinear Dynamics*. Dordrecht, Netherlands: Kluwer.
- Zimmermann, H. G., Grothmann, R., Schäfer, A. M. and Tietz, Ch. (2006). Modeling large dynamical systems with dynamical consistent neural networks. In Haykin, S., Principe, J. C., Sejnowski, T. J. and McWhirter, J. (eds.), *New Directions in Statistical Signal Processing: From systems to brain*. Cambridge, MA: MIT Press.
- Zimmermann, H. G., Grothmann R., Tietz Ch. and v. Jouanne-Diedrich, H. (2011). Market modeling, forecasting and risk analysis with historical consistent neural networks. In Hu, B. (ed.), *Oper. Res. Proc. 2010, Select. Papers Annu. Int. Conf. German OR Soc. (GOR)*. Munich, Sept 2010. Berlin and Heidelberg: Springer.
- Zimmermann, H. G., Grothmann, R. and Tietz, C. (2012). Forecasting market prices with causal-retro-causal neural networks. In Klatte, D., Lethi, H.-J. and Schmedders, K. (eds.), *Oper. Res. Proc. 2011, Select. Papers Int. Conf. Oper. Res. 2011 (OR 2011)*. Zurich, Switzerland: Springer.
- Zimmermann, H. G. (1994). Neuronale Netze als Entscheidungskalkül. In Rehkugler, H. and Zimmermann, H. G. (eds.), *Neuronale Netze in der Ökonomie, Grundlagen und wissenschaftliche Anwendungen*. Munich: Vahlen.
- Zimmermann, H. G. and Neuneier, R. (2000). Neural network architectures for the modeling of dynamical systems. In Kolen, J. F. and Kremer, S. (eds.), *A Field Guide to Dynamical Recurrent Networks*. New York, USA: IEEE Press.

Chapter 11

Evolving Connectionist Systems for Adaptive Learning and Pattern Recognition: From Neuro-Fuzzy-, to Spiking- and Neurogenetic

Nikola Kasabov

This chapter follows the development of a class of neural networks called evolving connectionist systems (ECOSs). ECOSs combine the adaptive/evolving learning ability of neural networks and the approximate reasoning and linguistically meaningful explanation features of symbolic representation, such as fuzzy rules. This review paper includes hybrid expert systems, evolving neuro-fuzzy systems, spiking neural networks, neurogenetic systems, and quantum inspired systems, all discussed from the point of few of their adaptability and model interpretability. The chapter covers both methods and their numerous applications for data modeling, predictive systems, data mining, pattern recognition, across application areas of: engineering, medicine and health, neuroinformatics, bioinformatics, adaptive robotics, etc.

11.1. Early Neuro-Fuzzy Hybrid Systems

The human brain uniquely combines low level neuronal learning in the neurons and the connections between them and higher level abstraction leading to knowledge discovery. This is the ultimate inspiration for the development of the evolving connectionist systems described in this paper.

In the past 50 years or so, several seminal works in the areas of neural networks (Amari, 1967, 1990) and fuzzy systems (Zadeh, 1965, 1988) opened a new field of information science — the creation of new types of hybrid systems that combine the learning ability of neural networks, at a lower level of information processing, and the reasoning and explanation ability of fuzzy rule-based systems, at the higher level. An exemplar system is shown in [Figure 11.1](#) where, at a lower level, a neural network (NN) module predicts the level of a stock index and, at a higher level, a fuzzy reasoning module combines the predicted values with some macro-economic variables, using the following types of fuzzy rules (Kasabov, 1996).

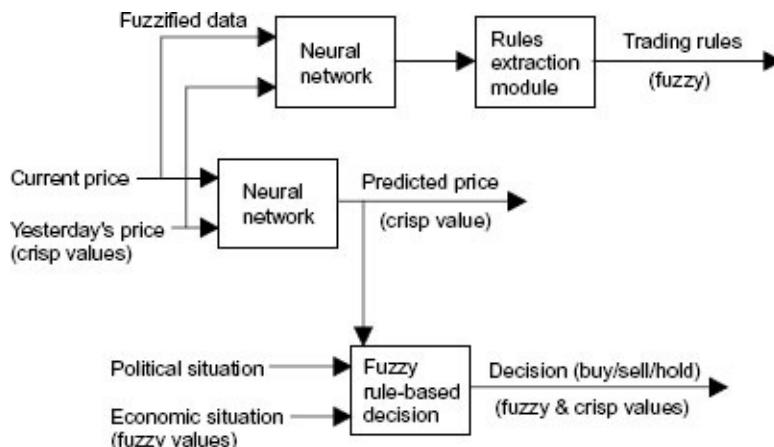


Figure 11.1: A hybrid NN-fuzzy rule-based expert system for financial decision support (Kasabov, 1996).

$$\begin{aligned} &\text{IF } < \text{the predicted by the NN module stock is high} > \\ &\quad \text{AND } < \text{the economic situation is good} > \\ &\quad \text{THEN } < \text{buy stock} >. \end{aligned} \tag{1}$$

These fuzzy expert systems continued the development of the hybrid NN- rule-based expert systems that used crisp propositional and fuzzy rules (Hopfield, 1995; Izhikevich, 2004; Kasabov and Shishkov, 1993).

The integration of neural networks and fuzzy systems into expert systems attracted many researchers to join this theme. The low-level integration of fuzzy rules into a single neuron model and larger neural network structures, tightly coupling learning and fuzzy reasoning rules into connectionists structures, was initiated by Professor Takeshi Yamakawa and other Japanese scientists and promoted at a series of IIZUKA conferences in Japan (Yamakawa *et al.*, 1992; Zadeh, 1965). Many models of fuzzy neural networks were developed based on these principles (Furuhashi *et al.*, 1993; Lin and Lee, 1996; Kasabov, 1996, 1998; Kasabov *et al.*, 1997; Angelov, 2002; Angelov *et al.*, 2010)).

11.2. Evolving Connectionist Systems (ECOS)

11.2.1. Principles of ECOS

The evolving neuro-fuzzy systems developed these ideas further, where instead of training a fixed connectionist structure, the structure and its functionality were evolving from incoming data, often in an online, one-pass learning mode. This is the case with the evolving connectionist systems, ECOS (Kasabov, 1998, 2001, 2003; Kasabov *et al.*, 1997; Kasabov and Song, 2002; Angelov, 2002).

ECOS are modular connectionist-based systems that evolve their structure and functionality in a continuous, self-organized, online, adaptive, interactive way from incoming information (Kasabov, 1998). They can process both data and knowledge in a supervised and/or unsupervised way. ECOS learn local models from data through clustering of the data and associating a local output function for each cluster represented in a connectionist structure. They can learn incrementally single data items or chunks of data and also incrementally change their input features (Kasabov, 2003, 2007). Elements of ECOS have been proposed as part of the classical NN models, such as SOM, RBF, FuzyARTMap, Growing neural gas, neuro-fuzzy systems, RAN (for a review, see Kasabov, 2003). Other ECOS models, along with their applications, have been reported in (Watts, 2009; Futschik and Kasabov, 2002).

The principle of ECOS is based on *local learning*—neurons are allocated as centers of data clusters and the system creates local models in these clusters. Fuzzy clustering, as a mean to create local knowledge-based systems, was stimulated by the pioneering work of Bezdek, Yager and Filev (Bezdek, 1987; Yager and Filev, 1994).

To summarize, the following are the main principles of ECOS as stated in Kasabov (1998):

- (1) Fast learning from large amount of data, e.g., using ‘one-pass’ training, starting with little prior knowledge;
- (2) Adaptation in a real time and in an online mode where new data is accommodated as it comes based on local learning;
- (3) ‘Open’, evolving structure, where new input variables (relevant to the task), new outputs (e.g., classes), new connections and neurons are added/evolved ‘on the fly’;
- (4) Both data learning and knowledge representation are facilitated in a comprehensive and flexible way, e.g., supervised learning, unsupervised learning, evolving clustering, ‘sleep’ learning, forgetting/pruning, fuzzy rule insertion and extraction;
- (5) Active interaction with other ECOSs and with the environment in a multi-modal fashion;
- (6) Representing both space and time in their different scales, e.g., clusters of data, short-

and long-term memory, age of data, forgetting, etc.;

- (7) System's self-evaluation in terms of behavior, global error and success and related knowledge representation.

In 1998, Walter Freeman, who attended the ICONIP conference, commented on the proposed ECOS concepts: "... Through the 'chemicals' and let the system grow ...".

The development of ECOS as a trend in neural networks and computational intelligence that started in 1998 (Kasabov, 1998) continued as many improved or new computational *methods* that use the ECOS principles have been developed along many *applications*.

11.2.2. Neuro-Fuzzy ECOS: EFuNN and DENFIS

Here we will briefly illustrate the concepts of ECOS on two implementations: EFuNN (Kasabov, 2001) and DENFIS (Kasabov and Song, 2002). Examples of EFuNN and DENFIS are shown in Figures 11.2 and 11.3 respectively. In ECOS, clusters of data are created based on similarity between data samples either in the input space (this is the case in some of the ECOS models, e.g., the dynamic neuro-fuzzy inference system DENFIS), or in both the input and output space (this is the case e.g., in the EFuNN models). Samples (examples) that have a distance to an existing node (cluster center, rule node) less than a certain threshold are allocated to the same cluster. Samples that do not fit into existing clusters form new clusters. Cluster centers are continuously adjusted according to new data samples, and new clusters are created incrementally. ECOS learn from data and automatically create or update a local fuzzy model/function, e.g.,

$$IF < \text{data is in a fuzzy cluster } Ci > \text{ THEN } < \text{the model is } Fi >, \quad (2)$$

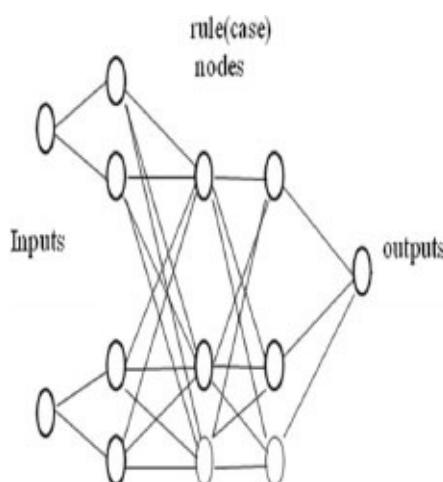


Figure 11.2: An example of EFuNN model.

Source: Kasabov (2001).

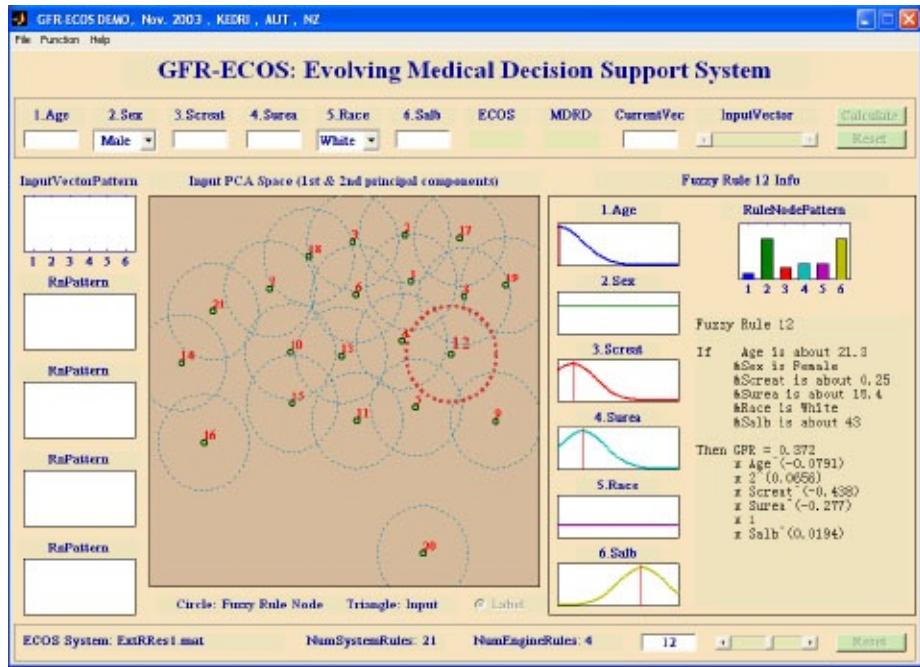


Figure 11.3: An example of DENFIS model.

Source: Marshall *et al.* (2005); Kasabov (2002, 2003).

where F_i can be a fuzzy value, a logistic or linear regression function (Figure 11.3) or a NN model (Kasabov and Song, 2002; Kasabov, 2003)

A special development of ECOS is *transductive reasoning and personalized modeling*. Instead of building a set of local models (e.g., prototypes) to cover the whole problem space and then use these models to classify/predict any new input vector, in transductive modeling for every new input vector, a new model is created based on selected nearest neighbor vectors from the available data. Such ECOS models are NFI and TWNFI (Song and Kasabov, 2006). In TWNFI, for every new input vector, the neighborhood of closest data vectors is optimized using both the distance between the new vector and the neighboring ones and the weighted importance of the input variables, so that the error of the model is minimized in the neighborhood area (Kasabov and Hu, 2010).

11.2.3. Methods that Use Some ECOS Principles

Among other methods that use or have been inspired by the ECOS principles are (publications are available from www.ieeexplore.ieee.org; Google Scholar; Scopus):

- Evolving Self-organized Maps (ESOM) (Deng and Kasabov, 2003);
- Evolving Clustering Method (ECM) (Song and Kasabov, 2002);
- Incremental feature learning in ECOS (Ozawa *et al.*, 2010);
- Online ECOS optimization (Minku and Ludemir, 2005; Chan *et al.*, 2004);
- Evolving Takagi–Sugeno fuzzy model based on switching to neighboring models (Angelov and Filev, 2004);
- Clustering and co-evolution to construct neural network ensembles: An experimental

study (Minku and Ludermir, 2006).

11.2.4. ECOS-Based Applications

Based on the ECOS concepts and methods, sustained engineering applications have been developed, some of them are included in the list below and presented in (Kasabov, 2007) in detail:

- Discovery of diagnostic markers for early detection of cancer, (see also, www.pebl.co.nz);
 - Medical diagnosis of renal function evaluation using DENFIS (also in Marshall *et al.*, 2005);
 - Risk analysis and discovery of evolving economic clusters in Europe (see Kasabov, 2003);
 - Adaptive robot control system based on ECOS (see Huang, 2008);
 - Personalized modeling systems (see Kasabov and Hu, 2010)
- Other applications include:
- Phoneme-base speech recognition;
 - EEG signal processing;
 - Adaptive image processing;
 - Language modeling;
 - Ecological event prediction;
 - Decision support in economics;
 - A software agent framework to overcome malicious host threats and uncontrolled agent clones (Sujitha and Amudha, 2012);
 - Autonomous visual self-localization in completely unknown environment using evolving fuzzy rule-based classifier (Zhou and Angelov, 2007).

While the ECOS methods presented above use McCulloch and Pitts model of a neuron, the further developed evolving spiking neural network (eSNN) architectures use a spiking neuron model applying the same or similar ECOS principles and applications.

11.3. Evolving Spiking Neural Networks (eSNN)

11.3.1. Main Principles, Methods and Examples of eSNN

A single biological neuron and the associated synapses is a complex information processing machine that involves short term information processing, long-term information storage, and evolutionary information stored as genes in the nucleus of the neuron. A spiking neuron model assumes input information represented as trains of spikes over time. When sufficient input information is accumulated in the membrane of the neuron, the neuron's post synaptic potential exceeds a threshold and the neuron emits a spike at its axon ([Figure 11.4](#)).

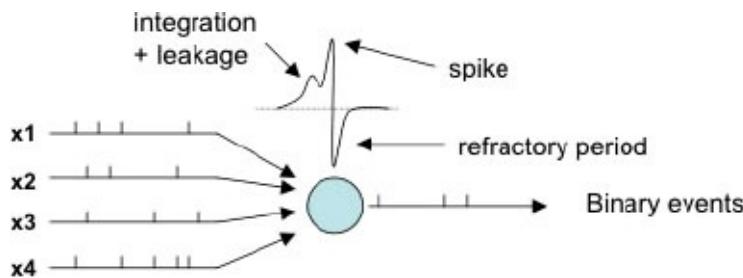


Figure 11.4: The structure of the LIFM.

Some of the-state-of-the-art models of a spiking neuron include: early models by Hodgkin and Huxley (1952); more recent models by Maas, Gerstner, Kistler, Izhikevich and others, e.g., Spike Response Models (SRM); Integrate-and-Fire Model (IFM) ([Figure 11.4](#)); Izhikevich models; adaptive IFM; probabilistic IFM (for details, see: Hebb, 1949; Gerstner, 1995; Hopfield, 1995; Izhikevich, 2004; Kasabov, 2010).

Based on the ECOS principles, an evolving spiking neural network architecture (eSNN) was proposed (Kasabov, 2007; Wysoski *et al.*, 2010). It was initially designed as a visual pattern recognition system. The first eSNNs were based on the Thorpe's neural model (Thorpe and Delorme, 2001), in which the importance of early spikes (after the onset of a certain stimulus) is boosted, called rank-order coding and learning. Synaptic plasticity is employed by a fast supervised one-pass learning algorithm. Different eSNN models were developed, including:

- Reservoir-based eSNN for spatio- and spectro-temporal pattern recognition shown in [Figure 11.5](#) (following the main principles from Verstraeten *et al.*, 2007);
- Dynamic eSNN (deSNN) (Kasabov *et al.*, 2013a) a model that uses both rank-order and time-based STDP learning (Song *et al.*, 2000) to account for spatio-temporal data; and many more (for references, see Schliebs and Kasabov, 2013):

Extracting fuzzy rules from an eSNN would make the eSNN not only efficient learning models, but also knowledge-based models. A method was proposed (Soltic and Kasabov, 2010) and illustrated in [Figures 11.6](#) and [11.7](#). Based on the connection weights (W) between the receptive field layer (L1) and the class output neuron layer (L2), the

following fuzzy rules are extracted:

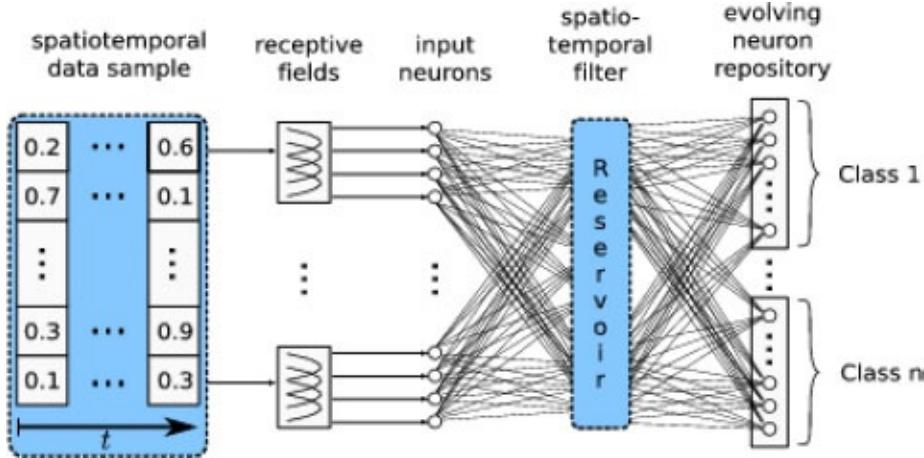


Figure 11.5: A reservoir-based eSNN for spatio-temporal pattern classification.

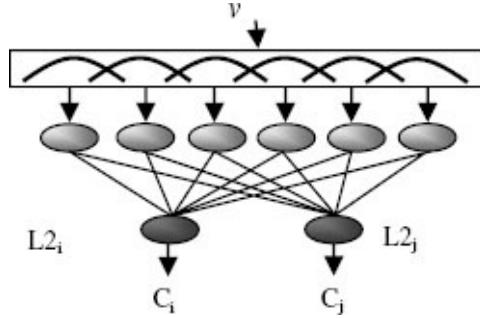


Figure 11.6: A simplified structure of an eSNN for two-class classification showing only one input variable using six receptive fields to convert the input values into spike trains.

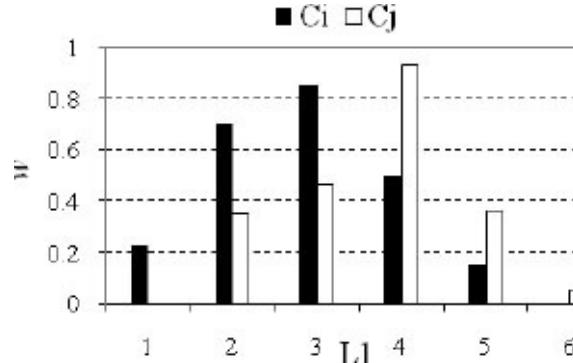


Figure 11.7: The connection weights of the connections to class C_i and C_j output neurons respectively are interpreted as fuzzy rules (Equation (3)).

$$\begin{aligned} & \text{IF (input variable } v \text{ is SMALL) THEN class } C_i; \\ & \text{IF } (v \text{ is LARGE) THEN class } C_j. \end{aligned} \quad (3)$$

In principle, eSNN uses spike information representation, spiking neuron models and SNN learning and encoding rules and the structure is evolving to capture spatio-temporal relationship from data. The eSNNs have been further developed as computational neurogenetic models as discussed below.

11.3.2. Quantum Inspired Optimization of eSNN

eSNNs have several parameters that need to be optimized for an optimal performance.

Several successful methods have been proposed for this purpose, among them are: Quantum-inspired evolutionary algorithm, QiEA (Defoin-Platel *et al.*, 2009); and Quantum inspired particle swarm optimization method, QiPSO (Nuzly *et al.*, 2010).

Quantum inspired optimization methods use the principle of superposition of states to represent and optimize features (input variables) and parameters of the eSNN (Kasabov, 2007). Features and parameters are represented as qubits that are in a superposition of 1 (selected), with a probability α , and 0 (not selected) with a probability β . When the model has to be calculated, the quantum bits ‘collapse’ in 1 or 0.

11.3.3. Some Applications Based on eSNN

Numerous applications based on the different eSNN models have been reported, among them:

- Advanced spiking neural network technologies for neurorehabilitation (Chen *et al.*, 2013);
- Object movement recognition (EU FP7 Marie Curie EvoSpike project 2011–2012, INI/ETH/UZH, <http://ncs.ethz.ch/projects/evospike>);
- Multimodal audio and visual information processing (Wysoski *et al.*, 2010);
- Ecological data modeling and prediction of the establishment of invasive species (Schliebs *et al.*, 2009);
- Integrated brain data analysis (Kasabov, 2014);
- Predictive modeling method and case study on personalized stroke occurrence prediction (Kasabov, *et al.*, 2014d);
- Financial time series prediction (Widiputra *et al.*, 2011);
- Other applications (Schliebs and Kasabov, 2013).

11.4. Computational Neuro-Genetic Models (CNGM) Based on eSNN

11.4.1. Main Principles

A neurogenetic model of a neuron is proposed in (Kasabov, 2007) and studied in (Benuskova and Kasabov, 2007). It utilizes information about how some proteins and genes affect the spiking activities of a neuron such as *fast excitation, fast inhibition, slow excitation, and slow inhibition*. An important part of the model is a dynamic gene/protein regulatory network (GRN) model of the dynamic interactions between genes/proteins over time that affect the spiking activity of the neuron—[Figure 11.8](#).

New types of neuro-genetic fuzzy rules can be extracted from such CNGM in the form of:

$$\begin{aligned} \text{IF } &< \text{GRN is represented by a function } F > \text{ AND } < \text{input is Small} > \\ &\text{THEN } < \text{Class } C >. \end{aligned} \quad (4)$$

This type of eSNN is further developed into a comprehensive SNN framework called NeuCube for spatio/spectro temporal data modeling and analysis as presented in the next section.

11.4.2. The NeuCube Architecture

The latest development of neurogenetic systems is NeuCube (Kasabov, 2014), initially designed for spatio-temporal brain data modeling, but then it was used for climate data modeling, stroke occurrence prediction and other applications (Kasabov, *et al.*, 2015).

The NeuCube framework is depicted in [Figure 11.9](#). It consists of the following functional parts (modules):

- Input information encoding module;
- 3D SNN reservoir module (SNNr);

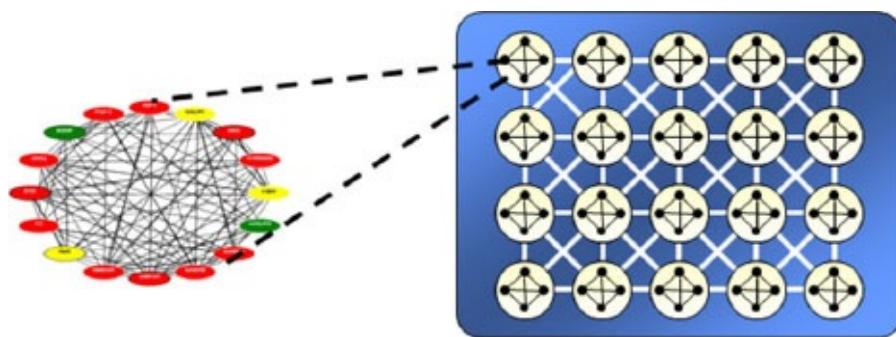


Figure 11.8: A schematic diagram of a CNGM framework, consisting of a GRN as part of a eSNN (Benuskova and Kasabov, 2007).

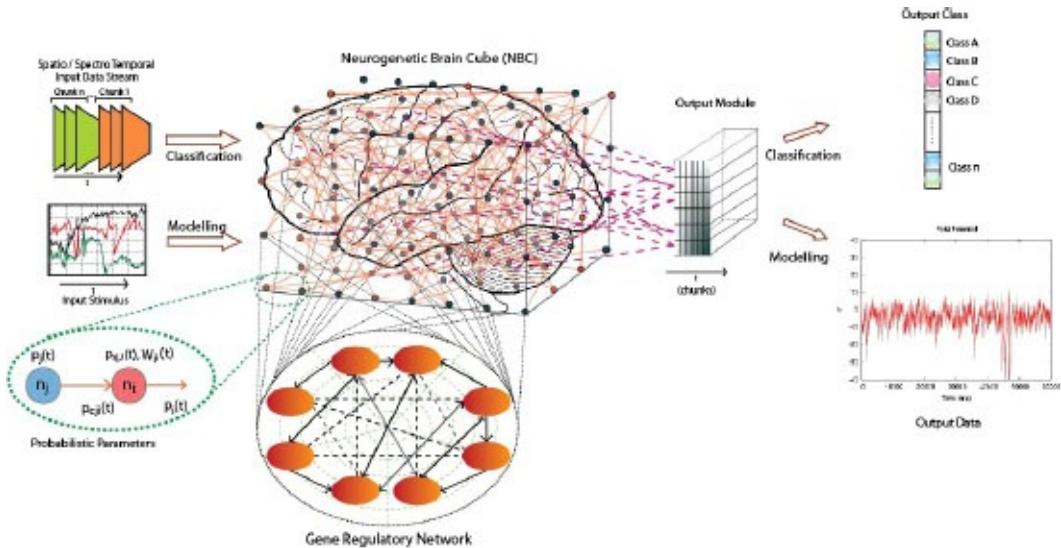


Figure 11.9: A block diagram of the NeuCube architecture (from Kasabov, 2014).

- Output (e.g., classification) module;
- Gene regulatory network (GRN) module (Optional);
- Optimization module.

The input module transforms input data into trains of spikes. Spatio-temporal data (such as EEG, fMRI, climate) is entered into the main module—the 3D SNN cubes (SNNc). Input data is entered into *pre-designated spatially located* areas of the SNNc that correspond to the spatial location in the origin where data was collected (if there is such).

Learning in the SNN is performed in two stages:

- Unsupervised training, where spatio-temporal data is entered into relevant areas of the SNNc over time. Unsupervised learning is performed to modify the initially set connection weights. The SNNc will learn to activate same groups of spiking neurons when similar input stimuli are presented, also known as a *polychronization* effect (Izhikevich, 2004).
- Supervised training of the spiking neurons in the output classification module, where the same data that was used for unsupervised training is now propagated again through the trained SNNc and the output neurons are trained to classify the spatio-temporal spiking pattern of the SNNc into pre-defined classes (or output spike sequences). As a special case, all neurons from the SNN are connected to every output neuron. Feedback connections from output neurons to neurons in the SNN can be created for reinforcement learning. Different SNN methods can be used to learn and classify spiking patterns from the SNNr, including the deSNN (Kasabov *et al.*, 2013b) and SPAN models (Mohammed *et al.*, 2013). The latter is suitable for generating spike trains in response to certain patterns of activity of the SNNc.

Memory in the NeuCube architecture is represented as a combination of the three types of memory described below, which are mutually interacting:

- Short-term memory, represented as changes of the PSP and temporary changes of synaptic efficacy;
- Long-term memory, represented as a stable establishment of synaptic efficacy—LTP and LTD;
- Genetic memory, represented as a genetic code.

In NeuCube, similar activation patterns (called ‘polychronous waves’) can be generated in the SNNc with recurrent connections to represent short term memory. When using STDP learning connection weights change to form LTP or LTD, which constitute long-term memory. Results of the use of the NeuCube suggest that the NeuCube architecture can be explored for learning long (spatio-) temporal patterns and to be used as associative memory. Once data is learned, the SNNc retains the connections as a long-term memory. Since the SNNc learns functional pathways of spiking activities represented as structural pathways of connections, when only a small initial part of input data is entered the SNNc will ‘synfire’ and ‘chain-fire’ *learned connection pathways* to reproduce *learned functional pathways*. Thus, a NeuCube can be used as an associative memory and as a predictive system.

11.4.3. Neuromorphic Implementations

The different types of eSNN and neurogenetic systems, especially the NeuCube architecture, are suitable to be implemented as a neuromorphic hardware system for embedded applications. For this purpose, both digital (e.g., Furber, 2012) and analogue (e.g., Indiveri *et al.*, 2011) realizations can be used.

11.5. Conclusion

This chapter presents briefly the methods of ECOS. ECOS facilitate adaptive learning and knowledge discovery from complex data. ECOS principles are derived from neural networks, fuzzy systems, evolutionary computation, quantum computing and brain information processing. ECOS applications are manifold, but perhaps most welcome in the environmental and health sciences, where the diagnostic phenomena are chaotic in nature and the datasets are massive and often incomplete. In the field of sustainability science, whether it is in analyzing issues related to sustainable resource utilization, countering global environmental issues, or the assurance of the continuity of life, (particularly human life) on earth, the speed of transformation is most rapid. Massive datasets with the characteristics just described need to be analyzed, virtually in real time, for prognoses to be made and solutions to the issues sought at a level of urgency. In this sense, evolving connectionist systems for adaptive learning and knowledge discovery can make a great contribution to the methodologies of intelligent evolving systems (Angelov *et al.*, 2010). In the future, new methods of ECOS can be developed based on the integration of principles from computational intelligence, bioinformatics and neuroinformatics (Kasabov, 2014).

Acknowledgment

The work on this chapter is supported by the Knowledge Engineering and Discovery Research Institute (KEDRI, <http://www.kedri.aut.ac.nz>). Some of the methods, techniques and applications are publicly available from: <http://www.kedri.aut.ac.nz>; <http://www.theneucom.com>; <http://www.kedri.aut.ac.nz/neucube/>. I would like to acknowledge the work of many authors who developed methods and applications of ECOS referring to the presented here material in more than 10,000 citations (Google Scholar, September 2015). I have not been able to reference all or even part of their work. Many of them have contributed greatly to the development and the applications of ECOS e.g., S. Amari, T. Yamakawa, W. Freeman, J. Taylor, S. Grossberg, P. Angelov, D. Filev, J. Pratt, T. Ludemir, S. Ozawa, P. Pang, Lughofe, B. Gabrys, M. Watts, Dovzan and Skrjanc, F. Gomide, Ang and Quek, Rubio, Aznarte, Cetisli, Subramian and Suresh, Babu and Suresh, Tung, Boubacar, Yamauchi, Hayashi, L. Goh, Ishibuchi, Hwang and Song, Capo, Vachkov, Rutkowski and Cpalka, Han and Qiao, Zanchettin, O'Hora, Hernandez and Castaeda, Guzaitis, and many more. I would like to acknowledge the editor of this volume Plamen Angelov for his incredible energy and efforts to put all these chapters together.

References

- Amari, S. (1967). A theory of adaptive pattern classifiers. *IEEE Trans. Electron. Comput.*, 16, pp. 299–307.
- Amari, S. (1990). Mathematical foundations of neurocomputing. *Proc. IEEE*, 78, pp. 1143–1163.
- Angelov, P. (2002). *Evolving Rule-based Models: A Tool for Design of Flexible Adaptive Systems*. Heidelberg, Germany: Springer-Verlag.
- Angelov, P. and Filev, D. (2004). An approach to on-line identification of evolving Takagi–Sugeno models. *IEEE Trans. Systems, Man and Cybernetics, Part B*, 34(1), pp. 484–498.
- Angelov, P., Filev, D. and Kasabov, N. (eds.). (2010). *Evolving Intelligent Systems*. USA: IEEE Press and Wiley.
- Benuskova, L and Kasabov, N. (2007). *Computational Neuro-Genetic Modeling*. New York: Springer.
- Bezdek, J. (ed.). (1987). *Analysis of Fuzzy Information, Volumes 1–3*. Boca Raton, Florida: CRC Press.
- Chan, Z. and Kasabov, N. (2004). Evolutionary computation for on-line and off-line parameter tuning of evolving fuzzy neural networks. *Int. J. Comput. Intell. Appl.*, 4(3), pp. 309–319.
- Chen, Y., Hu, J., Kasabov, N., Hou, Z. and Cheng, L. (2013). NeuroCubeRehab: a pilot study for EEG classification in rehabilitation practice based on spiking neural networks. *Proc. ICONIP 2013, Springer LNCS*, 8228, pp. 70–77.
- Defoin-Platel, M., Schliebs, S. and Kasabov, N. (2009). Quantum-inspired evolutionary algorithm: a multi-model EDA. *IEEE Trans. Evol. Comput.*, 13(6), pp. 1218–1232.
- Deng, D. and Kasabov, N. (2003). On-line pattern analysis by evolving self-organizing maps. *Neurocomput.*, 51, pp. 87–103.
- EU FP7 Marie Curie EvoSpike project (2011–2012). INI/ETH/UZH. (<http://ncs.ethz.ch/projects/evospike>).
- Furber, S. (2012). To build a brain. *IEEE Spectr.*, 49(8), pp. 39–41.
- Furuhashi, T., Hasegawa, T., Horikawa, S. and Uchikawa, Y. (1993). An adaptive fuzzy controller using fuzzy neural networks. In *Proc. Fifth IFSA World Congr.*, pp. 769–772.
- Futschik, M. and Kasabov, N. (2002) Fuzzy clustering in gene expression data analysis. *Proc. World Congr. Comput. Intell. WCCI'2002*, May 2002. Hawaii: IEEE Press.
- Gerstner, W. (1995). Time structure of the activity of neural network models. *Phys. Rev.*, 51, pp. 738–758.
- Hebb, D. (1949). *The Organization of Behavior*. New York: John Wiley and Sons.
- Hodgkin, A. L. and Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.*, 117, pp. 500–544.
- Hopfield, J. (1995). Pattern recognition computation using action potential timing for stimulus representation. *Nature*, 376, pp. 33–36.
- Huang, L. (2008). Evolving connectionist system based role allocation for robotic soccer. *Int. J. Adv. Robot. Syst.*, 5(1), pp. 59–62.
- Indiveri, G., Linares-Barranco, B., Hamilton, T., Van Schaik, A., Etienne-Cummings, R., Delbrück, T., Liu, S., Dudek, P., Hafliger, P. and Renaud, S. (2011). Neuromorphic silicon neuron circuits. *Front. Neurosci.*, 5, pp. 1–23.
- Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons? *IEEE TNN*, 15(5), pp. 1063–1070.
- Kasabov, N. (1991). Incorporating neural networks into production systems and a practical approach towards the realisation of fuzzy expert systems. *Comput. Sci. Inf.*, 21(2), pp. 26–34.
- Kasabov, N. (1993). Hybrid connectionist production systems. *J. Syst. Eng.*, 3(1), pp. 15–21.
- Kasabov, N. (1994). Connectionist fuzzy production systems. *LNCS/AI*, 847, pp. 114–128.
- Kasabov, N. (1995). Hybrid connectionist fuzzy production systems: Towards building comprehensive AI. *Intell. Autom. Soft Comput.*, 1(4), pp. 351–360.
- Kasabov, N. (1996). *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering*. Cambridge, Massachussets: MIT Press, p. 550.
- Kasabov, N. (1998). Evolving fuzzy neural networks: Algorithms, applications and biological motivation. In Yamakawa,

- T. and Matsumoto, G. (eds.), *Methodologies for the Conception, Design and Application of Soft Computing*. Singapore: World Scientific, pp. 271–274.
- Kasabov, N. (2001). Evolving fuzzy neural networks for on-line supervised/unsupervised, knowledge-based learning. *IEEE Trans. SMC/B, Cybern.*, 31(6), pp. 902–918.
- Kasabov, N. (2003). *Evolving Connectionist Systems: Methods and Applications in Bioinformatics, Brain Study and Intelligent Machines*. London, New York, Heidelberg: Springer Verlag.
- Kasabov, N. (2007). *Evolving Connectionist Systems: The Knowledge Engineering Approach*. Berlin, Heidelberg: Springer.
- Kasabov, N. (2010). To spike or not to spike: A probabilistic spiking neural model. *Neural Networks*, 23(1), pp. 16–19.
- Kasabov, N. (2014). NeuCube: a spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data. *Neural Netw.*, 52, pp. 62–76. <http://dx.doi.org/10.1016/j.neunet.2014.01.006>.
- Kasabov, N. (ed.) (2014). *The Springer Handbook of Bio- and Neuroinformatics*. New York: Springer.
- Kasabov, N. et al. (2015). Design methodology and selected applications of evolving spatio-temporal data machines in the NeuCube neuromorphic framework. *Neural Networks*, in print.
- Kasabov, N. and Hu, Y. (2010). Integrated optimization method for personalized modeling and case study applications. *Int. J. Funct. Inf. Person. Med.*, 3(3), pp. 236–256.
- Kasabov, N. and Shishkov, S. (1993). A connectionist production system with partial match and its use for approximate reasoning. *Connect. Sci.*, 5(3/4), pp. 275–305.
- Kasabov, N., Kim, J. S., Watts, M. and Gray, A. (1997). FuNN/2: a fuzzy neural network architecture for adaptive learning and knowledge acquisition. *Inf. Sci. Appl.*, 101(3–4), pp. 155–175.
- Kasabov, N. and Song, Q. (2002). DENFIS: dynamic, evolving neural-fuzzy inference systems and its application for time-series prediction. *IEEE Trans. Fuzzy Syst.*, 10, pp. 144–154.
- Kasabov, N., Liang, L., Krishnamurthi, R., Feigin, V., Othman, M., Hou, Z. and Parmar, P. (2014). Evolving spiking neural networks for personalized modeling of spatio-temporal data and early prediction of events: a case study on stroke. *Neurocomputing*, 134, pp. 269–279.
- Kasabov, N., Dhoble, K., Nuntalid, N. and Indiveri, G. (2013b). Dynamic evolving spiking neural networks for on-line spatio- and spectro-temporal pattern recognition. *Neural Netw.*, 41, pp. 188–201.
- Lin, C. T. and Lee, C. S. G. (1996). *Neuro Fuzzy Systems*. US: Prentice Hall.
- Marshall, M. R. et al. (2005). Evolving connectionist system versus algebraic formulae for prediction of renal function from serum creatinine. *Kidney Int.*, 6, pp. 1944–1954.
- Minku, F. L. and Ludermir, T. B. (2006). EFuNNs ensembles construction using a clustering method and a coevolutionary genetic algorithm. *Proc. IEEE Congress on Evolutionary Computation*, Vancouver, July 16–21, IEEE Press, pp. 5548–5555.
- Minku, F. L. and Ludermir, T. B. (2005). Evolutionary strategies and genetic algorithms for dynamic parameter optimisation of evolving fuzzy neural networks. *Proc. IEEE Congress on Evolutionary Computation (CEC)*, Edinburgh, September, pp. 1951–1958.
- Mohammed, A., Schliebs, S., Matsuda, S. and Kasabov, N. (2013). Evolving spike pattern association neurons and neural networks. *Neurocomputing*, 107, pp. 3–10.
- Nuzly, H., Kasabov, N. and Shamsuddin, S. (2010). Probabilistic evolving spiking neural network optimization using dynamic quantum inspired particle swarm optimization. In *Springer LNCS*, 6443.
- Ozawa, S., Pang, S. and Kasabov, N. (2008). Incremental learning of chunk data for on-line pattern classification systems. *IEEE Trans. Neural Networks*, 19(6), pp. 1061–1074.
- Schliebs, S. and Kasabov, N. (2013). Evolving spiking neural network—a survey. *Evolving Systems*, 4(2), pp. 87–98.
- Schliebs, S. et al. (2009). Integrated feature and parameter optimization for evolving spiking neural networks: exploring heterogeneous probabilistic models. *Neural Netw.*, 22, pp. 623–632.
- Soltic, S. and Kasabov, N. (2010). Knowledge extraction from evolving spiking neural networks with rank order population coding. *Int. J. Neural Syst.*, 20(6), pp. 437–445.

- Song, Q. and Kasabov, N. (2006). TWNFI: A transductive neuro-fuzzy inference system with weighted data normalization for personalized modeling. *Neural Netw.*, 19(10), pp. 1591–1596.
- Song, S., Miller, K. and Abbott, L. (2000). Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neurosci.*, 3, pp. 919–926.
- Sujitha, G. A. and Amudha, T. (2012). *Int. J. Adv. Inf. Technol.*, 2(2), pp. 13–27.
- Thorpe, S. and Delorme, A. (2001). Spike-based strategies for rapid processing. *Neural Netw.*, 14(6–7), pp. 715–725.
- Verstraeten, D., Schrauwen, B., D’Haene, M. and Stroobandt, D. (2007). An experimental unification of reservoir computing methods. *Neural Netw.*, 20(3), pp. 391–403.
- Watts, M. (2009). A decade of Kasabov’s evolving connectionist systems: a review. *IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev.*, 39(3), pp. 253–269.
- Widiputra, H., Pears, R. and Kasabov, N. (2011). Multiple time-series prediction through multiple time-series relationships profiling and clustered recurring trends. *Springer LNAI*, 6635(2), pp. 161–172.
- Wysoski, S., Benuskova, L. and Kasabov, N. (2010). Evolving spiking neural networks for audiovisual information processing. *Neural Netw.*, 23(7), pp. 819–835.
- Yager, R. R. and Filev, D. (1994). Generation of fuzzy rules by mountain clustering. *J. Intell. Fuzzy Syst.*, 2, pp. 209–219.
- Yamakawa, T., Uchino, E., Miki, T. and Kusanagi, H. (1992). A neo fuzzy neuron and Its application to system identification and prediction of the system behavior. *Proc. Second Int. Conf. Fuzzy Log. Neural Netw.*, Iizuka, Japan, pp. 477–483.
- Zadeh, L. (1965). Fuzzy Sets. *Inf. Control*, 8, pp. 338–353.
- Zadeh, L. A. (1988). Fuzzy Logic. *IEEE Comput.*, 21, pp. 83–93.
- Zanchettin, C. and Ludermir, T. B. (2004). Evolving fuzzy neural networks applied to odor recognition in an artificial nose. *Proc. IEEE Int. Joint Conf. Neural Networks*, Budapest, July 26–29, IEEE Press.
- Zhou, X. and Angelov, P. (2006). Real-time joint landmark recognition and classifier generation by an evolving fuzzy system. *Proc. of IEEE Int. Conf. Fuzzy Systems*, Vancouver, July 16–21, IEEE Press, pp. 6314–6321.

Chapter 12

Reinforcement Learning with Applications in Automation Decision and Feedback Control

Kyriakos G. Vamvoudakis, Frank L. Lewis and Draguna Vrabie

This book chapter is interested in showing how ideas from Control Systems Engineering and Computational Intelligence have been combined to obtain a new class of control techniques. Since reinforcement learning involves modifying the control policy based on responses from an unknown environment, we have the initial feeling that it is closely related to adaptive control. Moreover, reinforcement learning methods allow learning of optimal solutions relative to prescribed cost/payoff metrics by measuring data in real time along system trajectories, hence it also has relations to optimal feedback control. This book chapter is an exposition of research results that clarify these relations.

12.1. Background

Adaptive control is one of the most widespread techniques for feedback control of modern engineered systems since the 1960s. Its effective and reliable use in aerospace systems, the industrial process industry, vehicle control, communications, and elsewhere is firmly established. Optimal control is equally widespread since it enables the design of feedback controllers with the purpose of minimizing energy, fuel consumption, performance time, or other premium quantities.

Optimal controllers are normally designed offline by solving Hamilton–Jacobi–Bellman (HJB) equations, for example, the Riccati equation, using complete knowledge of the system dynamics. Optimal control policies for nonlinear systems can be obtained by solving nonlinear HJB equations that cannot usually be solved. By contrast, adaptive controllers learn online to control unknown systems using data measured in real time along the system trajectories. Adaptive controllers are generally not optimal in the sense of minimizing user-prescribed performance functions. Indirect adaptive controllers have been designed that use system identification techniques to first identify the system parameters, then use the obtained model to solve optimal design equations (Astrom and Wittenmark, 1995). It is shown that adaptive controllers may satisfy certain inverse optimality conditions (Ioannou and Fidan, 2006). The design of adaptive controllers that learn online, in real time, the solutions to user-prescribed optimal control problems can be studied by means of reinforcement learning and adaptive dynamic programming techniques, and forms the object of this book chapter which is a condensation of the paper that appeared in IEEE Control Systems Magazine in December 2012 (Lewis *et al.*, 2012a).

The real-time learning of optimal controllers for unknown systems occurs in nature. The limits within which living organisms can survive are often quite narrow and the resources available to most species are meager. Therefore, most organisms occurring in nature act in an optimal fashion to conserve resources while achieving their goals. Living organisms learn by acting on their environment, observing and evaluating the resulting reward stimulus, and adjusting their actions accordingly to improve the reward.

Inspired by natural learning mechanisms, that is, those occurring in biological organisms including animals and humans, reinforcement learning techniques for machine learning and artificial intelligence have been developed and used mainly in the Computational Intelligence community (Lewis *et al.*, 2012b; Li and Krstic, 1997; Powell, 2011; Sutton and Barto, 1998; Werbos, 1991). The purpose of this book chapter is to give an exposition of the usefulness of reinforcement learning techniques in designing feedback controllers for engineered systems, where optimal actions may be driven by objectives such as minimum fuel, minimum energy, minimum risk, maximum reward, and so on. We specifically focus on a family of techniques known as approximate or adaptive dynamic programming (ADP) (Balakrishnan *et al.*, 2008; Barto and Powell, 2004; Busoniu *et al.*, 2009; Lewis and Liu, 2013; Lewis *et al.*, 2008; Prokhorov, 2008; Wang *et al.*, 2009). We

show that reinforcement learning ideas can be used to design a family of adaptive control algorithms that converge in real time to optimal control solutions while using data measured along the system trajectories. Such techniques may be referred to as optimal adaptive control. The use of reinforcement learning techniques provides optimal control solutions for linear or nonlinear systems using adaptive control techniques. In effect, reinforcement learning methods allow the solution of HJB equations, or for linear quadratic design, the Riccati equation, online and without knowing the full system dynamics.

In machine learning, reinforcement learning (Lewis *et al.*, 2012b; Powell, 2011; Sutton and Barto, 1998) refers to the set of optimization problems that involve an actor or agent that interacts with its environment and modifies its actions, or control policies, based on stimuli received in response to its actions. reinforcement learning is based on evaluative information from the environment and could be called action-based learning.

Reinforcement learning implies a cause and effect relationship between actions and reward or punishment. It implies goal directed behavior at least insofar as the agent has an understanding of reward versus lack of reward or punishment. The idea behind reinforcement learning is that of modifying actions or behavior policy in order to reduce the prediction error between anticipated future rewards and actual performance as computed based on observed rewards. The reinforcement learning algorithms are constructed on the idea that effective control decisions must be remembered, by means of a reinforcement signal, such that they become more likely to be used a second time. Although the idea originates from experimental animal learning, where it is observed that the dopamine neurotransmitter acts as a reinforcement informational signal that favors learning at the level of the neuron (Doya *et al.*, 2001; Schultz, 2004), reinforcement learning is strongly connected from a theoretical point of view with adaptive control and optimal control methods.

Although RL algorithms have been widely used to solve the optimal regulation problems, some research efforts considered solving the optimal tracking control problem for discrete-time (Zhang *et al.*, 2008) and continuous-time systems (Dierks and Jagannathan, 2010; Zhang *et al.*, 2011). Moreover, existing methods require the exact knowledge of the system dynamics *a priori* while finding the feedforward part of the control input using the dynamic inversion concept. In order to attain the required knowledge of the system dynamics, in Zhang *et al.* (2011), a plant model was first identified and then an RL-based optimal tracking controller was synthesized using the identified model.

One class of reinforcement learning methods is based on the actor-critic structure shown in [Figure 12.1](#), where an actor component applies an action, or control policy, to the environment, and a critic component assesses the value of that action. Based on this assessment of the value, one of several schemes can then be used to modify or improve the

action, or control policy, in the sense that the new policy yields a value that is improved relative to the previous value. The learning mechanism supported by the actor-critic structure implies two steps, namely, policy evaluation, executed by the critic, followed by policy improvement, performed by the actor. The policy evaluation step is performed by observing from the environment the results of applying current actions.

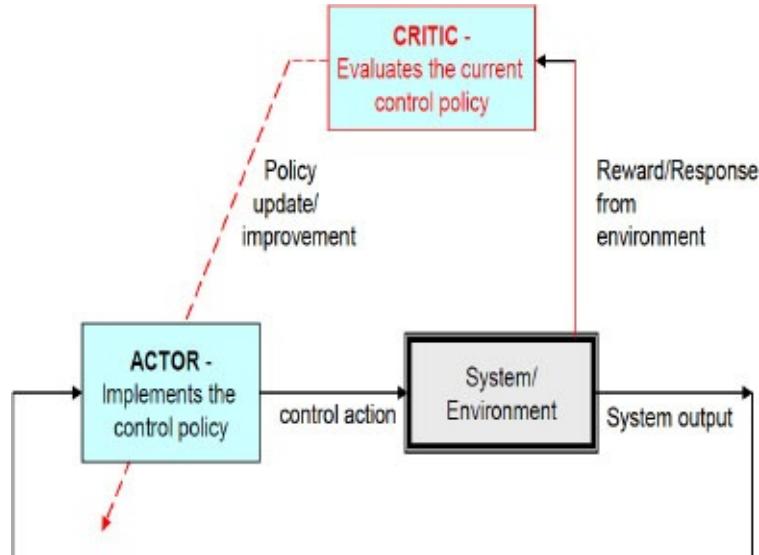


Figure 12.1: Reinforcement learning with an actor/critic structure. This structure provides methods for learning optimal control solutions online based on data measured along the system trajectories.

It is of interest to study reinforcement learning systems that have an actor-critic structure wherein the critic assesses the value of current policies based on some sort of optimality criteria (Busoniu *et al.*, 2009; Cao, 2007; Sutton and Barto, 1998; Werbos, 1992). It is worth noting that the actor-critic structure is similar to one encountered in adaptive control. However, adaptive control schemes make use of system identification mechanisms to first obtain a model of the system dynamics and then use this model to guide adaptation of the parameters of the controller/actor. In contrast, in the optimal reinforcement learning algorithm case the learning process is moved to a higher level, having no longer as object of interest the details of a system's dynamics, but a performance index that quantifies how close to optimality the closed loop control system operates. In this scheme, reinforcement learning is a means of learning optimal behaviors by observing the real-time responses from the environment to non-optimal control policies. Two new research monographs on this subject are Vrabie *et al.* (2012) and Zhang *et al.* (2013).

The intention of this book chapter is to present the main ideas and algorithms of reinforcement learning and approximate dynamic programming and to show how they are applied in the design of optimal adaptive feedback controllers. A few connections are also shown between reinforcement learning and decision and control on systems distributed over communication graphs, including shortest path problems, relaxation methods, and cooperative control. In fact, we show that reinforcement learning provides connections between optimal control, adaptive control, and cooperative control and decisions on finite

graphs. This book chapter presents an expository development of ideas from reinforcement learning and ADP and their applications in automatic control systems. Surveys of ADP are given in (Barto *et al.* (1983); Bertsekas and Tsitsiklis, 1996; Bellman, 1957; Vrabie and Lewis (2009); Werbos, 1989).

12.2. Markov Decision Processes and Stochasticity

A framework for studying reinforcement learning is provided by Markov decision processes (MDP). In fact, dynamical decision problems can be cast into the framework of MDP. Included are feedback control systems for human engineered systems, feedback regulation mechanisms for population balance and survival of species (Prokhorov and Wunsch, 1997), the balance of power between nations, decision-making in multi-player games, and economic mechanisms for regulation of global financial markets. Therefore, we provide a development of MDP here.

Consider the MDP (X, U, P, R) where X is a set of states and U is a set of actions or controls. The transition probabilities $P : X \times U \times X \rightarrow [0, 1]$ give for each state $x \in X$ and action $u \in U$ the conditional probability $P_{x,x'}^u = Pr\{x'|x, u\}$ of transitioning to state $x' \in X$ given the MDP is in state x and takes action u . The cost function $R : X \times U \times X \rightarrow \text{IR}$ gives the expected immediate cost $R_{xx'}^u$ paid after transition to state $x' \in X$ given the MDP starts in state $x \in X$ and takes action $u \in U$. The Markov property refers to the fact that transition probabilities $P_{x,x'}^u$ depend only on the current state x and not on the history of how the MDP attained that state.

The basic problem for MDP is to find a mapping $\pi : X \times U \rightarrow [0, 1]$ that gives for each state x and action u the conditional probability $\pi(x, u) = Pr\{u|x\}$ of taking action u given the MDP is in state x . Such a mapping is termed a closed-loop control or action strategy or policy. The strategy or policy is called stochastic or mixed if there is a non-zero probability of selecting more than one control when in state x . We can view mixed strategies as probability distribution vectors having as component i the probability of selecting the i th control action while in state $x \in X$. If the mapping $\pi : X \times U \rightarrow [0, 1]$ admits only one control with probability 1, when in every state x the mapping is called a deterministic policy. Then $\pi(x, u) = Pr\{u|x\}$ corresponds to a function mapping states into controls $\mu(x) : X \rightarrow U$.

12.2.1. Optimal Sequential Decision Problems

Dynamical systems evolve causally through time. Therefore, we consider sequential decision problems and impose a discrete stage index k such that the MDP takes an action and changes states at nonnegative integer stage values k . The stages may correspond to time or more generally to sequences of events. We refer to the stage value as the time. Denote state values and actions at time k by x_k, u_k . MDP evolve in discrete time. It is often desirable for human engineered systems to be optimal in terms of conserving resources such as cost, time, fuel, energy. Thus, the notion of optimality should be captured in selecting control policies for MDP. Define, therefore, a stage cost at time k by $r_k \equiv r_k(x_k, u_k, x_{k+1})$. Then $R_{xx'}^u = E\{r_k|x_k = x, u_k = u, x_{k+1} = x'\}$, with $E\{\cdot\}$ the expected value operator. Define a performance index as the sum of future costs over the time interval $[k, k + T]$, $T > 0$,

$$J_{k,T} = \sum_{i=0}^T \gamma^i r_{k+i} \equiv \sum_{i=k}^{k+T} \gamma^{i-k} r_i, \quad (1)$$

where $0 \leq \gamma < 1$ is a discount factor that reduces the weight of costs incurred further in the future.

Usage of MDP in the fields of computational intelligence and economics usually considers r_k as a reward incurred at time k , also known as utility and $J_{k,T}$ as a discounted return also known as strategic reward. We refer instead to state costs and discounted future costs to be consistent with objectives in the control of dynamical systems. For convenience we call r_k the utility.

Consider that an agent selects a control policy $\pi_k(x_k, u_k)$ and uses it at each stage k of the MDP. We are primarily interested in stationary policies, where the conditional probabilities $\pi_k(x_k, u_k)$ are independent of k . Then $\pi_k(x, u) \equiv \pi(x, u) = Pr\{u|x\}, \forall k$. Non-stationary deterministic policies have the form $\pi = \{\mu_0, \mu_1, \dots\}$, where each entry is a function $\mu_k(x) : X \rightarrow U ; k = \{0, 1, \dots\}$. Stationary deterministic policies are independent of time so that $\pi = \{\mu, \mu, \dots\}$.

Select a fixed stationary policy $\pi(x, u) = Pr\{u|x\}$. Then the ‘closed-loop’ MDP reduces to a Markov chain with state space X . That is, the transition probabilities between states are fixed with no further freedom of choice of actions. The transition probabilities of this Markov chain are given by,

$$p_{x,x'} \equiv P_{x,x'}^\pi = \sum_u Pr\{x'|x, u\} Pr\{u|x\} = \sum_u \pi(x, u) P_{x,x'}^u, \quad (2)$$

where the Chapman–Kolmogorov identity is used (Papoulis, 2002).

A Markov chain is ergodic if all the states are positive recurrent and aperiodic (Papoulis, 2002). Under the assumption that the Markov chain corresponding to each policy, with transition probabilities given as in Equation (2), is ergodic, it can be shown that every MDP has a stationary deterministic optimal policy (Al-Tamimi *et al.*, 2008; Bellman, 1957; Bertsekas and Tsitsiklis, 1996; Liu *et al.*, 2013; Papoulis, 2002; Prokhorov and Wunsch, 1997; Vrabie *et al.*, 2009; Werbos, 1989; Wheeler and Narendra, 1986). Then for a given policy there exists a stationary distribution $p_\pi(x)$ over X that gives the steady-state probability the Markov chain is in state x .

The value of a policy is defined as the conditional expected value of future cost when starting in state x at time k and following policy $\pi(x, u)$ thereafter,

$$V_k^\pi(x) = E_\pi\{J_{k,T}|x_k = x\} = E_\pi \left\{ \sum_{i=k}^{k+T} \gamma^{i-k} r_i | x_k = x \right\}, \quad (3)$$

where $V^\pi(x)$ is known as the value function for policy $\pi(x, u)$.

A main objective of MDP is to determine a policy $\pi(x, u)$ to minimize the expected future cost,

$$\pi^*(x, u) = \arg \min_{\pi} V_k^\pi(x) = \arg \min_{\pi} E_\pi \left\{ \sum_{i=k}^{k+T} \gamma^{i-k} r_i | x_k = x \right\}. \quad (4)$$

This policy is termed as optimal policy, and the corresponding optimal value is given as,

$$V_k^*(x) = \min_{\pi} V_k^\pi(x) = \min_{\pi} E_\pi \left\{ \sum_{i=k}^{k+T} \gamma^{i-k} r_i | x_k = x \right\}. \quad (5)$$

In computational intelligence and economics, the interest is in utilities and rewards and the interest is in maximizing the expected performance index.

12.2.2. A Backward Recursion for the Value

By using the Chapman–Kolmogorov identity and the Markov property, we can write the value of policy $\pi(x, u)$ as,

$$V_k(x) = \sum_u \pi(x, u) \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma V_{k+1}^\pi(x')]. \quad (6)$$

This equation provides a backwards recursion for the value at time k in terms of the value at time $k + 1$.

12.2.3. Dynamic Programming

The optimal cost can be written as,

$$V_k^* = \min_{\pi} V_k^\pi(x) = \min_{\pi} \sum_u \pi(x, u) \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma V_{k+1}^\pi(x')]. \quad (7)$$

Bellman's optimality principle (Bellman, 1957) states that “An optimal policy has the property that no matter what the previous control actions have been, the remaining controls constitute an optimal policy with regard to the state resulting from those previous controls”. Therefore we can write,

$$V_k^* = \min_{\pi} V_k^\pi(x) = \min_{\pi} \sum_u \pi(x, u) \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma V_{k+1}^*(x')]. \quad (8)$$

Suppose an arbitrary control u is now applied at time k and the optimal policy is applied from time $k + 1$ on. Then Bellman's optimality principle says that the optimal control policy at time k is given by,

$$\pi^*(x, u) = \arg \min_{\pi} \sum_u \pi(x, u) \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma V_{k+1}^*(x')]. \quad (9)$$

Under the assumption that the Markov chain corresponding to each policy with transition probabilities given as in Equation (2) is ergodic, every MDP has a stationary deterministic optimal policy. Then we can equivalently minimize the conditional expectation over all actions u in state x . Therefore,

$$V_k^* = \min_u \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma V_{k+1}^*(x')], \quad (10)$$

$$u_k^* = \arg \min_u \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma V_{k+1}^*(x')]. \quad (11)$$

The backwards recursion, Equation (10), forms the basis for dynamic programming which gives offline methods for working backwards in time to determine optimal policies (Lewis *et al.*, 2012b). DP is an offline procedure (Howard, 1960) for finding the optimal value and optimal policies that requires knowledge of the complete system dynamics in the form of transition probabilities $P_{xx'}^u = Pr\{x'|x, u\}$ and expected costs, $R_{xx'}^u = E\{r_k|x_k = x, u_k = u, x_{k+1} = x'\}$.

12.2.4. Bellman Equation and Bellman Optimality Equation

Dynamic programming is a backwards in time method for finding the optimal value and policy. By contrast, reinforcement learning is concerned with finding optimal policies based on causal experience by executing sequential decisions that improve control actions based on the observed results of using a current policy. This procedure requires the derivation of methods for finding optimal values and optimal policies that can be executed forward in time. The key to this is the Bellman equation, which we now develop. References for this subsection include (Barto *et al.*, 1983; Busoniu *et al.*, 2009; Powell, 2011; Sutton and Barto, 1998). To derive forward-in-time methods for finding optimal values and optimal policies, set now the time horizon T to $+\infty$ and define the infinite-horizon cost,

$$J_k = \sum_{i=0}^{\infty} \gamma^i r_{k+i} = \sum_{i=k}^{\infty} \gamma^{i-k} r_i. \quad (12)$$

The associated infinite-horizon value function for *policy* $\pi(x, u)$ is,

$$V^\pi(x) = E_\pi \left\{ \gamma^{i-k} r_i | x_k = x \right\}. \quad (13)$$

By using (6) with $T = \infty$ it is seen that the value function for policy $\pi(x, u)$ satisfies the Bellman equation,

$$V^\pi(x) = \sum_u \pi(x, u) \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma V^\pi(x')]. \quad (14)$$

The key to deriving this equation is that the same value function appears on both sides, which is due to the fact that the infinite-horizon cost is used. Therefore, the Bellman

equation, Equation (14), can be interpreted as a consistency equation that must be satisfied by the value function at each time stage. It expresses a relation between the current value of being in state x and the value of being in next state x' given that policy $\pi(x, u)$ is used. The solution to the Bellman equation is the value given by the infinite sum in Equation (13).

The Bellman equation, Equation (14), is the starting point for developing a family of reinforcement learning algorithms for finding optimal policies by using causal experiences received stage-wise forward in time. The Bellman optimality equation, Equation (8), involves the minimum operator, and so does not contain any specific policy $\pi(x, u)$. Its solution relies on knowing the dynamics, in the form of transition probabilities. By contrast, the form of the Bellman equation is simpler than that of the optimality equation, and it is easier to solve. The solution to the Bellman equation yields the value function of a specific policy $\pi(x, u)$. As such, the Bellman equation is well suited to the actor-critic method of reinforcement learning shown in [Figure 12.1](#). It is shown subsequently that the Bellman equation provides methods for implementing the critic in [Figure 12.1](#), which is responsible for evaluating the performance of the specific current policy. Two key ingredients remain to be put in place. First, it is shown that methods known as policy iteration and value iteration use the Bellman equation to solve optimal control problems forward in time. Second, by approximating the value function in Equation (14) by a parametric structure, these methods can be implemented online using standard adaptive control system identification algorithms such as recursive least-squares.

In the context of using the Bellman equation, Equation (14), for reinforcement learning, $V^\pi(x)$ may be considered as a predicted performance $\sum_u \pi(x, u) \sum_{x'} P_{xx'}^u R_{xx'}^u$, the observed one step reward and $V^\pi(x')$ a current estimate of future behavior. Such notions can be capitalized on in the subsequent discussion of temporal different learning, which uses them to apply adaptive control algorithms that can learn optimal behavior online in real-time applications.

If the MDP is finite and has N states then the Bellman equation, Equation (14), is a system of N simultaneous linear equations for the value $V^\pi(x)$ of being in each state x given the current policy $\pi(x, u)$.

The optimal value satisfies,

$$V^*(x) = \min_\pi V^\pi(x) = \min_\pi \sum_u \pi(x, u) \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma V^\pi(x)], \quad (15)$$

while the Bellman's optimality principle gives the Bellman optimality equation,

$$V^*(x) = \min_\pi V^\pi(x) = \min_\pi \sum_u \pi(x, u) \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma V^*(x)], \quad (16)$$

Now under the assumption of ergodicity on the Markov chains corresponding to each

action/policy, the Bellman optimality equation, Equation (16), can be written as,

$$V^*(x) = \min_{\pi} V^\pi(x) = \min_{\pi} \sum_u \pi(x, u) \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma V^\pi(x')], \quad (17)$$

which is known as Hamilton–Jacobi–Bellman (HJB) equation in control systems. The optimal control is given by,

$$u^* = \arg \min_u \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma V^*(x')]. \quad (18)$$

Equations (17)–(18) can be written in the context of feedback control of dynamical systems. In the Linear Quadratic Regulator (LQR) case the Bellman equation, Equation (14), becomes a Lyapunov equation and the optimality equation, Equation (16), becomes the well known algebraic Riccati equation.

We shall see in the subsequent subsection two different algorithmic methods to solve the Bellman equation, Equation (14).

12.2.5. Policy Iteration and Value Iteration

Given a current policy $\pi(x, u)$, its value, Equation (13), can be determined by solving Equation (14). This procedure is known as policy evaluation. Moreover, given the value for some policy $\pi(x, u)$ we can always use it to find another policy that is at least no worse. This step is known as policy improvement. Specifically, suppose $V^\pi(x)$ satisfies Equation (14). Then define a new policy $\pi'(x, u)$ by

$$u^* = \arg \min_u \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma V^*(x')]. \quad (19)$$

and it can be shown that the value is monotonically decreasing (Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998). The policy, Equation (19), is said to be greedy with respect to the value function. In the special case that $V^{\pi'}(x) = V^\pi(x)$ in Equation (19), then the value and the policy satisfy Equations (17)–(18). In computational intelligence, greedy refers to quantities determined by optimizing over short or 1-step horizons, without regards to potential impacts far into the future. Now we will form the basis for the policy and value iteration algorithms in the form of two steps, a policy evaluation and a policy improvement. The policy evaluation step can be written as,

$$V^\pi(x) = \sum_u \pi(x, u) \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma V^\pi(x')], \quad \forall x \in S \subseteq X, \quad (20)$$

and the policy improvement step as,

$$\pi'(x, u) = \arg \min_{\pi'} \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma V^\pi(x')], \quad \forall x \in S \subseteq X. \quad (21)$$

Here, S is a suitably selected subspace of the state space, to be discussed later. We call an application of Equation (20) followed by an application of Equation (21) one step. This terminology is in contrast to the decision time stage k defined above. At each step of such algorithms, we obtain a policy that is no worse than the previous policy. Therefore, it is not difficult to prove convergence under fairly mild conditions to the optimal value and optimal policy. Most such proofs are based on the Banach Fixed Point Theorem. Note that Equation (17) is a fixed point equation for $V^*(\cdot)$. Then the two equations, Equations (20)–(21), define an associated map that can be shown under mild conditions to be a contraction map (Bertsekas and Tsitsiklis, 1996; Mehta and Meyn, 2009; Powell, 2011) which converges to the solution of Equation (17). A large family of algorithms is available that implement the policy evaluation and policy improvement procedures in different ways, or interleave them differently, or select subspace $S \subseteq X$ in different ways, to determine the optimal value and optimal policy. The relevance for feedback control systems of this discussion is that these two procedures can be implemented for dynamical systems online in real time by observing data measured along the system trajectories. The result is a family of adaptive control algorithms that converge to optimal control solutions. Such algorithms are of the actor-critic class of reinforcement learning systems, shown in [Figure 12.1](#). There, a critic agent evaluates the current control policy using methods based on Equation (20). After this evaluation is completed, the action is updated by an actor agent based on Equation (21).

One method of reinforcement learning for using Equations (20)–(21) to find the optimal value and policy is called policy iteration and is described next.

Algorithm 1: Policy Iteration

1: procedure

2: Given an initial admissible policy π_0

3: **while** $\|V^{\pi^{(j)}} - V^{\pi^{(j-1)}}\| \geq \epsilon_{ac}$ **do**

4: Solve for the value $V_j(x)$ using Bellman's equation

$$V_j(x) = \sum_u \pi_j(x, u) \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma V_j(x')], \quad \forall x \in X$$

5: Update the control policy $\pi_{(j+1)}$ using

$$\pi_{(j+1)}(x, u) = \arg \min_{\pi'} \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma V_j(x')], \quad \forall x \in X$$

6: $j := j + 1$

7: **end while**

8: end procedure

In Algorithm 1, ϵ_{ac} is a small number used to terminate the algorithm when two

consecutive value functions differ by less than ϵ_{ac} .

At each step j , the policy iteration algorithm determines the solution of the Bellman equation to compute the value $V_j(x)$ of using the current policy $\pi_j(x, u)$. This value corresponds to the infinite sum, Equation (13) for the current policy. Then the policy is improved. The steps are continued until there is no change in the value or the policy. Note that j is not the time or stage index k , but a policy iteration step iteration index. As detailed in the next sections, policy iteration can be implemented for dynamical systems online in real time by observing data measured along the system trajectories. Data for multiple times k are needed to solve the Bellman equation at each step j . The policy iteration algorithm must be suitably initialized to converge. The initial policy $\pi_0(x, u)$ and value V_0 must be selected so that $V_1 \leq V_0$. Then, for finite Markov chains with N states, policy iteration converges in a finite number of steps, less than or equal to N , because there are only a finite number of policies (Bertsekas and Tsitsiklis, 1996).

If the MDP is finite and has N states, then the policy evaluation equation is a system of N simultaneous linear equations, one for each state. Instead of directly solving the Bellman equation, it can be solved by an iterative policy evaluation procedure.

A second method for using Equations (20) and (21) in reinforcement learning is value iteration.

Algorithm 2: Value Iteration

1: procedure

2: Given an initial policy π_0

3: **while** $\|V^{\pi^j} - V^{\pi^{(j-1)}}\| \geq \epsilon_{ac}$ **do**

4: Solve for the value $V_j(x)$ using Bellman's equation

$$V_{j+1}(x) = \sum_u \pi_j(x, u) \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma V_j(x')], \quad \forall x \in S_j \subseteq X$$

5: Update the policy π_{j+1} using

$$\pi_{j+1}(x, u) = \arg \min_{\pi} \sum_{\pi'} P_{xx'}^u [R_{xx'}^u + \gamma V_j(x')], \quad \forall x \in S_j \subseteq X$$

6: $j := j + 1$

7: **end while**

8: end procedure

In Algorithm 2, ϵ_{ac} is a small number used to terminate the algorithm when two consecutive value functions differ by less than ϵ_{ac} .

In subsequent sections, we show how to implement value iteration for dynamical

systems online in real time by observing data measured along the system trajectories. Data for multiple times k are needed to solve the Bellman equation for each step j . Standard value iteration takes the update set as $S_j = X$, for all j . That is, the value and policy are updated for all states simultaneously. Asynchronous value iteration methods perform the updates on only a subset of the states at each step. In the extreme case, updates can be performed on only one state at each step.

It is shown in Bertsekas and Tsitsiklis (1996) that standard value iteration, which has, $S_j = X, \forall j$, converges for finite MDP for all initial conditions when the discount factor satisfies $0 < \gamma < 1$. When $S_j = X, \forall j$ and $\gamma = 1$, an absorbing state is added and a ‘properness’ assumption is needed to guarantee convergence to the optimal value. When a single state is selected for value and policy updates at each step, the algorithm converges, for all choices of initial value, to the optimal cost and policy if each state is selected for update infinitely often. More universal algorithms result if value update is performed multiple times for different choices of S_j prior to a policy improvement. Then, it is required that the updates are performed infinitely often for each state, and a monotonicity assumption must be satisfied by the initial starting value.

Considering Equation (16) as a fixed point equation, value iteration is based on the associated iterative map, which can be shown under certain conditions to be a contraction map.

In contrast to policy iteration, which converges under certain conditions in a finite number of steps, value iteration usually takes an infinite number of steps to converge (Bertsekas and Tsitsiklis, 1996). Consider finite MDP and consider the transition probability graph having probabilities (2) for the Markov chain corresponding to an optimal policy $\pi^*(x, u)$. If this graph is acyclic for some $\pi^*(x, u)$, then value iteration converges in at most N steps when initialized with a large value. Having in mind the dynamic programming equation, Equation (6), and examining the value iteration value update $V_j(x')$ can be interpreted as an approximation or estimate for the future stage cost-to-go from the future state x' . Those algorithms wherein the future cost estimate are themselves costs or values for some policy are called roll-out algorithms in Bertsekas and Tsitsiklis (1996). Such policies are forward looking and self-correcting. It is shown that these methods can be used to derive algorithms for receding horizon control (Zhang *et al.*, 2009). MDP, policy iteration, and value iteration are closely tied to optimal and adaptive control.

12.2.6. Generalized Policy Iteration

In policy iteration, the system of linear equations (see Algorithm 1) is completely solved at each step to compute the value Equation (13) of using the current policy $\pi_j(x, u)$. This solution can be accomplished by running iterations of the policy evaluation step until convergence. By contrast, in value iteration one takes only one iteration in the value

update step. Generalized policy iteration algorithms make several iterations in their value update step. Usually, policy iteration converges to the optimal value in fewer steps j since it does more work in solving equations at each step. On the other hand, value iteration is the easiest to implement as it only takes one iteration of a recursion. Generalized policy iteration provides a suitable compromise between computational complexity and convergence speed. Generalized policy iteration is a special case of the value iteration algorithm given above, where we select $S_j = X$, $\forall j$ and perform value update multiple times before each policy update.

12.2.7. Q-Learning

The conditional expected value in Equation (10) at iteration k is

$$\begin{aligned} Q_k^*(x, u) &= \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma V_{k+1}^*(x')] \\ &= E_\pi \{r_k + \gamma V_{k+1}^*(x') | x_k = x, u_k = u\}, \end{aligned} \quad (22)$$

and is known as optimal Q function (Watkins, 1989; Watkins and Dayan, 1992). In other words the Q function is equal to the expected return for taking an arbitrary action u at time k in state x and thereafter following an optimal policy. The Q function is a function of the current state x and the action u .

In terms of the Q function, the Bellman optimality equation has the particularly simple form,

$$V_k^*(x) = \min_u Q_k^*(x, u), \quad (23)$$

and

$$u_k^* = \arg \min_u Q_k^*(x, u), \quad (24)$$

Given some fixed policy $\pi(x, u)$ define the Q function for that policy as,

$$\begin{aligned} Q_k^\pi(x, u) &= E_\pi \{r_k + \gamma V_{k+1}^\pi(x') | x_k = x, u_k = u\} \\ &= \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma V_{k+1}^\pi(x')], \end{aligned} \quad (25)$$

where this function is equal to the expected return for taking an arbitrary action u at time k in state x and thereafter following the existing policy $\pi(x, u)$.

Note that $V_k^\pi(x) = Q_k^\pi(x, \pi(x, u))$ hence Equation (25) can be written as the backwards recursion in the Q function,

$$Q_k^\pi(x, u) = \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma Q_{k+1}^\pi(x', \pi(x', u'))]. \quad (26)$$

The Q function is a two-dimensional function of both the current state x and the action u . By contrast, the value function is a one-dimensional function of the state. For finite MDP, the Q function can be stored as a 2D lookup table at each state/action pair. Note that direct minimization in Equations (8)–(9) requires knowledge of the state transition probabilities, which correspond to the system dynamics, and costs. By contrast, the minimization in Equations (23)–(24) requires knowledge only of the Q function and not the system dynamics. The utility of the Q function is twofold. First, it contains information about control actions in every state. As such, the best control in each state can be selected using Equation (24) by knowing only the Q function. Second, the Q function can be estimated online in real time directly from date observed along the system trajectories, without knowing the system dynamics information, that is, the transition probabilities. We shall see how this is accomplished later. The infinite horizon Q function for a prescribed fixed policy is given by,

$$Q^\pi(x, u) = \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma V^\pi(x')] \quad (27)$$

which also satisfies the Bellman equation. Note that for a fixed policy $\pi(x, u)$

$$V^\pi(x) = Q^\pi(x, \pi(x, u)), \quad (28)$$

whence according to Equation (27) the Q function satisfies the Bellman equation,

$$Q^\pi(x, u) = \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma Q^\pi(x', \pi(x', u'))]. \quad (29)$$

The Bellman optimality equation for the Q function is given by

$$Q^*(x, u) = \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma \min_{u'} Q^*(x', u')]. \quad (30)$$

It is interesting to compare Equations (17) and (30) where the minimum operator and the expected value operator are reversed.

The algorithms defined in the previous subsections, namely, policy iteration and value iteration are really easy to implement in terms of the Q function, Equation (25) as follows,

Algorithm 3: Policy Iteration Using Q Function

1: procedure

2: Given an initial admissible policy π_0

3: **while** $\|Q^{\pi^j} - Q^{\pi^{(j-1)}}\| \geq \epsilon_{ac}$ **do**

4: Solve for $Q_j(x)$ using

$$Q_j(x) = \sum_u \pi_j(x, u) \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma Q_j(x', \pi(x', u'))], \forall x \in X$$

```

5:      Update the policy  $\pi_{j+1}$  using
         
$$\pi_{j+1}(x, u) = \arg \min_{\pi} Q_j(x, u), \forall x \in X$$

6:       $j := j + 1$ 
7:  end while
8: end procedure

```

Algorithm 4: Value Iteration Using Q Function

```

1: procedure
2:   Given an initial policy  $\pi_0$ 
3:   while  $\|Q^{\pi^j} - Q^{\pi^{(j-1)}}\| \geq \epsilon_{ac}$  do
4:     Solve for the value  $Q_j(x)$  using Bellman's equation
         
$$Q_{j+1}(x) = \sum_u \pi_j(x, u) \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma Q_j(x', \pi(x', u'))], \forall x \in S_j \subseteq X$$

5:     Update the policy  $\pi_{j+1}$  using
         
$$\pi_{j+1}(x, u) = \arg \min_{\pi} Q_j(x, u), \forall x \in S_j \subseteq X$$

6:      $j := j + 1$ 
7:   end while
8: end procedure

```

In Algorithms 3 and 4, ϵ_{ac} is a small number used to terminate the algorithm when two consecutive Q functions differ by less than ϵ_{ac} .

As we show, the utility of the Q function is that these algorithms can be implemented online in real-time, without knowing the system dynamics, by measuring data along the system trajectories. They yield optimal adaptive control algorithms, that is, adaptive control algorithms that converge online to optimal control solutions.

12.3. Methods for Implementing Policy Iteration and Value Iteration

Different methods are available for performing the value and policy updates for policy iteration and value iteration (Bertsekas and Tsitsiklis, 1996; Powell, 2011; Sutton and Barto, 1998) The main three are exact computation, Monte Carlo methods, and temporal difference learning. The last two methods can be implemented without the knowledge of the system dynamics. Temporal difference learning is the means by which optimal adaptive control algorithms can be derived for dynamical systems.

Policy iteration requires solution at each step of the Bellman equation for the value update. For a finite MDP with N states, this is a set of linear equations in N unknowns, namely, the values of each state. Value iteration requires performing the one-step recursive update at each step for the value update. Both of these can be accomplished exactly if we know the transition probabilities $P_{x,x'}^u$ and costs $R_{xx'}^u$ of the MDP, which corresponds to knowing full system dynamics information. Likewise, the policy improvements can be explicitly computed if the dynamics are known.

Monte Carlo learning is based on the definition, Equation (13), for the value function, and uses repeated measurements of data to approximate the expected value. The expected values are approximated by averaging repeated results along sample paths. An assumption on the ergodicity of the Markov chain with transition probabilities, Equation (2), for the given policy being evaluated is implicit. This assumption is suitable for episodic tasks, with experience divided into episodes (Sutton and Barto, 1998), namely, processes that start in an initial state and run until termination, and are then restarted at a new initial state. For finite MDP, Monte Carlo methods converge to the true value function if all states are visited infinitely often. Therefore, in order to ensure accurate approximations of value functions, the episode sample paths must go through all the states $x \in X$ many times. This issue is called the problem of maintaining exploration. Several methods are available to ensure this, one of which is to use ‘exploring starts’, in which every state has non-zero probability of being selected as the initial state of an episode. Monte Carlo techniques are useful for dynamic systems control because the episode sample paths can be interpreted as system trajectories beginning in a prescribed initial state. However, no updates to the value function estimate or the control policy are made until after an episode terminates. In fact, Monte Carlo learning methods are closely related to repetitive or iterative learning control (Moore, 1993). They do not learn in real time along a trajectory, but learn as trajectories are repeated.

12.4. Temporal Difference Learning and its Application to Feedback Control

It is now shown that the temporal difference method (Sutton and Barto, 1998) for solving Bellman equations leads to a family of optimal adaptive controllers, that is, adaptive controllers that learn online the solutions to optimal control problems without knowing the full system dynamics. Temporal difference learning is true online reinforcement learning, wherein control actions are improved in real time based on estimating their value functions by observing data measured along the system trajectories.

Policy iteration requires solution at each step of N linear equations. Value iteration requires performing a recursion at each step. Temporal difference reinforcement learning methods are based on the Bellman equation without using any systems dynamics knowledge, but using data observed along a single trajectory of the system. Therefore, temporal difference learning is applicable for feedback control applications. Temporal difference updates the value at each time step as observations of data are made along a trajectory. Periodically, the new value is used to update the policy. Temporal difference methods are related to adaptive control in that they adjust values and actions online in real time along system trajectories. Temporal difference methods can be considered to be stochastic approximation techniques whereby the Bellman equation, Equation (14), is replaced by its evaluation along a single sample path of the MDP. Then, the Bellman equation becomes a deterministic equation that allows the definition of a temporal difference error. Equation (6) is used to write the Bellman equation, Equation (14) for the infinite-horizon value, Equation (13). An alternative form for the Bellman equation is,

$$V^\pi(x_k) = E_\pi\{r_k|x_k\} + \gamma E_\pi\{V_\pi(x_{k+1})|x_k\}, \quad (31)$$

which is the basis for temporal difference learning.

Temporal difference reinforcement learning uses one sample path, namely the current system trajectory, to update the value. Then, Equation (31) is replaced by the deterministic Bellman equation,

$$V^\pi(x_k) = r_k + \gamma V_\pi(x_{k+1}), \quad (32)$$

which holds for each observed data experience set (x_k, x_{k+1}, r_k) at each time step k . This dataset consists of the current state x_k the observed cost r_k and the next state x_{k+1} . Hence, we can define the temporal difference error as

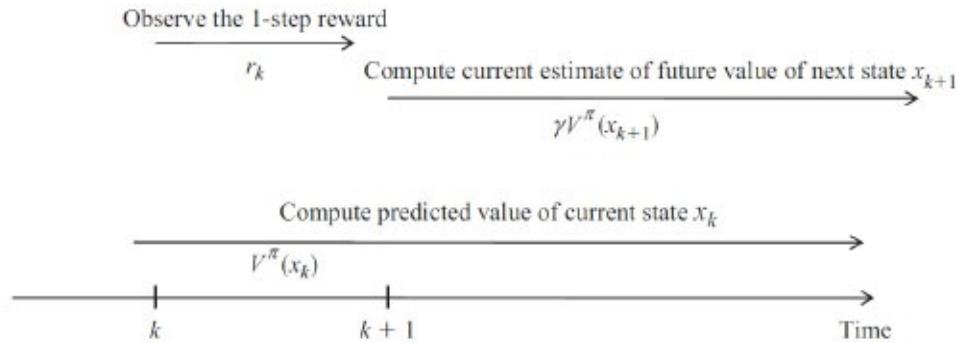
$$e_k = -V^\pi(x_k) + r_k + \gamma V^\pi(x_{k+1}), \quad (33)$$

and the value estimate is updated to make the temporal difference error small.

In the context of temporal difference learning, the interpretation of the Bellman

equation is shown in [Figure 12.2](#), where $V^\pi(x_k)$ may be considered as a predicted performance or value, r_k as the observed one step reward, and $\gamma V^\pi(x_{k+1})$ as a current estimate of future value. The Bellman equation can be interpreted as a consistency equation that holds if the current estimate for the predicted value $V^\pi(x_k)$ is correct. Temporal difference methods update the predicted value estimate $\hat{V}^\pi(x_k)$ to make the temporal difference error small. The idea, based on stochastic approximation, is that if we use the deterministic version of Bellman's equation repeatedly in policy iteration or value iteration, then on average these algorithms converge towards the solution of the stochastic Bellman equation.

1. Apply control action



2. Update predicted value to satisfy the Bellman equation

$$V^\pi(x_k) = r_k + \gamma V^\pi(x_{k+1})$$

3. Improve control action

Figure 12.2: Temporal difference interpretation of Bellman equation. Shows how use of the Bellman equation captures the action, observation, evaluation, and improvement mechanisms of reinforcement learning.

12.5. Optimal Adaptive Control for Discrete-Time Deterministic Systems

A family of optimal adaptive control algorithms can now be developed for dynamical systems. These algorithms determine the solutions to Hamilton–Jacobi (HJ) design equations online in real-time without knowing the system drift dynamics. In the LQR case, this means they solve the Riccati equation online without knowing the system A matrix. Physical analysis of dynamical systems using Lagrangian mechanics or Hamiltonian mechanics produces system descriptions in terms of nonlinear ordinary differential equations. Discretization yields nonlinear difference equations. Most research in reinforcement learning is conducted for systems that operate in discrete time (Sutton and Barto, 1998; Werbos, 1992; Wheeler and Narendra, 1986). Therefore, we cover discrete-time dynamical systems first, then continuous time systems.

Temporal difference learning is a stochastic approximation technique based on the deterministic Bellmans equation, Equation (32). Therefore, we lose little by considering deterministic systems here. Therefore, consider a class of discrete-time systems described by deterministic nonlinear dynamics in the affine state space difference equation form,

$$x_{k+1} = f(x_k) + g(x_k)u_k, \quad k = 0, 1, \dots, \quad (34)$$

with state $x_k \in \mathbb{R}^n$ and control input $u_k \in \mathbb{R}^m$. We use this form because its analysis is convenient. The following development can be generalized to the sampled-data form $x_{k+1} = F(x_k, u_k)$.

A deterministic control policy is defined as a function from state space to control space $h(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^m$. That is, for every state x_k , the policy defines a control action,

$$u_k = h(x_k), \quad (35)$$

which is a feedback controller.

The value function with a deterministic cost function can be written as,

$$V^h(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, u_i) = \sum_{i=k}^{\infty} \gamma^{i-k} (Q(x_i) + u_i^T R u_i), \quad (36)$$

where $0 < \gamma \leq 1$ is a discount factor, $Q(x_k) > 0$, $R > 0$ and $u_k = h(x_k)$ is a prescribed feedback control policy. That is, the stage cost is

$$r(x_k, u_k) = Q(x_k) + u_k^T Q u_k, \quad (37)$$

which is taken as quadratic in u_k to simplify the developments but can be any positive definite function of the control. We assume that the system is stabilizable on some set $\Omega \in \mathbb{R}^n$, that is there exists a control policy $u_k = h(x_k)$ such that the closed-loop system $x_{k+1} = f(x_k) + g(x_k)h(x_k)$ is asymptotically stable on Ω .

For the deterministic value (36), the optimal value is given by Bellman's optimality equation,

$$v^*(x_k) = \min_{h(\cdot)} (r(x_k, h(x_k)) + \gamma V^*(x_{k+1})), \quad (38)$$

which is just the discrete time Hamilton–Jacobi–Bellman (HJB) equation. The optimal policy is then given as,

$$h^*(x_k) = \arg \min_{h(\cdot)} (r(x_k, h(x_k)) + \gamma V^*(x_{k+1})). \quad (39)$$

In this setup, the deterministic Bellman's equation, Equation (32) is,

$$\begin{aligned} V^h(x_k) &= r(x_k, u_k) + \gamma V^h(x_{k+1}) \\ &= Q(x_k) + \mu_k^T R u_k + \gamma V^h(x_{k+1}), \quad V^h(0) = 0, \end{aligned} \quad (40)$$

which is a difference equation equivalent of the value, Equation (36). That is, instead of evaluating the infinite sum, Equation (36), the difference equation, Equation (40), can be solved with boundary condition $V(0) = 0$ to obtain the value of using a current policy $u_k = h(x_k)$.

The discrete-time Hamiltonian function can be defined as,

$$H(x_k, h(x_k), \Delta V_k) = r(x_k, h(x_k)) + \gamma V^h(x_{k+1}) - V^h(x_k), \quad (41)$$

where $\Delta V_k \equiv \gamma V^h(x_{k+1}) - V^h(x_k)$ is the forward difference operator. The Hamiltonian function captures the energy content along the trajectories of a system as reflected in the desired optimal performance. In fact, the Hamiltonian is the temporal difference error, Equation (33). The Bellman equation requires that the Hamiltonian be equal to zero for the value associated with a prescribed policy.

For the discrete-time linear quadratic regulator case we have,

$$x_{k+1} = Ax_k + Bu_k, \quad (42)$$

$$V^h(x_k) = \frac{1}{2} \sum_{i=k}^{\infty} \gamma^{i-k} (x_i^T Q x_i + u_i^T R u_i) \quad (43)$$

and the Bellman equation can be easily derived.

12.5.1. Policy Iteration and Value Iteration for Discrete-Time Deterministic Dynamical Systems

Two forms of reinforcement learning can be based on policy iteration and value iteration.

Algorithm 5: Policy Iteration Using Temporal Difference Learning

1: procedure

- 2: Given an initial admissible policy $h_0(x_k)$
 3: **while** $\|V^{h^j} - V^{h^{(j-1)}}\| \geq \epsilon_{ac}$ **do**
 4: Solve for $V_j(x_k)$ using

$$V_{j+1}(x_k) = r(x_k, h_j(x_k)) + \gamma V_j(x_{k+1})$$

 5: Update the policy h_{j+1} using

$$h_{j+1}(x_k) = -\frac{\gamma}{2} R^{-1} g^T(x_k) \nabla V_{j+1}(x_{k+1}),$$

 6: $j := j + 1$
 7: **end while**
 8: **end procedure**
-

Algorithm 6: Value Iteration Using Temporal Difference Learning**1: procedure**

- 2: Given an initial policy $h_0(x_k)$
 3: **while** $\|V^{h^j} - V^{h^{(j-1)}}\| \geq \epsilon_{ac}$ **do**
 4: Solve for $V_j(x_k)$ using

$$V_{j+1}(x_k) = r(x_k, h_j(x_k)) + \gamma V_j(x_{k+1})$$

 5: Update the policy h_{j+1} using

$$h_{j+1}(x_k) = -\frac{\gamma}{2} R^{-1} g^T(x_k) \nabla V_{j+1}(x_{k+1}),$$

 6: $j := j + 1$
 7: **end while**
 8: **end procedure**
-

In Algorithms 5 and 6, ϵ_{ac} is a small number used to terminate the algorithm when two consecutive value functions differ by less than ϵ_{ac} and $\nabla V(x) = \frac{\partial V(x)}{\partial x}$. Note that in value iteration we can select any initial control policy, not necessarily admissible or stabilizing. The policy iteration and value iteration algorithms just described are offline design methods that require knowledge of the discrete-time dynamics. By contrast, we next desire to determine online methods for implementing policy iteration and value iteration that do not require full dynamics information.

Policy iteration and value iteration can be implemented for finite MDP by storing and updating lookup tables. The key to implementing policy iteration and value iteration online for dynamical systems with infinite state and action spaces is to approximate the

value function by a suitable approximator structure in terms of unknown parameters. Then, the unknown parameters are tuned online exactly as in system identification. This idea of value function approximation (VFA) is used by Werbos (1989, 1992), and called approximate dynamic programming (ADP) or adaptive dynamic programming. The approach is used by Bertsekas and Tsitsiklis (1996) and called neuro-dynamic programming (Bertsekas and Tsitsiklis, 1996; Busoniu *et al.*, 2009; Powell, 2011). For nonlinear systems, Equation (34), the value function contains higher order nonlinearities. Then, we assume the Bellman equation, Equation (40), has a local smooth solution (Van der Schaft, 1992). Then, according to the Weierstrass Higher Order Approximation Theorem, there exists a dense basis set $\phi_i(x)$ such that,

$$V(x) = W^T \phi(x) + \epsilon_L(x), \quad (44)$$

where $\phi(x) = [\phi_1(x) \phi_2(x) \dots \phi_L(x)] : \mathbb{R}^n \rightarrow \mathbb{R}^L$ and $\epsilon_L(x)$ converges uniformly to zero as the basis sets $L \rightarrow \infty$. Note that in the LQR case the weight vector consists of the elements of the symmetric Riccati matrix P . We are now in a position to present several adaptive control algorithms based on temporal difference reinforcement learning that converge online to the optimal control solution.

Algorithm 7: Optimal Adaptive Control Using Policy Iteration

- 1: **procedure**
 - 2: Given an initial admissible policy $h_0(x_k)$
 - 3: **while** No convergence **do**
 - 4: Determine the least squares solution W_{j+1} using

$$W_{j+1}^T(\phi(x_k) - \gamma\phi(x_{k+1})) = r(x_k, h_j(x_k))$$
 - 5: Update the policy h_{j+1} using

$$h_{j+1}(x_k) = -\frac{\gamma}{2} R^{-1} g^T(x_k) \nabla \phi^T(x_{k+1}) W_{j+1},$$
 - 6: $j := j + 1$
 - 7: **end while**
 - 8: **end procedure**
-

Algorithm 8: Optimal Adaptive Control Using Value Iteration

- 1: **procedure**
- 2: Given an initial policy $h_0(x_k)$
- 3: **while** No convergence **do**
- 4: Determine the least squares solution W_{j+1} using

$$W_{j+1}^T \phi(x_k) = r(x_k, h_j(x_k)) + \gamma W_j^T \phi(x_{k+1})$$

5: Update the policy h_{j+1} using

$$h_{j+1}(x_k) = -\frac{\gamma}{2} R^{-1} g^T(x_k) \nabla \phi^T(x_{k+1}) W_{j+1},$$

6: $j := j + 1$

7: **end while**

8: **end procedure**

12.5.2. Introducing a Second ‘Actor’ Neural Network

Using value function approximation (VFA) allows standard system identification techniques to be used to find the value function parameters that approximately solve the Bellman equation. The approximator structure just described that is used for approximation of the value function is known as the critic neural network, as it determines the value of using the current policy. Using VFA, the policy iteration reinforcement learning algorithm solves a Bellman equation during the value update portion of each iteration step j by observing only the dataset $(x_k, x_{k+1}, r(x_k, h_j(x_k)))$ at each time along the system trajectory. In the case of value iteration, VFA is used to perform a value update.

The critic network solves the Bellman equation using observed data without knowing the system dynamics. However, note that in the Linear Quadratic Case (systems of the form $x_{k+1} = Ax_k + Bu_k$ with quadratic costs) case the policy update is given by

$$K^{j+1} = -(B^T P^{j+1} B + R)^{-1} B^T P^{j+1} A, \quad (45)$$

which requires full knowledge of the dynamics.

This problem can be solved by introducing a second neural network for the control policy, known as the actor neural network (Al-Tamimi *et al.*, 2008; Vrabie and Lewis, 2009; Werbos, 1989). Therefore, consider a parametric approximator structure for the control action,

$$u_k = h(x_k) = U^T \sigma(x_k) \quad (46)$$

with $\sigma(x) : \text{IR}^n \rightarrow \text{IR}^m$ a vector of M activation functions and $U \in \text{IR}^{M \times m}$ a matrix of weights or unknown parameters. Note that in the LQR case the basis set can be taken as the state vector. After convergence of the critic neural network parameters to W_{j+1} in policy iteration or value iteration, it is required to perform the policy update. To achieve this aim, we can use a gradient descent method for tuning the actor weights U such as,

$$U_{j+1}^{i+1} = U_{j+1}^i - \beta \sigma(x_k) \left(2R(U_{j+1}^i)^T \sigma(x_k) + \gamma g^T(x_k) \nabla \phi^T(x_{k+1}) W_{j+1} \right)^T, \quad (47)$$

where $\beta \in \text{IR}^+$ is a tuning parameter. The tuning index i can be incremented with the time index k .

Note that the tuning of the actor neural network requires observations at each time k of the dataset x_k, x_{k+1} , that is, the current state and the next state. However, as per the formulation Equation (46), the actor neural network yields the control u_k at time k in terms of the state x_k at time k . The next state x_{k+1} is not needed in Equation (46). Thus, after Equation (47) converges, Equation (46) is a legitimate feedback controller. Note also that, in the LQR case, the actor neural network Equation (46) embodies the feedback gain computation Equation (45). Equation (45) contains the state internal dynamics A , but Equation (46) does not. Therefore, the A matrix is not needed to compute the feedback control. The reason is that the actor neural network learns information about A in its weights, since x_k, x_{k+1} are used in its tuning. Finally, note that only the input function $g(\cdot)$ or, in the LQR case, the B matrix, is needed in Equation (47) to tune the actor neural network. Thus, introducing a second actor neural network completely avoids the need for knowledge of the state drift dynamics $f(\cdot)$, or A in the LQR case.

12.6. Actor/Critic Structures for Optimal Adaptive Control

The implementation of reinforcement learning using two neural networks, one as a critic and one as an actor, yields the actor/critic reinforcement learning structure shown in [Figure 12.1](#). In this control system, the critic and the actor are tuned online using the observed data $(x_k, x_{k+1}, r(x_k, h_j(x_k)))$ along the system trajectory. The critic and actor are tuned sequentially in both the policy iteration and the value iteration algorithms. That is, the weights of one neural network are held constant while the weights of the other are tuned until convergence. This procedure is repeated until both neural networks have converged. Thus, the controller learns the optimal controller online. This procedure amounts to an online adaptive optimal control system wherein the value function parameters are tuned online and the convergence is to the optimal value and control. The convergence of value iteration using two neural networks for the discrete-time nonlinear system [Equation \(34\)](#) is proven in (Al-Tamimi *et al.*, 2008).

According to reinforcement learning principles, the optimal adaptive control structure requires two loops, a critic and an actor, and operates at multiple timescales. A fast loop implements the control action inner loop, a slower timescale operates in the critic loop, and a third timescale operates to update the policy. Several actor/critic structures for optimal adaptive control have been proposed in Ferrari and Stengel (2002); Prokhorov *et al.* (1995) and Hanselmann *et al.* (2007).

12.6.1. Q-learning for Optimal Adaptive Control

It has just been seen how to implement an optimal adaptive controller using reinforcement learning that only requires knowledge of the system input function $g(x_k)$. The Q learning reinforcement learning method gives an adaptive control algorithm that converges online to the optimal control solution for completely unknown systems. That is, it solves the Bellman equation, [Equation \(40\)](#), and the HJB equation, [Equation \(38\)](#), online in real time by using data measured along the system trajectories, without any knowledge of the dynamics $f(x_k)$, $g(x_k)$. Writing the Q function Bellman equation, [Equation \(29\)](#) along a sample path gives,

$$Q^\pi(x_k, u_k) = r(x_k, u_k) + \gamma Q^\pi(x_{k+1}, h(x_{k+1})), \quad (48)$$

which defines a temporal difference error,

$$e_k = -Q^\pi(x_k, u_k) + r(x_k, u_k) + \gamma Q^\pi(x_{k+1}, h(x_{k+1})). \quad (49)$$

The Q function is updated using the algorithm,

$$\begin{aligned} Q_k(x_k, u_k) &= Q_{k-1}(x_k, u_k) \\ &+ \alpha_k [r(x_k, u_k) + \gamma \min_u Q_{k-1}(x_{k+1}, u_k) - Q_{k-1}(x_k, u_k)], \end{aligned} \quad (50)$$

with α_k a tuning parameter. This algorithm is developed for finite MDP and the convergence proven by Watkins (1989) using Stochastic Approximation (SA) methods. The Q learning algorithm Equation (50) is similar to SA methods of adaptive control or parameter estimation used in control systems. Let us now derive methods for Q learning for dynamical systems that yield adaptive control algorithms that converge to optimal control solutions. Policy iteration and value iteration algorithms can be given using the Q function. A Q learning algorithm is easily developed for discrete-time dynamical systems using Q function approximation (Bradtko *et al.*, 1994; Werbos, 1989, 1991, 1992). Assume therefore that for nonlinear systems the Q function is parameterized as,

$$Q(x, u) = W^T \phi(z), \quad (51)$$

where $z_k \equiv [x_k^T \ u_k^T]^T$, W is the unknown parameter vector and $\phi(z)$ is the basis set vector. Substituting the Q-function approximation into the temporal difference error Equation (49) yields,

$$e_k = -W^T \phi(z_k) + r(x_k, u_k) + \gamma W^T \phi(z_{k+1}), \quad (52)$$

on which either policy iteration or value iteration algorithms can be based. Considering the policy iteration algorithm defined before the Q-function evaluation step is,

$$W_{j+1}^T (\phi(z_k) - \gamma \phi(z_{k+1})) = r(x_k, h_j(x_k)), \quad (53)$$

and the policy improvement step,

$$h_{j+1}(x_k) = \arg \min_u (W_{j+1}^T \phi(x_k, u)), \quad \forall x \in X. \quad (54)$$

Now using value iteration, the Q learning is given by,

$$W_{j+1}^T \phi(z_k) = r(x_k, h_j(x_k)) + \gamma W_j^T \phi(z_{k+1}), \quad (55)$$

and Equation (54). Note that these equations do not require knowledge of the dynamics.

12.6.2. Approximate Dynamic Programming Using Output feedback

The methods discussed have relied on full state variable feedback. Little work has been done in applications of reinforcement learning for feedback control using output feedback. This corresponds to partially observable Markov processes. Design of an ADP controller that uses only output feedback is given in Lewis and Vamvoudakis (2011).

12.7. Integral Reinforcement Learning for Optimal Adaptive Control of Deterministic Continuous-Time Systems

Reinforcement learning is considerably more difficult for continuous-time systems (Doya, 2000) than for discrete-time systems, and fewer results are available. See Abu-Khalaf *et al.* (2006) for the development of an offline policy iteration method for continuous-time systems. Using a method known as integral reinforcement learning (IRL) (Vrabie and Lewis, 2009; Vrabie *et al.*, 2009; Vrabie and Vamvoudakis, 2012; Zhang *et al.*, 2013) allows the application of reinforcement learning to formulate online optimal adaptive control methods for continuous-time systems. These methods find solutions to optimal HJ design equations and Riccati equations online in real-time without knowing the system drift dynamics $f(x)$, or in the LQR case without knowing the A matrix.

Consider the continuous-time nonlinear dynamical system,

$$\dot{x} = f(x) + g(x)u, \quad (56)$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the control input. Also consider as $x = 0$ an equilibrium point e.g., $f(0) = 0$ and $f(x) + g(x)u$ Lipschitz on a set $\Omega \subseteq \mathbb{R}^n$ that contains the origin. We assume the system is stabilizable on Ω , that is, there exists a continuous control input function $u(t)$ such that the closed-loop system is asymptotically stable on Ω .

Define a performance measure or cost function that has the value associated with the feedback control policy $u = \mu(x)$ given by,

$$V^\mu(x(t)) = \int_t^\infty r(x(\tau), u(\tau)) d\tau, \quad (57)$$

where $r(x, u) = Q(x) + u^T R u$ the utility function, $Q(x) > 0$, $\forall x$ and $x = 0 \rightarrow Q(x) = 0$, and $R = R^T$ positive definite matrix.

For the continuous-time LQR, we have,

$$\dot{x} = Ax + Bu, \quad (58)$$

$$V^\mu(x(t)) = \frac{1}{2} \int_t^\infty (x^T Q x + u^T R u) d\tau. \quad (59)$$

If the cost is smooth, then an infinitesimal equivalent to Equation (57) can be found by Leibniz's formula to be,

$$0 = r(x, \mu(x)) + (\nabla V^\mu)^T (f(x) + g(x)\mu(x)), \quad V^\mu(0) = 0, \quad (60)$$

where ∇V^μ denotes the column gradient vector with respect to x .

This is the continuous-time Bellman equation. This equation is defined based on the continuous-time Hamiltonian function,

$$H(x, \mu(x), \nabla V^\mu) = r(x, \mu(x)) + (\nabla V^\mu)^T(f(x) + g(x)\mu(x)) \quad (61)$$

and the HJB equation is given by

$$0 = \min_{\mu} H(x, \mu(x), \nabla V^*) \quad (62)$$

with an optimal control,

$$\mu^* = \arg \min_{\mu} H(x, \mu(x), \nabla V^*). \quad (63)$$

We now see the problem with continuous-time systems immediately. Comparing the continuous-time Hamiltonian Equation (61) to the discrete-time Hamiltonian equation (41), it is seen that Equation (61) contains the full system dynamics $f(x) + g(x)u$, while Equation (41) does not. What this means is that the continuous-time Bellman equation, Equation (86) cannot be used as a basis for reinforcement learning unless the full dynamics are known. Reinforcement learning methods based on Equation (60) can be developed (Mehta and Meyn, 2009). These have limited use for adaptive control purposes because the system dynamics must be known, state derivatives must be measured, or integration over an infinite horizon is required. In another approach, Euler's method can be used to discretize the continuous-time Bellman equation, Equation (60) [see (Baird, 1994)]. Noting that

$$0 = r(x, \mu(x)) + (\nabla V^\mu)^T(f(x) + g(x)\mu(x)) = r(x, \mu(x)) + \dot{V}^\mu, \quad (64)$$

we use Euler's forward method to discretize this partial differential equation (PDE) to obtain,

$$0 = r(x, \mu(x)) + \frac{V^\mu(x_{k+1}) - V^\mu(x_k)}{T} \equiv \frac{r_S(x_k, u_k)}{T} + \frac{V^\mu(x_{k+1}) - V^\mu(x_k)}{T}, \quad (65)$$

where $T > 0$ is a sample period i.e., $t = kT$ and $r_S(x_k, u_k) = Tr(x_k, u_k)$.

Now note that the discretized continuous-time Bellman equation, Equation (65) has the same form as the discrete-time Bellman equation, Equation (40). Therefore, all the reinforcement learning methods just described for discrete-time systems can be applied.

However, this equation is an approximation only. An alternative exact method for continuous-time reinforcement learning is given in Vrabie and Lewis (2009); Vrabie *et al.* (2009). That method is termed integral reinforcement learning (IRL). Note that the cost Equation (57) can be written in the integral reinforcement form,

$$V^\mu(x(t)) = \int_t^{t+T} r(x(\tau), u(\tau)) d\tau + V^\mu(x(t+T)), \quad (66)$$

for some $T > 0$. This equation is exactly in the form of the discrete-time Bellman equation, Equation (40). According to Bellmans principle, the optimal value is given in terms of this construction as (Lewis *et al.*, 2012b),

$$V^*(x(t)) = \min_{\bar{u}(t:t+T)} \left(\int_t^{t+T} r(x(\tau), u(\tau)) d\tau + V^*(x(t+T)) \right)$$

where $\bar{u}(t : t+T) = \{u(\tau) : t \leq \tau < t+T\}$ and the optimal control is given by,

$$\mu^*(x(t)) = \arg \min_{\bar{u}(t:t+T)} \left(\int_t^{t+T} r(x(\tau), u(\tau)) d\tau + V^*(x(t+T)) \right).$$

It is shown in Vrabie *et al.* (2009) that the Bellman equation, Equation (60), is equivalent to the integral reinforcement form Equation (66). That is, the positive definite solution of both is the value Equation (57) of the policy $u = \mu(x)$. The integral reinforcement form Equation (66) serves as a Bellman equation for continuous-time systems, and is a fixed point equation. Therefore, the temporal difference error for continuous-time systems can be defined as,

$$e(t : t+T) = \int_t^{\infty} r(x(\tau), u(\tau)) d\tau + V^\mu(x(t+T)) - V^\mu(x(t)), \quad (67)$$

which is an equation that does not involve the system dynamics. Now, it is direct to formulate policy iteration and value iteration for continuous-time systems. The following algorithms are termed integral reinforcement learning for continuous time systems in Vrabie and Lewis (2009); Vrabie *et al.* (2009). Both algorithms give optimal adaptive controllers for continuous time systems, that is, adaptive control algorithms that converge to optimal control solutions.

Algorithm 9: IRL Optimal Adaptive Control Using Policy Iteration

1: procedure

2: Given an initial admissible policy $\mu_0(x)$

3: **while** $\|V^{\mu^j} - V^{\mu^{(j-1)}}\| \geq \epsilon_{ac}$ **do**

4: Solve for V_j using

$$V_{j+1}(x(t)) = \int_t^{t+T} r(x(s), \mu_j(x(s))) ds + V_{j+1}(x(t+T)), \quad V_{j+1}(0) = 0$$

5: Update the policy μ_{j+1} using

$$\mu_{j+1}(x_k) = -\frac{1}{2}R^{-1}g^T(x)\nabla V_{j+1},$$

6: $j := j + 1$

7: **end while**

8: **end procedure**

Algorithm 10: IRL Optimal Adaptive Control Using Value Iteration

1: **procedure**

2: Given an initial policy $\mu_0(x)$

3: **while** $\|V^{\mu^j} - V^{\mu^{(j-1)}}\| \geq \epsilon_{ac}$ **do**

4: Solve for V_j using

$$V_{j+1}(x(t)) = \int_t^{t+T} r(x(s), \mu_j(x(s)))ds + V_j(x(t+T))$$

5: Update the policy μ_{j+1} using

$$\mu_{j+1}(x) = -\frac{1}{2}R^{-1}g^T(x)\nabla V_{j+1},$$

6: $j := j + 1$

7: **end while**

8: **end procedure**

In Algorithms 9 and 10, ϵ_{ac} is a small number used to terminate the algorithm when two consecutive value functions differ by less than ϵ_{ac} and note that neither algorithm requires knowledge about the system drift dynamics function $f(x)$. That is, they work for partially unknown systems. Convergence of IRL policy iteration is proved in Vrabie *et al.* (2009). An IRL algorithm for nonlinear systems has been proposed in Liu *et al.* (2013).

12.7.1. Online Implementation of IRL — a Hybrid Optimal Adaptive Controller

Both of these IRL algorithms can be implemented online by reinforcement learning techniques using value function approximation $V(x) = W^T \phi(x)$ in a critic approximator network. Using VFA in the policy iteration algorithm just defined yields

$$W_{j+1}^T[\phi(x(t)) - \phi(x(t+T))] = \int_t^\infty r(x(s), \mu_j(x(s)))ds. \quad (68)$$

Using VFA in the value iteration algorithm just defined yields

$$W_{j+1}^T \phi(x(t)) = \int_t^\infty r(x(s), \mu_j(x(s))) ds + W_j^T \phi(x(t+T)). \quad (69)$$

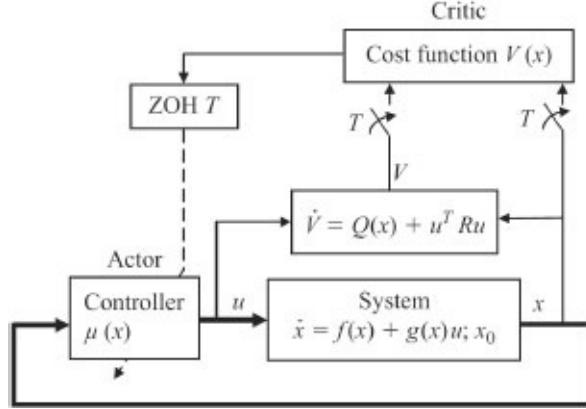


Figure 12.3: Hybrid optimal adaptive controller based on Integral Reinforcement Learning. Shows the two-time scale hybrid nature of the IRL controller. The integral reinforcement signal $\rho(t)$ is added as an extra state and functions as the memory of the controller. The Critic runs on a slow time scale and learns the value of using the current control policy. When the Critic converges, the Actor control policy is updated to obtain an improved value.

Then RLS or least-squares batch is used to update the value function parameters in these equations. On convergence of the value parameters, the action is updated.

IRL provides an optimal adaptive controller, that is, an adaptive controller that measures data along the system trajectories and converges to optimal control solutions. Note that only the system input coupling dynamics $g(x)$ is needed to implement these algorithms. The drift dynamics $f(x)$ is not needed. The implementation of IRL optimal adaptive control is shown in Figure 12.3. The time is incremented at each iteration by the period T . The reinforcement learning time interval T need not be the same at each iteration. T can be changed depending on how long it takes to receive meaningful information from the observations and is not a sample period in the standard meaning. The measured data at each time increment are $(x(t), x(t+T), \rho(t : t+T))$ where,

$$\rho(t : t+T) = \int_t^{t+T} r(x(\tau), u(\tau)) d\tau \quad (70)$$

is the integral reinforcement measured on each time interval. The integral reinforcement can be computed in real time by introducing an integrator $\dot{\rho} = r(x(t), u(t))$ as shown in Figure 12.3. That is, the integral reinforcement $\rho(t)$ is added as an extra continuous-time state. It functions as the memory or controller dynamics. The remainder of the controller is a sampled-data controller. Note that the control policy $\mu(t)$ is updated periodically after the critic weights have converged to the solution to Equation (68) or Equation (69). Therefore, the policy is piecewise constant in time. On the other hand, the control varies continuously with the state between each policy update. It is seen that IRL for continuous-time systems is in fact a hybrid continuous-time/discrete-time adaptive controller that converges to the optimal control solution in real-time without knowing the drift dynamics $f(x)$. The optimal

adaptive controller has multiple control loops and several timescales. The inner control action loop operates in continuous-time. Data is sampled at intervals of length T . The critic network operates at a slower timescale depending on how long it takes to solve Equation (68) or Equation (69). Due to the fact that the policy update for continuous-time systems does not involve the drift dynamics $f(x)$, no actor neural network is needed in IRL. Only a critic neural network is needed for VFA.

12.8. Synchronous Optimal Adaptive Control for Continuous-Time System

The IRL controller just gives tunes to the critic neural network for determining the value while holding the control policy fixed. Then, a policy update is performed. Now we develop an adaptive controller that has two neural networks, one for value function approximation and one to approximate the control. We could call these the critic neural network and actor neural network. The two neural networks are tuned simultaneously, that is, synchronously in time. This procedure is more nearly in line with accepted practice in adaptive control. Though this synchronous controller does require knowledge of the dynamics, it converges to the approximate local solutions to the HJB equation and the Bellman equation online, yet does not require explicitly solving either one. The HJB is usually impossible to solve for nonlinear systems except for special cases.

Based on the continuous-time Hamiltonian equation (61) and the stationarity condition $0 = \frac{\partial H(x, u, \nabla V^\mu)}{\partial \mu}$ a policy iteration algorithm for continuous-time systems could be written based on the policy evaluation step,

$$\begin{aligned} 0 &= H(x, \mu_j(x), \nabla V_{j+1}) \\ &\equiv r(x, \mu_j(x)) + (\nabla V_{j+1})^T(f(x) + g(x)\mu_j(x)), V_{j+1}(0) = 0 \end{aligned} \quad (71)$$

and the policy improvement step

$$\mu_{j+1} = \arg \min_{\mu} H(x, \mu, \nabla V_{j+1}). \quad (72)$$

Unfortunately, Equation (71) is a nonlinear partial differential equation and cannot usually be solved analytically. However, this policy iteration algorithm provides the structure needed to develop another adaptive control algorithm that can be implemented online using measured data along the trajectories and converges to the optimal control. Specifically, select a value function approximation (VFA), or critic neural network, structure as

$$V(x) = W_1^T \phi(x) \quad (73)$$

and a control action approximation structure or actor neural network as

$$u(x) = -\frac{1}{2} R^{-1} g^T(x) \nabla \phi^T W_2. \quad (74)$$

These approximators could be, for instance, two neural networks with unknown parameters or weights W_1 , W_2 , and $\phi(x)$ the basis set or activation functions of the first neural network. The structure of the second action neural network comes from the policy iteration algorithm. Then tuning the neural network weights as

$$\dot{W}_1 = -\alpha_1 \frac{\sigma}{(\sigma^T \sigma + 1)^2} [\sigma^T W_1 + Q(x) + u^T R u], \quad (75)$$

and

$$\dot{W}_2 = -\alpha_2 \left\{ (F_2 W_2 - F_2 \bar{\sigma}^T W_1) - \frac{1}{4} D(x) W_2 m^T(x) W_1 \right\}, \quad (76)$$

guarantees stability as well as convergence to the optimal value and control (Vamvoudakis and Lewis, 2010). In these parameter estimation algorithms $\alpha_1, \alpha_2, F_1, F_2$ are positive tuning parameters, $D(x) = \nabla \phi(x) g(x) R^{-1} g^T(x) \nabla \phi^T(x)$, $\sigma = \nabla \phi(f + gu)$, $\bar{\sigma} = \frac{\sigma}{\sigma^T \sigma + 1}$ and $m(x) = \frac{\sigma}{(\sigma^T \sigma + 1)^2}$. A Persistence of Excitation $\bar{\sigma}(t)$ is needed to achieve convergence to the optimal value.

This synchronous policy iteration controller is an adaptive control algorithm that requires full knowledge of the system dynamics $f(x), g(x)$, yet converges to the optimal control solution. That is, it solves, locally approximately, the HJB equation, which is usually intractable for nonlinear systems. In the continuous-time LQR case, it solves the ARE using data measured along the trajectories and knowledge of A, B . The utility of this algorithm is that it can approximately solve the HJB equation for nonlinear systems using data measured along the system trajectories in real time. The HJB is usually impossible to solve for nonlinear systems except in some special cases.

The VFA tuning algorithm for W_1 is based on gradient descent, while the control action tuning algorithm is a form of backpropagation (Werbos, 1989) that is, however, also tuned by the VFA weights W_i . This adaptive structure is similar to the actor-critic reinforcement learning structure in [Figure 12.1](#). However, in contrast to IRL, this algorithm is a continuous-time optimal adaptive controller with two parameter estimators tuned simultaneously, that is, synchronously and continuously in time. An algorithm that combines the advantages of the synchronous policy iteration algorithm and IRL has been proposed in Vamvoudakis *et al.* (2013). The authors in Jiang and Jiang (2012) have proposed a model-free IRL algorithm while the authors in Bhasin *et al.* (2013) have used neural network-based identifiers for a model-free synchronous policy iteration.

12.9. Using Reinforcement Learning for Multi-Player Nash and Stackelberg Games

The ideas presented in this book chapter can be applied to multi-player games and H-infinity control. Game theory provides an ideal environment to study multi-player decision and control problems, and offers a wide range of challenging and engaging problems. Game theory has been successful in modeling strategic behavior, where the outcome for each player depends on the actions of himself and all the other players. Every player chooses a control to minimize independently from the others his own performance objective. Multi-player cooperative games rely on solving coupled Hamilton–Jacobi (HJ) equations, which in the linear quadratic case reduce to the coupled algebraic Riccati equations. Reinforcement learning techniques have been applied to design adaptive controllers that converge to the solution of two-player zero-sum games in Vrabie and Lewis (2011); Zhang *et al.* (2013), of multi-player nonzero-sum games in Vamvoudakis and Lewis (2011); Vamvoudakis *et al.* (2012a) and of Stackelberg games in Vamvoudakis *et al.* (2012b). In these cases, the adaptive control structure has multiple loops, with action networks and critic networks for each player. The adaptive controller for zero-sum games finds the solution to the H-infinity control problem online in real time. A Q-learning approach to finding the H-infinity controller online is given in Kim and Lewis (2010). This adaptive controller does not require any systems dynamics information.

12.10. Conclusion

This book chapter uses computational intelligence techniques to bring together adaptive control and optimal control. Adaptive controllers do not generally converge to optimal solutions, and optimal controllers are designed offline using full dynamics information by solving matrix design equations. The book chapter shows that methods from reinforcement learning can be used to design new types of adaptive controllers that converge to optimal control solutions online in real time by measuring data along the system trajectories. These are called optimal adaptive controllers, and they have multi-loop, multi-timescale structures that come from reinforcement learning methods of policy–iteration and value iteration. These controllers learn the solutions to Hamilton-Jacobi design equations such as the Riccati equation online without knowing the full dynamical model of the system. A method known as Q learning allows the learning of optimal control solutions online, in the discrete-time case, for completely unknown systems. Q learning has not yet been fully investigated for continuous-time systems.

References

- Abu-Khalaf, M., Lewis, F. L. and Huang, J. (2006). Policy iterations on the Hamilton–Jacobi–Isaacs equation for state feedback control with input saturation. *IEEE Trans. Autom. Control*, 51(12), pp. 1989–1995.
- Al-Tamimi, A., Lewis, F. L. and Abu-Khalaf, M. (2008). Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof. *IEEE Trans. Syst. Man Cybern. Part B*, 38(4), pp. 943–949, special issue on ADP/RL.
- Astrom, K. J. and Wittenmark, B. (1995). *Adaptive Control*. UK: Addison-Wesley.
- Baird, L. C. (1994). Reinforcement learning in continuous time: advantage updating. *Proc. Int. Conf. Neural Netw.*, Orlando, FL, June, pp. 2448–2453.
- Balakrishnan, S. N., Ding, J. and Lewis, F. L. (2008). Issues on stability of ADP feedback controllers for dynamical systems. *IEEE Trans. Syst. Man Cybern. Part B*, 38(4), pp. 913–917, special issue on ADP/RL, invited survey paper.
- Barto, A. G., Sutton, R. S. and Anderson, C. (1983). Neuron-like adaptive elements that can solve difficult learning control problems. *IEEE Trans. Syst. Man Cybern.*, pp. 834–846.
- Bellman, R. E. (1957) *Dynamic Programming*. Princeton, NJ: Princeton University Press.
- Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. MA: Athena Scientific.
- Bhasin, S., Kamalapurkar, R., Johnson, M., Vamvoudakis, K. G., Lewis, F. L. and Dixon, W. E. (2013). A novel actor–critic–identifier architecture for approximate optimal control of uncertain nonlinear systems. *Autom.*, 49(1), pp. 82–92.
- Bradtko, S., Ydstie, B. and Barto, A. (1994). Adaptive linear quadratic control using policy iteration. *Proc. Am. Control Conf.*, Baltimore, pp. 3475–3479.
- Busoniu, L., Babuska, R., De Schutter, B. and Ernst, D. (2009). *Reinforcement Learning and Dynamic Programming Using Function Approximators*. Boca Raton: CRC Press.
- Cao, X. (2007). *Stochastic Learning and Optimization*. Berlin: Springer-Verlag.
- Dierks, T. and Jagannathan, S. (2010). Optimal control of affine nonlinear continuous-time systems. *Proc. Am. Control Conf.*, pp. 1568–1573.
- Doya, K. (2000). Reinforcement learning in continuous time and space. *Neural Comput.*, 12, pp. 219–245.
- Doya, K., Kimura, H. and Kawato, M. (2001). Neural mechanisms for learning and control. *IEEE Control Syst. Mag.*, pp. 42–54.
- Ferrari, S. and Stengel, R. F. (2002). An adaptive critic global controller. *Proc. Am. Control Conf.*, May, pp. 2665–2670.
- Hanselmann, T., Noakes, L. and Zaknich, A. (2007). Continuous-time adaptive critics. *IEEE Trans. Neural Netw.*, 18(3), pp. 631–647.
- Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. Cambridge, MA: MIT Press.
- Ioannou, P. and Fidan, B. (2006). *Adaptive Control Tutorial*. Philadelphia: SIAM Press.
- Jiang, Y. and Jiang, Z. P. (2012). Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics. *Autom.*, 48, pp. 2699–2704.
- Kim, J. H. and Lewis, F. L. (2010). Model-free H-infinity control design for unknown linear discrete-time systems via Q-learning with LMI. *Autom.*, 46(8), pp. 1320–1326.
- Lewis, F. L. and Liu, D. (2013). Reinforcement learning and approximate dynamic programming for feedback control. *IEEE Press Series on Computational Intelligence*.
- Lewis, F. L. and Vamvoudakis, K. G. (2011). Reinforcement learning for partially observable dynamic processes: adaptive dynamic programming using measured output data. *IEEE Trans. Syst. Man Cybern.-Part B*, 41(1), pp. 14–25.
- Lewis, F. L., Lendaris, G. and Liu, D. (2008) Special issue on approximate dynamic programming and reinforcement learning for feedback control. *IEEE Trans. Syst. Man Cybern. Part B*, 38(4).

- Lewis, F. L., Vrabie, D., Vamvoudakis, K. G. (2012a). Reinforcement learning and feedback control: using natural decision methods to design optimal adaptive controllers. *IEEE Control Syst. Mag.*, 32(6), pp. 76–105.
- Lewis, F. L., Vrabie, D. and Syrmos, V. (2012b). *Optimal Control, Third Edition*. New York: John Wiley.
- Li, Z.-H. and Krstic, M. (1997). Optimal design of adaptive tracking controllers for nonlinear systems. *Autom.*, 33(8), pp. 1459–1473.
- Liu, D., Yang, X. and Li, H. (2013). Adaptive optimal control for a class of continuous-time affine nonlinear systems with unknown internal dynamics, to appear in *Neural Comput. Appl.*
- Mehta, P. and Meyn, S. (2009). Q-learning and Pontryagin's minimum principle. *Proc. IEEE Conf. Decis. Control*, pp. 3598–3605.
- Mendel, J. M. and MacLaren, R. W. (1970). Reinforcement learning control and pattern recognition systems. In Mendel, J. M. and Fu, K. S. (eds.), *Adaptive, Learning, and Pattern Recognition Systems: Theory and Applications*. New York: Academic Press, pp. 287–318.
- Moore, K. L. (1993). *Iterative Learning Control for Deterministic Systems*. London: Springer-Verlag.
- Papoulis, A. (2002). *Probability, Random Variables and Stochastic Processes*. New York: McGraw-Hill.
- Powell, W. B. (2011). *Approximate Dynamic Programming, Second edition*. Hoboken: Wiley.
- Prokhorov, D. (2008). (ed.), *Computational Intelligence in Automotive Applications*, Heidelberg: Springer.
- Prokhorov, D. and Wunsch, D. (1997). Adaptive Critic Designs. *IEEE Trans. Neural Netw.*, 8(5), pp. 997–1007.
- Prokhorov, D., Santiago, R. A. and Wunsch II, D. C. (1995). Adaptive critic designs: A case study for neurocontrol. *Neural Netw.*, 8(9), pp. 1367–1372.
- Schultz, W. (2004). Neural coding of basic reward terms of animal learning theory, game theory, microeconomics and behavioral ecology. *Curr. Opinion Neurobiol.*, 14, pp. 139–147.
- Si, J., Barto, A., Powell, W. and Wunsch, D. (2004). *Handbook of Learning and Approximate Dynamic Programming*. USA: IEEE Press.
- Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning — An Introduction*. Cambridge, Massachusetts: MIT Press.
- Vamvoudakis, K. G. and Lewis, F. L. (2010). Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Autom.*, 46(5), pp. 878–888.
- Vamvoudakis, K. G. and Lewis, F. L. (2011). Multi-player non-zero sum games: online adaptive learning solution of coupled Hamilton-Jacobi equations. *Autom.*, 47, pp. 1556–1569.
- Vamvoudakis, K. G., Lewis, F. L. and Hudas, G. R. (2012). Multi-agent differential graphical games: online adaptive learning solution for synchronization with optimality. *Autom.*, 48(8), pp. 1598–1611.
- Vamvoudakis, K. G., Lewis, F. L., Johnson, M. and Dixon, W. E. (2012). Online Learning Algorithm for Stackelberg Games in Problems with Hierarchy. *Proc. 51st IEEE Conf. Decis. Control*. Maui, HI, pp. 1883–1889.
- Vamvoudakis, K. G., Vrabie, D. and Lewis, F. L. (2013). Online learning algorithm for optimal control with integral reinforcement learning, to appear in *Int. J. Robust Nonlin. Control*.
- Van der Schaft, A. J. (1992). L₂-gain analysis of nonlinear systems and nonlinear state feedback H ∞ control. *IEEE Trans. Autom. Control*, 37(6), pp. 770–784.
- Vrabie, D. and Lewis, F. L. (2009). Neural network approach to continuous-time direct adaptive optimal control for partially-unknown nonlinear systems. *Neural Netw.*, 22(3), pp. 237–246.
- Vrabie, D. and Lewis, F. L. (2011). Adaptive dynamic programming for online solution of a zero-sum differential game. *J. Control Theory Appl.*, 9(3), pp. 353–360.
- Vrabie, D., Pastravanu, O., Abu-Khalaf, M. and Lewis, F. L. (2009). Adaptive optimal control for continuous-time linear systems based on policy iteration. *Autom.*, 45, pp. 477–484.
- Vrabie, D., Vamvoudakis, K. G. and Lewis, F. L. (2012). *Optimal Adaptive Control and Differential Games by Reinforcement Learning Principles, Control Engineering Series*. IET Press.
- Wang, D., Liu, D., Wei, Q. (2012). Finite-horizon neuro-optimal tracking control for a class of discrete-time nonlinear

- systems using adaptive dynamic programming approach. *Neurocomputing*, 78, p. 1422.
- Wang, F. Y., Zhang, H. and Liu, D. (2009). Adaptive dynamic programming: an introduction. *IEEE Comput. Intell. Mag.*, pp. 39–47.
- Watkins, C. (1989). *Learning from Delayed Rewards*. Ph.D. Thesis. Cambridge, England: Cambridge University.
- Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Mach. Learn.*, 8, pp. 279–292.
- Werbos, P. J. (1989). Neural networks for control and system identification. *Proc. IEEE Conf. Decis. Control*.
- Werbos, P. J. (1991). A menu of designs for reinforcement learning over time. In Miller, W. T., Sutton, R. S. and Werbos, P. J. (eds.), *Neural Networks for Control*. Cambridge: MIT Press.
- Werbos, P. J. (1992). Approximate dynamic programming for real-time control and neural modeling. In White, D. A. and Sofge, D. A. (eds.), *Handbook of Intelligent Control*. New York: Van Nostrand Reinhold.
- Wheeler, R. M. and Narendra, K. S. (1986). Decentralized learning in finite Markov chains. *IEEE Trans. Autom. Control*, 31(6).
- Zhang, H., Cui, L., Zhang, X. and Luo, X. (2011). Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method. *IEEE Trans. Neural Netw.*, 22, pp. 2226–2236.
- Zhang, H., Huang, J. and Lewis, F. L. (2009). Algorithm and stability of ATC receding horizon control. *Proc. IEEE Symp. ADPRL*. Nashville, March, pp. 28–35.
- Zhang, H., Liu, D., Luo, Y. and Wang, D. (2013). *Adaptive Dynamic Programming for Control: Algorithms and Stability*. Heidelberg: Springer.
- Zhang, H., Wei, Q., Luo, Y. (2008). A novel infinite-time optimal tracking control scheme for a class of discrete-time nonlinear systems via the greedy HDP iteration algorithm. *IEEE Trans. Syst. Man Cybern. Part B: Cybern.*, 38, pp. 937–942.

Chapter 13

Kernel Models and Support Vector Machines

Denis Kolev, Mikhail Suvorov and Dmitry Kangin

Though linear models remain very attractive for researchers in the area of pattern recognition, they cannot meet some of the emerging difficulties of classification and regression problems. In this chapter, the authors discuss kernel methods as a theoretically justified tool, introducing non-linearity into originally linear models while preserving the overall model structure. Based on solid theoretical foundations, they describe some interesting kernel models and then focus on the SVM framework as it is a particularly important family of models within the area of kernel methods.

13.1. Introduction

Nowadays, the amount of information produced by different sources is steadily growing. For that reason, statistical data analysis and pattern recognition algorithms become more and more pervasive and highly demanded (Bishop, 2006; Hastie *et al.*, 2003). The most studied and widely applied approaches are based on parametric linear models. They combine learning and evaluation simplicity and well-grounded theoretical basis. On the other hand, linear methods impose strong restrictions on the data structure model, significantly reducing, in some cases, performance. Nonlinear methods provide more flexible models but usually they suffer from very complex, time demanding learning procedures and possible overfitting.

In this chapter, we discuss a framework for parametric linear methods that determine nonlinear models: kernel methods. These methods preserve theoretical basis of the linear models providing, at the same time, more flexibility. This framework extends linear machine learning methods.

Support vector machines (SVMs) play a significant role in the scope of linear and kernel methods. It is one of the most popular approaches for regression and classification problems due to a large number of reasons. The given approach is theoretically well-grounded and studied that helps to adapt it to various problems. SVMs are known to be universal approximators in some specific conditions. Also, due to the sparseness of the model (described further), SVMs are widely used for processing of streaming/large data. SVM classifier is rated as one of the best-performance machine learning algorithms (Wu *et al.*, 2008). A number of problem specific techniques are based on “SVM-driven” ideas. In this chapter, we discuss the SVM method and its extensions focusing on their relationship with kernel methods.

The rest of the chapter is structured as follows. In the second section, we discuss theoretical basis of kernel methods and give illustrative examples. In the third section, we list main kernel machines explaining their principles. The fourth section is focused on the SVM method and its extensions.

13.2. Basic Concepts and Theory

Here, we present a brief overview of the basic concepts and theorems about kernels and kernel models. Hereafter, we will use the following notation, unless stated otherwise.

$X \subseteq \mathbb{R}^n$ — set of objects/vectors, Universum.

n — dimensionality of the object space.

$Y \subseteq \mathbb{R}^m$ — set of responses.

m — dimensionality of responses.

$X_N = \{(x_1, y_1), \dots, (x_N, y_N)\}$ — sample, where,

$x_i \in X$ — sample object/vector, $i = 1, \dots, N$,

$y_i \in Y$ — response to x_i , $i = 1, \dots, N$.

\mathcal{H} — Hilbert space of functions or Reproducing Kernel Hilbert Space.

$K(x, t)$ — kernel function defined on $X \times X$.

13.2.1. Regression and Classification Problems

In many real-world problems, the classical approach of building a mathematical model of relationship between inputs and outputs turns to be unsound. Complicated dependencies, inputs of complex structure and noise are all the reasons that hinder mathematical modeling. In fact, development of a strict mathematical model is often infeasible or is related to strong performance and application constraints, which often limits the model's scalability. The alternative is information models that allow for restoring rather complex input–output relationships, dealing with noise and outliers. In this chapter, we speak about regression and classification problems (Bishop, 2006), and kernel approaches to solve them.

Inputs and outputs of real-world problems have various complicated structures. But in many cases they can be quantified. So hereafter we consider sets of data objects as subsets of \mathbb{R}^n . Here, we state the problem of recognition.

Consider a subset $X \subseteq \mathbb{R}^n$. We refer its elements as *objects*. Coordinates x_i of object $x \in X$ are called *features* and marked by Roman script i . Each object x is associated with a vector $y \in Y \subseteq \mathbb{R}^m$, regarded as a kind of characteristic of objects in X . We assume that there exists a law f that puts objects and their characteristics in correspondence. So f can be regarded as a map $X \rightarrow Y$ and

$$y = f(x), \quad x \in X, \quad y \in Y.$$

We have some assumptions about this law/mapping and a *training set*. Training set is a very important concept in recognition theory as it is the main source of recovering f or at

least its approximation \hat{f} . Usually, we assume that mapping \hat{f} belongs to a certain family of maps $F \subseteq X^Y$. Then the training set is used to choose one mapping from this family. More formally, *training set* is a set X_N of pairs (x_i, y_i) :

$$X_N = \{(x_i, y_i) | y_i = f(x_i), i = 1, \dots, N\}.$$

Then learning algorithm can be regarded as a mapping μ , so that

$$\begin{aligned}\hat{f} &= \mu(X_N), \\ \hat{f} &\in F.\end{aligned}$$

When Y is a finite set,

$$Y = \{L_1, \dots, L_R\},$$

then the problem is called classification problem. L_r are called *labels*.

When $Y \subseteq \mathbb{R}$ and contains innumerable number of elements, then the problem is called a regression problem.

13.2.2. Support Vector Machines, Classification Problem, Separating Hyperplane

To illustrate recognition problem discussed above, we will give a simple example of a classification problem inspired by Vapnik and Chervonenkis (Vapnik, 1995; Vapnik and Chervonenkis, 1964). Let $X = \mathbb{R}^2$, $Y = \{-1, 1\}$. Then objects can be regarded as two-dimensional points represented by either squares ($y = -1$), or circles ($y = 1$). Following Vapnik and Chervonenkis, we start with considering a simple case of linearly separable training set, presented in [Figure 13.1](#). As points of training set of different classes are linearly divisible, the goal is to find a hyperplane H that will divide X into two parts H_+ and H_- . Hyperplane is determined by its normal vector w and intercept ρ . Then a vector x belongs to the hyperplane if and only if

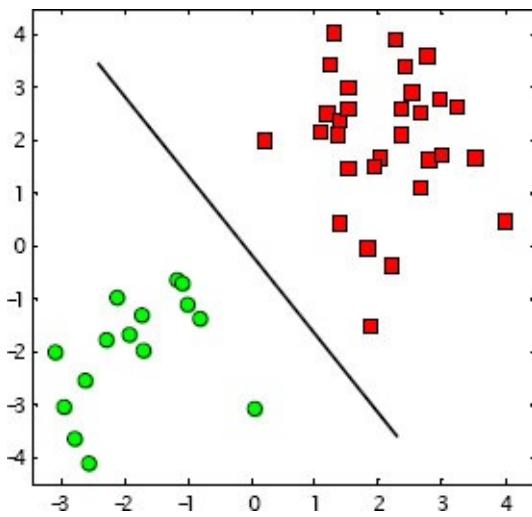


Figure 13.1: Linearly separable case.

$$\langle w, x \rangle - \rho = 0.$$

The hyperplane can be chosen in different ways. Vapnik and Chervonenkis proposed to choose an some optimal hyperplane. They formalized optimality using the notation of

separating band, which is the space between two hyperplanes parallel to H , that contain no points of the training set \mathbf{X}_N . They showed that the width of this band equals (for proof refer to [Section 13.4.1](#)).

$$\frac{2}{\|\mathbf{w}\|}.$$

Then, they stated the problem of quadratic programming

$$\begin{aligned} \|\mathbf{w}\|^2 &\rightarrow \min_{\mathbf{w}} \\ \text{w.r.t.} \\ y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle - \rho) &\geq 1, \\ i &= 1, \dots, N. \end{aligned}$$

This can be generalized for the case of a linearly non-separable training set by introducing the so called *slack* variables $\xi_i \geq 0$ that allow violating the constraint. So the final version of the problem becomes

$$\begin{aligned} y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle - \rho) &\geq 1 - \xi_i, \\ \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i &\rightarrow \min_{\mathbf{w}, \xi} \\ \text{w.r.t.} \\ y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle - \rho) &\geq 1 - \xi_i, \\ \xi_i &\geq 0, \\ i &= 1, \dots, N. \end{aligned}$$

Formally speaking, in the case of linearly non-separable sample, the set of the optimization problem constraints defines an empty set. Introduction of slack variables prevents such situation allowing misclassifications, which are penalized by the functional under optimization.

13.2.3. SVM Solver, Towards Nonlinearity, Polynomial Kernel

In this section, we do not investigate the main properties of the problem solution. Instead, we focus on the techniques that introduce nonlinearity into the model, describing briefly the background theory and most vital theorems.

Consider the solution of the optimization problem stated by two-class SVM classifier. Earlier, we have presented the convex quadratic programming problem, which has a single solution. Usually, the SVM model is investigated from the point of the dual optimization problem (Boyd and Vandenberghe, 2004), which is formulated as follows:

$$\begin{aligned}
-\frac{1}{2} \alpha Q \alpha^T + \sum_{i=1}^N \alpha_i &\rightarrow \max_{\alpha} \\
\text{w.r.t.} \\
&\left\{ \begin{array}{l} 0 \leq \alpha_i \leq C, \quad \forall i = 1, \dots, N, \\ \sum_{i=1}^N \alpha_i y_i = 0. \end{array} \right.
\end{aligned}$$

Here the N -dimensional vector $\alpha \in \mathbb{R}^N$ represents the vector of Lagrange multipliers, corresponding to each constraint in the primal problem. Matrix $Q \in \mathbb{R}^{N \times N}$ is a matrix of “class-corresponding” inner products with elements

$$Q_{ij} = \langle x_i, x_j \rangle y_i y_j.$$

According to the differential Karush–Kuhn–Tucker conditions of the primal problem (it will be detailed further), the normal vector of the resulting hyperplane can be expressed as

$$w = \sum_{i=1}^N y_i \alpha_i x_i.$$

Therefore, the class label of arbitrary vector x can be evaluated as

$$y(x) = \text{sign} \left(\sum_{i=1}^N \alpha_i y_i \langle x_i, x \rangle - \rho \right).$$

Thus, one can see that the optimization problem depends on the values of the Lagrange multipliers and inner products between the vectors from the learning set and classified vector. This information is sufficient for performing classification.

The SVM classifier discussed here is linear, which means that it performs adequately only in the cases when the classes in the learning set (and in the Universum at all) are linearly separable. However, this limitation can be overcome. There are a lot of specially developed techniques that can be applied, for instance, feature engineering, mixture of experts, etc. But for the SVM model (and a wide range of other machine learning methods) there is a special approach, which keeps the overall linear model structure, but allows fitting nonlinear separation surfaces. This approach is referred as “mapping to linearization space” or just “kernel trick” (Silverman, 1986).

Coming back to the SVM’s dual optimization problem and final separating function we note that the optimization problem depends on pairwise inner products in the learning set. The final classification rule also depends only on the inner products between vectors in the learning set and vector classified. It gives the way to the described approach. The idea is to replace the standard inner product by a two argument nonlinear function (Schölkopf *et al.*, 1999), so that

$$\begin{aligned}\langle \mathbf{x}, \mathbf{t} \rangle &\rightarrow K(\mathbf{x}, \mathbf{t}), \\ Q_{ij} &= K(\mathbf{x}_i, \mathbf{x}_j) y_i y_j, \\ y(\mathbf{x}) &= \text{sign} \left(\sum_{i=1}^N \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) - \rho \right).\end{aligned}$$

A simple example can be provided. Suppose that the replacement is

$$\langle \mathbf{x}, \mathbf{t} \rangle \rightarrow (\langle \mathbf{x}, \mathbf{t} \rangle + 1)^2.$$

Then, the decision rule becomes a second-order polynomial, i.e.,

$$y(\mathbf{x}) = \text{sign} \left(\sum_{i=1}^N \alpha_i y_i \langle \mathbf{x}, \mathbf{x}_i \rangle^2 + 2 \sum_{i=1}^N \alpha_i y_i \langle \mathbf{x}, \mathbf{x}_i \rangle + \sum_{i=1}^N \alpha_i y_i - \rho \right).$$

However, two important questions must be answered to ensure a proper use of such approach:

- (1) What function $K(\mathbf{x}, \mathbf{t})$, *kernel if for any finite subset*) can be used in place of the inner product?
- (2) What is the set of possible separating surfaces that can be fitted using given $K(\mathbf{x}, \mathbf{t})$?

13.2.4. Kernel Function Discussion

Kernel trick can be defined in different ways. Consider a function Φ that maps the data from the initial space \mathbb{R}^n to a Hilbert space \mathcal{H} , $\Phi: \mathbb{R}^n \rightarrow \mathcal{H}$. Inner product in the space \mathcal{H} is denoted by $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. In general case, the space \mathcal{H} can be infinite-dimensional.

Classification model is learned in the space \mathcal{H} using mapped data as a learning set,

$$\Phi(X_N) = \{\Phi(\mathbf{x}_1), \dots, \Phi(\mathbf{x}_N)\}.$$

It is supposed that for all $\mathbf{x}, \mathbf{t} \in \mathbb{R}^n$, the inner product for the mapped vectors is evaluated as

$$\langle \Phi(\mathbf{x}), \Phi(\mathbf{t}) \rangle_{\mathcal{H}} = K(\mathbf{x}, \mathbf{t}),$$

where $K(\mathbf{x}, \mathbf{t})$ is a symmetric two-argument function, in general nonlinear, defined for vectors in \mathbb{R}^n . $\Phi(\mathbf{x})$ can be considered as a new feature map, that makes “kernel trick” similar to the feature engineering.

Therefore, the final classification model can be expressed as

$$\begin{aligned}y(\mathbf{x}) &= \text{sign} \left(\sum_{i=1}^N \alpha_i y_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \rangle_{\mathcal{H}} - \rho \right) \\ &= \text{sign} \left(\sum_{i=1}^N \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) - \rho \right).\end{aligned}$$

Kernel trick helps building linear classification model in a different space resulting in a generally nonlinear classification model in the initial data space. For that reason, the function $K(\mathbf{x}, \mathbf{t})$ must be an inner product of the $\Phi(\mathbf{x})$ and $\Phi(\mathbf{t})$ in a Hilbert space for all \mathbf{x}, \mathbf{t}

$\in \mathbb{R}^n$. Such functions are usually referred as “kernels” (this is not a strict definition) and there are a lot of examples of such functions:

- Linear kernel $K(\mathbf{x}, \mathbf{t}) = \langle \mathbf{x}, \mathbf{t} \rangle$,
- Polynomial kernel $K(\mathbf{x}, \mathbf{t}) = \langle \mathbf{x}, \mathbf{t} \rangle^d$ or $K(\mathbf{x}, \mathbf{t}) = (\langle \mathbf{x}, \mathbf{t} \rangle + \beta)^d$,
- Exponential kernel $K(\mathbf{x}, \mathbf{t}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{t}\|^2}{2\sigma^2}\right)$,
- Student-type RBF kernel $K(\mathbf{x}, \mathbf{t}) = (\beta + \gamma \|\mathbf{x} - \mathbf{t}\|^2)^{-\alpha}$, $\beta > 0$, $\gamma > 0$, $\alpha > 0$.

Discussing the diversity of the set of SVM’s separating functions that can be learned using a given kernel function, it is essential to point that the classification rule is based on a linear combination of kernel functions with one fixed input argument. Therefore, the set of functions that can be obtained is limited by the linear spectrum of the kernel:

$$\text{span}(K) = \left\{ f : \mathbb{R}^n \rightarrow \mathbb{R} \mid \exists \mathbf{x}_1, \dots, \mathbf{x}_N \in \mathbb{R}^n : f(\mathbf{x}) = \sum_{i=1}^N \alpha_i K(\mathbf{x}, \mathbf{x}_i) \right\}.$$

Further, we investigate the underlying theory of the “kernel trick” in detail, including the criteria for a function to be a kernel and the importance of the spectrum of the kernels. We finalize the chapter by the representer theorem, showing the significance of the kernel theory for the machine learning.

13.2.5. Kernel Definition via Matrix

This discussion leads us to the following definition of a kernel function.

Definition 1 (Cuturi, 2010): *Function $K : X \times X \rightarrow \mathbb{R}$ is called a positive definite kernel if for any finite subset $X_N = \{\mathbf{x}_1, \dots, \mathbf{x}_N\} \subset X$ the corresponding matrix $K(X_N, X_N)$,*

$$K(X_N, X_N) = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & \cdots & K(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ K(\mathbf{x}_N, \mathbf{x}_1) & \cdots & K(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix},$$

is symmetric, i.e., $K(X_N, X_N) = K(X_N, X_N)^T$, and nonnegative definite.

The latter we understand as for any $\mathbf{t} \in \mathbb{R}^N$

$$\langle K(X_N, X_N)\mathbf{t}, \mathbf{t} \rangle \geq 0.$$

Directly from the definition it follows that

$$\begin{aligned} \forall \mathbf{x}, \mathbf{t} \in X, \\ K(\mathbf{x}, \mathbf{t}) &\geq 0, \\ K(\mathbf{x}, \mathbf{t}) &= K(\mathbf{t}, \mathbf{x}). \end{aligned}$$

It should be pointed that this fact provides important link to kernel SVM, as non-negative matrix $K(X_N, X_N)$ ensures the concavity of the dual problem, which is essential for the optimization.

13.2.6. Reproducing Kernel (RKHS) via Linear Operator

Up to this moment we introduced the definition of the positive definite kernel. Now we consider one of the most important entities in the theory of reproducing kernels — Reproducing Kernel Hilbert space (RKHS) (Aronszajn, 1950).

Let X be a non-empty subset of \mathbb{R}^n . Then let \mathcal{H} be a Hilbert space of functions over X :

$$f : X \rightarrow \mathbb{R}.$$

Definition 2: A function $K : X \times X \rightarrow \mathbb{R}$ is called a *reproducing kernel* of \mathcal{H} if and only if

1. $\forall x \in X K(\cdot, x) \in \mathcal{H}$.
2. $\forall f \in \mathcal{H} \langle f, K(\cdot, x) \rangle_{\mathcal{H}} = f(x)$.

The function $K(\cdot, x)$ is called reproducing kernel for point x . Here, $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ is a voluntarily chosen function meeting the inner product definition. From the definition, it follows that the reproducing kernel is a symmetric function as

$$\begin{aligned} K(x, t) &= \langle K(\cdot, t), K(\cdot, x) \rangle_{\mathcal{H}} \\ &= \{\text{property of real-valued inner product}\} \\ &= \langle K(\cdot, x), K(\cdot, t) \rangle_{\mathcal{H}} = K(t, x). \end{aligned}$$

The reproducing property may be defined in a different way. Consider a functional $L_x : \mathcal{H} \rightarrow \mathbb{R}$, so that

$$\forall f \in \mathcal{H} \quad L_x f = f(x).$$

The “evaluation functional” L_x can be defined for any $x \in X$. It is easy to show that L_x is a linear functional. Using this notation, we introduce another definition of RKHS.

Definition 3: A Hilbert space of functions \mathcal{H} is a RKHS if and only if $\forall x \in X$ the evaluation functional L_x is bounded, i.e.,

$$\forall x \in X \exists C_x : \forall f \in \mathcal{H} : |L_x f| \leq C_x \|f\|_{\mathcal{H}} = C_x \sqrt{\langle f, f \rangle_{\mathcal{H}}}.$$

So, it means that if two functions f and g are close in terms of RKHS norm, they will be close at every point:

$$|f(x) - g(x)| = |L_x(f - g)| \leq C_x \|f - g\|_{\mathcal{H}}.$$

Now we can state a theorem linking two definitions.

Theorem 1. Hilbert space \mathcal{H} is a RKHS \Leftrightarrow it has a reproducing kernel.

According to Riesz–Frechet theorem, for every bounded linear functional $A : \mathcal{H} \rightarrow \mathbb{R}$ there exists an element $\theta \in \mathcal{H} : \forall f \in \mathcal{H} Af = \langle f, \theta \rangle_{\mathcal{H}}$. In our case, $A \equiv L_x$. Therefore, one side of the theorem (\Rightarrow) is a specific case of Riesz–Frechet theorem. The other side (\Leftarrow) can be proved using Cauchy–Schwartz inequality.

13.2.7. Moore–Aronszajn Theorem: Equivalence

The problem of selecting a reproducing kernel arises. The Moore–Aronszajn theorem

builds a correspondence between positive definite kernels and RKHS.

Theorem 2 (Moore–Aronszajn (Aronszajn, 1950)). *Suppose $K : X \times X \rightarrow \mathbb{R}$ is a positive definite kernel $\Rightarrow \exists!$ Hilbert space \mathcal{H} of functions over X , where K is a reproducing kernel.*

Therefore, for each positive definite kernel there is a unique corresponding RKHS.

13.2.8. Mercer's Theorem

We further investigate properties of positive definite kernels in the RKHSs. For this purpose we introduce a strictly positive Borel measure μ on X . We define $L_\mu^2(X)$ as a linear space of square-integrable functions, i.e.,

$$f \in L_\mu^2(X) : \int_X |f(x)|^2 d\mu(x) < \infty.$$

A symmetric positive definite kernel $K : X \times X \rightarrow \mathbb{R}$, which is square integrable, i.e.,

$$\int_X \int_X K^2(x, t) d\mu(x) d\mu(t) < \infty,$$

induces (Minh *et al.*, 2006) a linear operator $L_K : L_\mu^2(X) \rightarrow L_\mu^2(X)$ defined by

$$L_K f(x) = \int_X K(x, t) f(t) d\mu(t).$$

It possesses a countable system of eigenfunctions $\{\phi_k\}_{k=1}^\infty$ forming an orthonormal basis of $L_\mu^2(X)$ with corresponding eigenvalues $\{\lambda_k\}_{k=1}^\infty$.

Theorem 3 (Mercer, 1909). *Let $X \subset \mathbb{R}^n$ be closed, μ a strictly positive Borel measure on X , which means that this measure of every nonempty subset in X is positive, K a square integrable continuous function on $X \times X$ being a positive definite kernel. Then*

$$K(x, t) = \sum_{k=1}^{\infty} \lambda_k \phi_k(x) \phi_k(t).$$

This series converges absolutely for each pair $(x, t) \in X \times X$ and uniformly on each compact subset of X .

This theorem allows building a map $\Phi : X \rightarrow \ell^2$, where ℓ^2 is a Hilbert space of square summable sequences, so that

$$[\Phi(x)]_k = \sqrt{\lambda_k} \phi_k(x).$$

Then kernel K can be regarded as a scalar product in ℓ^2 :

$$K(x, t) = \langle \Phi(x), \Phi(t) \rangle_{\ell^2} = \sum_{k=1}^{\infty} ([\Phi(x)]_k [\Phi(t)]_k) = \sum_{k=1}^{\infty} \lambda_k \phi_k(x) \phi_k(t).$$

As kernel K is supposed to be continuous, the feature map Φ is also continuous. Coming back to the SVM example, Mercer's theorem provides an explanation of the correctness of the positive definite kernels usage. Any square integrable positive definite kernel can be

considered as an inner product between maps of two vectors in some Hilbert space (namely ℓ^2) replacing the inner product in the initial space \mathbb{R}^n . $\Phi(\mathbf{x})$ can be regarded as a feature map to the transformed feature space,

$$\Phi : \mathcal{X} \rightarrow \ell^2.$$

Thus, SVM builds a linear classifier in the ℓ^2 space.

13.2.9. Polynomial Kernel Example

To illustrate the discussed theory we return to polynomial kernel example. This will build a link between RKHS reproducing kernel, positive definite kernel and “kernel trick”.

Consider a problem of fitting a linear regression in a two-dimensional space. For training set $\mathcal{X}_N = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$, $\mathbf{x}_i \in \mathbb{R}^2$, $y_i \in \mathbb{R}$ the goal is to restore parameter \mathbf{w} so that $\forall i = 1, \dots, N$, the output value of parametric function $f(\mathbf{x}_i, \mathbf{w})$ is close to y_i . The word “close” can be understood as an (sub)optimal solution of any regular empirical error measure like least squares, SVM-type measure, etc. For the current example, assume least-squares optimization is used as a learning procedure, i.e.,

$$\sum_{i=1}^N (f(\mathbf{x}_i, \mathbf{w}) - y_i)^2 \rightarrow \min_{\mathbf{w}}.$$

We choose function $f(\mathbf{x}, \mathbf{w})$ to be linear:

$$f(\mathbf{x}, \mathbf{w}) = \langle \mathbf{x}, \mathbf{w} \rangle = \sum_{i=1}^2 x_i w_i, \quad \mathbf{w} \in \mathbb{R}^2.$$

Suppose that “kernel trick” is performed, so regression is transformed to second order polynomial, i.e.,

$$\langle \mathbf{x}, \mathbf{t} \rangle_{\mathbb{R}^2} \rightarrow K(\mathbf{x}, \mathbf{t}) = [\langle \mathbf{x}, \mathbf{t} \rangle_{\mathbb{R}^2}]^2 = \langle \Phi(\mathbf{x}), \Phi(\mathbf{t}) \rangle_{\mathcal{H}}.$$

Then, the feature map $\Phi(\mathbf{x})$ is defined as

$$\Phi : \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 x_2 \end{bmatrix} \in \mathbb{R}^3.$$

Thus, \mathcal{H} may be understood as \mathbb{R}^3 . Inner product in this space is evaluated as,

$$\langle \Phi(\mathbf{x}), \Phi(\mathbf{t}) \rangle_{\mathbb{R}^3} = x_1^2 t_1^2 + x_2^2 t_2^2 + 2x_1 x_2 t_1 t_2 = [\langle \mathbf{x}, \mathbf{t} \rangle_{\mathbb{R}^2}]^2.$$

In this case, parameter $\mathbf{w} \in \mathbb{R}^3$ is growing in dimension. But the function model still depends linearly on \mathbf{w} , and this usually simplifies learning procedure. The model function is,

$$f(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^3 w_i [\Phi(\mathbf{x})]_i = w_1 x_1^2 + w_2 x_2^2 + w_3 \sqrt{2} x_1 x_2,$$

where $[\Phi(\mathbf{x})]_i$ denotes i th coordinate in the transformed feature map.

If we consider $f(\mathbf{x}, \mathbf{w})$ as a function-member of the Hilbert space of second-order polynomials over \mathbb{R}^2 , $P^2(\mathbb{R}^2)$, then every function in $P^2(\mathbb{R}^2)$ can be specified by its three coefficients, i.e.,

$$g(\mathbf{u}) = au_1^2 + bu_2^2 + cu_1u_2 \leftrightarrow \begin{bmatrix} a \\ b \\ c \\ \frac{c}{\sqrt{2}} \end{bmatrix}.$$

Here, \leftrightarrow denotes some schematic correspondence of the polynomial to vectors from \mathbb{R}^3 . Factor $\sqrt{2}$ is introduced for future convenience. We denote inner product in $P^2(\mathbb{R}^2)$ as a sum of pairwise multiplications of the corresponding coefficients,

$$\langle g_1, g_2 \rangle_{P^2(\mathbb{R}^2)} = a_1a_2 + b_1b_2 + \frac{c_1c_2}{2},$$

where $g_i(\mathbf{u}) = a_iu_1^2 + b_iu_2^2 + c_iu_1u_2 \in P^2(\mathbb{R}^2)$.

One can see that the polynomial

$$K(\mathbf{u}, \mathbf{x}) = x_1^2u_1^2 + x_2^2u_2^2 + 2x_1x_2u_1u_2 \leftrightarrow \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{bmatrix},$$

$$K(\cdot, \mathbf{x}) \in P^2(\mathbb{R}^2) \quad \forall \mathbf{x} \in \mathbb{R}^2,$$

is a reproducing kernel of $P^2(\mathbb{R}^2)$.

Indeed,

$$\forall g \in P^2(\mathbb{R}^2) \quad \langle g, K(\cdot, \mathbf{x}) \rangle_{P^2(\mathbb{R}^2)} = ax_1^2 + bx_2^2 + \frac{2x_1x_2c}{2} = g(\mathbf{x}).$$

Therefore, mapping $\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ is equivalent to mapping $\Psi : \mathbb{R}^2 \rightarrow P^2(\mathbb{R}^2)$, where

$$\Psi(\mathbf{x}) = K(\cdot, \mathbf{x}).$$

Inner product between the data mapped by Ψ is evaluated as $\langle \Psi(\mathbf{x}), \Psi(\mathbf{t}) \rangle_{P^2(\mathbb{R}^2)}$. Mapping Ψ is equivalent to mapping Φ as

$$K(\mathbf{x}, \mathbf{t}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{t}) \rangle_{\mathbb{R}^3} = \langle \Psi(\mathbf{x}), \Psi(\mathbf{t}) \rangle_{P^2(\mathbb{R}^2)} \quad \forall \mathbf{x}, \mathbf{t} \in \mathbb{R}^2.$$

Therefore, the regression learning procedure is reduced to the search of the polynomial

$$\pi \in P^2(\mathbb{R}^2), \quad \pi \leftrightarrow \begin{bmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \frac{\pi_3}{\sqrt{2}} \end{bmatrix},$$

so that $\forall i = 1, \dots, N \langle \pi, \Psi(\mathbf{x}_i) \rangle_{P^2(\mathbb{R}^2)} = \pi_1x_{i1}^2 + \pi_2x_{i2}^2 + \pi_3x_{i1}x_{i2}$ is close to y_i . However, due to the fact that π is defined by three coefficients, the function model is linear with respect to its parameters. Final optimization (“learning”) procedure is

$$\sum_{i=1}^N (\pi_1x_{i1}^2 + \pi_2x_{i2}^2 + \pi_3x_{i1}x_{i2} - y_i)^2 \rightarrow \min_{\pi},$$

where x_i corresponds to i th entry of i th learning sample.

The result provided here can be generalized. Every “kernel trick” is equivalent to defining a new feature map, where K is a reproducing kernel of the linearization space. However, the fitted parameters are not always included into the RKHS inner product in a linear way, which may cause problem during optimization. For instance, if we map the data into the space of all continuous functions, the learning procedure becomes equivalent to the variational optimization over all continuous functions. However, further we will show that in most of the cases this problem can be overcome and it is possible to leave the model linear with respect to the optimized parameters.

13.2.10. RKHS Approximation

The next question we address is an arbitrary separating surface using kernel K . Consider a RKHS \mathcal{H} , Hilbert space of real valued functions over $X \subseteq \mathbb{R}^n$, where K is a reproducing kernel. If X is a compact subset, we are able to form the space of kernel sections

$$K(X) = \overline{\text{span}}\{K_t, t \in X\},$$

which is a closure of a linear span of functions $K_t(\cdot) = K(\cdot, t)$. It is clear that $K(X) \subseteq \mathcal{H}$. But if we would like to build an arbitrary function (specifying a separating surface) we must call for the opposite embedding $\mathcal{H} \supseteq K(X)$. In other terms, for any positive number ε and any function $f \in \mathcal{H}$ there should exist such a function $g \in K(X)$ that function f is well approximated by g , i.e.,

$$\|f - g\|_{\mathcal{H}} \leq \varepsilon.$$

Returning to RKHS it may be shown that $K(X)$ is dense in \mathcal{H} , which means that any function from the RKHS can be approximated with any given precision ε by linear combination of reproducing kernels. Formally speaking (Micchelli *et al.*, 2006),

$$\forall \varepsilon > 0, \quad \forall f \in \mathcal{H} \exists N > 0, x_1, \dots, x_N \in X, \alpha_1, \dots, \alpha_N \in \mathbb{R}: \\ \left\| f - \sum_{i=1}^N \alpha_i K(\cdot, x_i) \right\|_{\mathcal{H}} \leq \varepsilon.$$

Thus, if we return to the SVM model we can state that the diversity of possible decision function forms is limited by the RKHS of the selected kernel.

In practice, most of the kernels are parametric functions. Thus it is reasonable to analyze how corresponding RKHS depends on kernel parameters. Consider a family of parameterized positive definite kernel functions

$$\mathcal{K} = \{K(\cdot, \cdot | \theta) : X \times X \rightarrow \mathbb{R} | \theta \in \Theta\},$$

where Θ is a set of acceptable values for θ . As an example of such a family, we can take Gaussian kernel with different variance values. A natural question that arises is whether corresponding RKHS depends on kernel parameters. In general, the answer is “yes”, but not in all cases. For instance, Gaussian kernel with different variances corresponds to the

RKHS of all continuous real-valued functions, which is independent from variance parameters. At the same time, polynomial kernels with different degrees correspond to different RKHS (spaces of polynomials with different degrees).

13.2.11. Universal Kernels and Examples

Using the notation from the previous subsection, we introduce the property of the *universality* of the kernel (Micchelli *et al.*, 2006).

Definition 4: Positive definite kernel $K : X \times X \rightarrow \mathbb{R}$, $X \subseteq \mathbb{R}^n$ is called universal if it is a reproducing kernel of RKHS $\mathbb{R}(X)$ — space of all continuous real-valued functions defined on X .

Using the previous result, we can state that any continuous real-valued function $f \in \mathbb{R}(X)$ can be approximated with any given precision by a function $g \in K(X)$.

So, the question is which kernels possess this property of universality. A number of universality criteria exist, but in this work we present just a number of examples of universal kernels and some sufficient conditions of universality.

Definition 5: Kernel $K : X \times X \rightarrow \mathbb{R}$ is strictly positive definite if it is a positive definite kernel and for any finite subset $X_N = \{x_1, \dots, x_N\} \subset X$ the corresponding matrix $K(X_N, X_N)$,

$$K(X_N, X_N) = \begin{bmatrix} K(x_1, x_1) & \cdots & K(x_1, x_N) \\ \vdots & \ddots & \vdots \\ K(x_N, x_1) & \cdots & K(x_N, x_N) \end{bmatrix},$$

is strictly positive definite.

Theorem 4. If positive definite kernel is strictly positive definite, it is a universal kernel.

The following positive definite kernels are the examples of the universal kernels:

- Exponential kernel $K(x, t) = \exp\left(-\frac{\|x-t\|^2}{2\sigma^2}\right)$,
- Survival kernel $K(x, t) = \prod_{i=1}^n \min(x_i, t_i)$,
- $K(x, t) = (\beta + \gamma \|x - t\|^2)^{-\alpha}$, $\beta > 0, \gamma > 0, \alpha > 0$.

13.2.12. Why Span? Representer Theorem

Here, we will present a very strong and a very general result, which is essential for the kernel learning and which extends the possibility of applying the “kernel trick” to a very wide range of machine learning models.

Theorem 5 (Representer theorem (Schölkopf *et al.*, 2001)). Let X be a non-empty set, K is a positive definite kernel over X and a reproducing kernel of RKHS \mathcal{H} of functions over X . Given a training sample $X_N = \{(x_1, y_1), \dots, (x_N, y_N)\}$, $x_i \in X$, $y_i \in \mathbb{R}$, $i = 1, \dots, N$, a strictly

monotonically increasing function $g : [0, +\infty) \rightarrow \mathbb{R}$. Then, for any arbitrary empirical risk function $E : (\mathbf{X} \times \mathbb{R}^2)^N \rightarrow \mathbb{R}$ if f^* is a solution of optimization problem

$$E((x_1, y_1, f(x_1)), \dots, (x_N, y_N, f(x_N))) + g(\|f\|_H) \rightarrow \min_{f \in \mathcal{H}}$$

then f^* has the structure

$$f^*(\cdot) = \sum_{i=1}^N \alpha_i K(\cdot, x_i).$$

Summand $g(\|f\|)$ in the functional under optimization corresponds to regularization term that is usually added for noise-stability purposes.

The theorem is very significant as it presents a parametric linear model of generally nonlinear function, which is an optimal solution for a very wide range of machine learning problems.

However, the theorem does not present the exact kernel form. It just states that such a reproducing kernel exists. Therefore, the theorem simplifies but does not remove the “model selection” problem.

13.3. Kernel Models and Applications

In this part of the chapter, we discuss kernel models, their properties and their applications to regression, density estimation, dimensionality reduction, filtering and finally clustering. Kernels provide elegant approach for incorporating nonlinearity into linear models, dramatically broadening the area of application for many algorithms, such as, linear regression, PCA, k -means, etc.

13.3.1. Kernel Density Estimation

One of the common problems in data mining is the probability density estimation aiming to restore an unknown probability distribution function basing on some given data points, which are supposed to be generated from the target distribution.

Formally, let $\mathbf{X}_N = \{x_1, \dots, x_N\}$ be an i.i.d. sample in \mathbb{R}^n drawn from an unknown distribution $p(x)$. The aim is to build an estimate distribution $\hat{p}(x)$. In general, $\hat{p}(x)$ should be a consistent estimate at every point.

Such problems appear as a subtask in many different practical applications such as classification, clustering, etc. Kernel models are widely used for probability density functions (pdf) estimation.

Kernel density estimation is closely related to the histogram-based density estimation approach which is also referred as histogram method. For one-dimensional data, the estimator is calculated as

$$\hat{p}_h(x) = \frac{1}{N} \frac{\sum_{i=1}^N I[x - \frac{h}{2} < x_i < x + \frac{h}{2}]}{h},$$

where h is the bandwidth parameter or window size, I denotes indicator function, i.e.,

$$I[c] = \begin{cases} 1, & \text{if } c \text{ is true;} \\ 0, & \text{if } c \text{ is false.} \end{cases}$$

The estimate is very sensitive to the bandwidth parameter, as it may cause over-or underfitting. Examples of estimates for different bandwidths are shown in [Figure 13.2](#). The original pdf is solid line, while its estimates are dashed lines. Too small bandwidth $h = 0.1$ results in very rough approximation. Too large bandwidth $h = 2$ leads to oversmoothing. Bandwidth $h = 0.56$ seems to be a compromise between too small and too large windows.

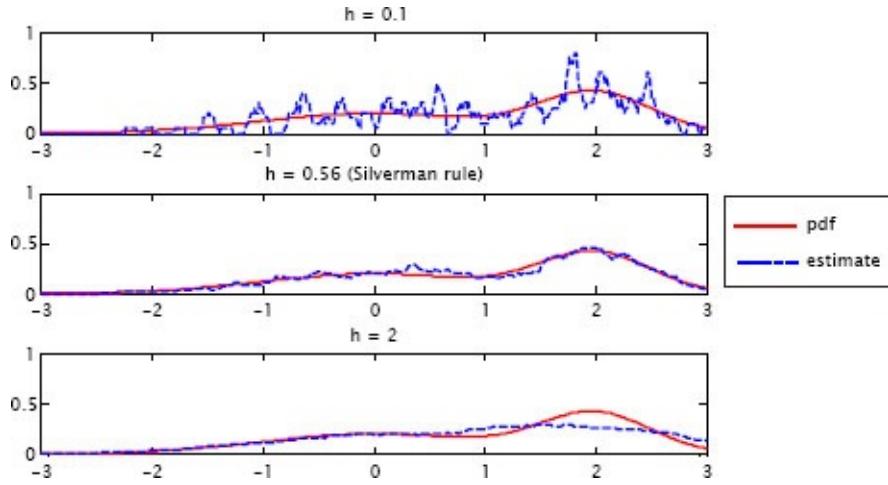


Figure 13.2: Kernel density estimation.

However, in the cases of small learning datasets histogram approach leads to unacceptable solutions. For this reason, the histogram function is often replaced by a nonnegative symmetric smooth function $K(x)$, normalized at 1, i.e.,

$$\int K(x)dx = 1.$$

Then the estimate becomes

$$\hat{p}_h(x) = \frac{1}{N} \sum_{i=1}^N K\left(\frac{x-x_i}{h}\right) = \frac{1}{N} \sum_{i=1}^N K_h(x - x_i),$$

where $K_h(x) = \frac{1}{h} K\left(\frac{x}{h}\right)$, usually referred as a scaled kernel. This approach is known as Parzen–Rosenblatt method. Notice that the histogram estimation is a special case of Parzen–Rosenblatt method (Parzen, 1962; Rosenblatt, 1956).

Such modification results in a smooth function. An example illustrating difference between regular histogram window and Gaussian kernel estimation is shown in [Figure 13.3](#).

As [Figure 13.2](#) shows, density estimation relies on good bandwidth parameter selection. There is a number (Jones *et al.*, 1996; Park and Marron, 1990; Park and Turlach, 1992; Sheather and Jones, 1991; Sheather, 1992) of different approaches but practically for one-dimensional data all of them are based on the result that the window size decreases as $O(N^{-\frac{1}{5}})$. Optimal bandwidth for kernel K estimating one-dimensional density p can be determined by minimizing asymptotic mean integrated squared error (AMISE):

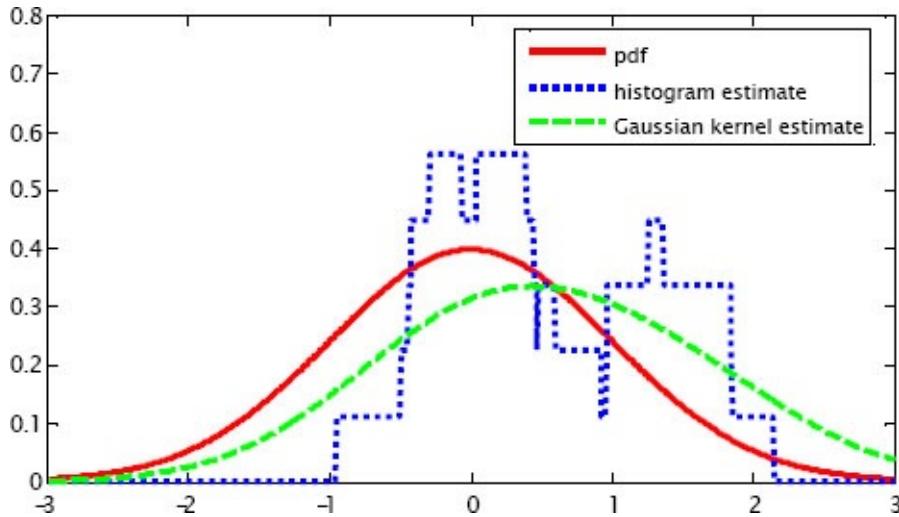


Figure 13.3: Kernel density estimation from 10 points, $h = 0.9$.

$$h^* = \frac{R(K)^{\frac{1}{5}}}{s(K)^{\frac{2}{5}} R\left(\frac{\partial^2 p}{\partial x^2}\right)^{\frac{1}{5}} N^{\frac{1}{5}}},$$

where

$$R(f) = \int f^2(x) dx,$$

$$s(f) = \int x^2 f(x) dx.$$

However, such estimate cannot be applied directly because it requires some prior knowledge about the estimated distribution (in particular, density $p(x)$). For some distributions this bandwidth estimate is known. For Gaussian distribution the following value, known as Silverman's rule of thumb (Silverman, 1986), is commonly used:

$$h^* = \left(\frac{4\hat{\sigma}^5}{3N}\right)^{\frac{1}{5}}.$$

It can be obtained from the asymptotic formula by direct substitution of the parametric variance by its estimate $\hat{\sigma}^2$.

13.3.2. The Nadaraya–Watson Kernel Regression

In 1964, (Nadaraya, 1964) and (Watson, 1964) proposed a method for kernel regression which is a nonparametric regression technique. Having a sample of pairs $\{(x_i, y_i)\}_{i=1}^N$, we look forward for finding the most expected value of y for any value of x . Assuming that pairs (x_i, y_i) come from some distribution f on $X \times Y$, where $x_i \in X, y_i \in Y \subseteq \mathbb{R}$, regression problem is formulated in terms of finding conditional expectation \bar{y} :

$$\bar{y} = \mathbb{E}(y|x) = \int yf(y|x)dy = \frac{\int yf(x, y)dy}{f(x)}.$$

Kernel density estimation helps us estimate the joint distribution $f(x, y)$ of two random variables (that can be, in fact, multivariate variables) x and y ,

$$\hat{f}(x, y) = \frac{1}{Nh_x h_y} \sum_{i=1}^N K\left(\frac{x - x_i}{h_x}\right) K\left(\frac{y - y_i}{h_y}\right),$$

and the marginal distribution $\hat{f}(x)$,

$$\hat{f}(x) = \frac{1}{Nh_x} \sum_{i=1}^N K\left(\frac{x - x_i}{h_x}\right).$$

Here h_x and h_y are corresponding bandwidths of the kernels.

Now these estimators are used to calculate the conditional expectation

$$\bar{y} = \frac{\int y \hat{f}(x, y) dy}{\hat{f}(x)} = \frac{\sum_{i=1}^N K\left(\frac{x - x_i}{h_x}\right) y_i}{\sum_{i=1}^N K\left(\frac{x - x_i}{h_x}\right)},$$

assuming that $\int y K\left(\frac{y - y_i}{h_y}\right) dy = h_y y_i$.

Bandwidth selection addresses the same approaches as were discussed in the previous section.

[Figure 13.4](#) illustrates dependency of Nadaraya–Watson estimator on the bandwidth parameter h_x . Again, too large bandwidth leads to over-smoothing, while too small bandwidth makes estimate excessively sensitive to outliers.

Speaking about kernel regression, we must mention a rather relative approach — locally weighted regression (LWR) (Atkeson *et al.*, 1997). LWR also makes use of kernels but in a different way, weighting examples by kernel function K . Linear regression weights w become a function of data, i.e.,

$$w(x) = \arg \min_w \sum_{i=1}^N K(x, x_i)(\langle w, x_i \rangle - y_i)^2,$$

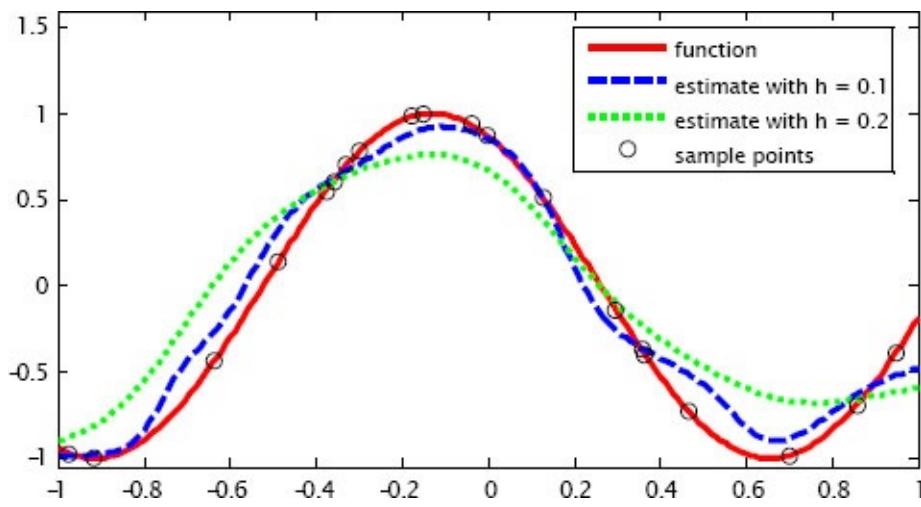


Figure 13.4: Nadaraya–Watson regression.

and

$$y(x) = \langle w(x), x \rangle.$$

Linear regression function $\langle w(x), x \rangle$ may be replaced with an arbitrary $f(x, w)$ resulting in

more complex models. It was shown that for a uniformly (or regularly) distributed data LWR is equivalent to kernel regression, while for non-uniform (irregular) data LWR outperforms kernel regression (Atkeson *et al.*, 1997; Jones *et al.*, 1999; Müller, 1987).

13.3.3. Kernel PCA

Another example of kernel methods is a kernel extension of principal component analysis (PCA). PCA is a feature aggregation technique aiming on finding a projection to an optimal linear subspace which occurs to be a span of eigenvectors of data matrix. Let

$$X_N = (x_1, \dots, x_N) \in \mathbb{R}^{n \times N}, \quad x_i \in \mathbb{R}^n, \quad i \in \{1, \dots, N\},$$

be a data matrix. PCA finds its decomposition in two matrices of lower rank

$$X_N \approx GW.$$

Columns of G represent an orthonormal basis of a linear L -dimensional subspace. W is a weight matrix determining projection onto this subspace. Subspace and orthogonal projection are chosen so that

$$\|X_N - GW\|^2 \rightarrow \min_{G, W}.$$

Initially linear, PCA approach can be easily extended to nonlinear cases (Schölkopf *et al.*, 1999).

Solution of PCA is based on eigenvalue decomposition of covariance matrix $C = X_N X_N^T$, which finds eigenvectors \mathbf{g} and eigenvalues λ ,

$$C\mathbf{g} = X_N X_N^T \mathbf{g} = \lambda \mathbf{g}.$$

For any kernel K there exists a map Φ and we can rewrite covariance matrix as

$$C_\Phi = \sum_{i=1}^N \Phi(x_i) \Phi(x_i)^T,$$

and find its eigenvalue decomposition. It gives us the first L eigenvectors $\mathbf{g}_1, \dots, \mathbf{g}_L$ so that

$$C_\Phi \mathbf{g}_l = \lambda_l \mathbf{g}_l.$$

As it is in a linear case of PCA eigenvectors are linear combinations of $\Phi(x_i)$, i.e.,

$$\mathbf{g}_l = \sum_{i=1}^N \alpha_{il} \Phi(x_i).$$

Vectors α_l can be calculated by eigenvalue decomposition of the pairwise inner product matrix $K = K [X_N, X_N]$, $K_{ij} = K(x_i, x_j)$, namely

$$K \alpha_l = \lambda_l \alpha_l.$$

By normalizing eigenvectors so that $\langle \mathbf{g}_l, \mathbf{g}_l \rangle_{\mathcal{H}} = 1$, we obtain a rule for finding nonlinear projection of any data vector x on each basis vector as

$$\langle \mathbf{g}_l, \Phi(x) \rangle_{\mathcal{H}} = \sum_{i=1}^N \alpha_{il} \langle \Phi(x_i), \Phi(x) \rangle_{\mathcal{H}}.$$

Note, we do not have to know an implicit form of the map Φ , just the corresponding kernel K , which is used for calculation of inner products.

Expression for C_Φ is tricky as, in general case, RKHS corresponding to the mapping Φ is infinite dimensional. Thus, transpose operator is incorrect. Then C_Φ can be understood as a linear operator

$$C_\Phi z = \sum_{i=1}^N \Phi(x_i) \langle \Phi(x_i), z \rangle_{\mathcal{H}}, \quad \forall z \in \mathcal{H}.$$

Kernel PCA allows finding nonlinear dependencies inside data. This property is illustrated in [Figure 13.5](#). In the original feature space, two-dimensional data points form three concentric circles, which are not linearly separated. Introduction of Gaussian or polynomial kernels into PCA allows obtaining linearly separable dataset.

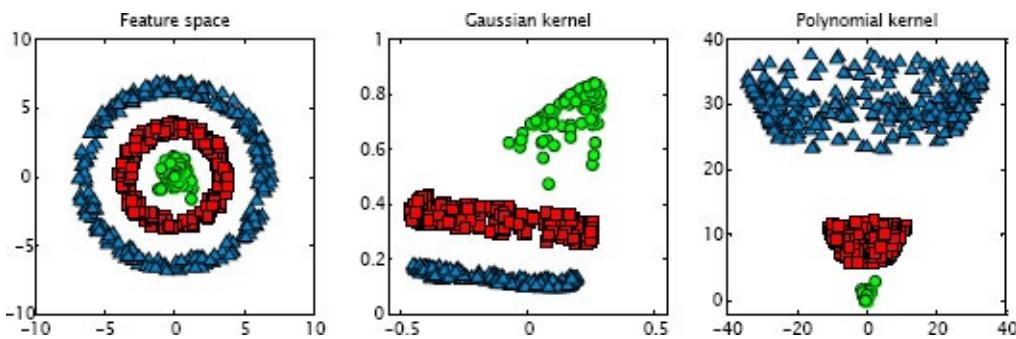


Figure 13.5: Kernel PCA with different kernels.

13.3.4. Kernel Linear Discriminant Analysis

Linear discriminant analysis relates to classification problem, i.e., for a given learning set

$$X_N = \{(x_1, y_1), \dots, (x_N, y_N)\},$$

$x_i \in \mathbb{R}^n$, $y_i \in Y$, where $Y = \{L_1, \dots, L_R\}$ is a set of class labels, the aim is to recover the labelling function $f: \mathbb{R}^n \rightarrow Y$.

Linear Discriminant Analysis finds the normal vector w of the “optimal” (in terms of given quality function) separating plane in the data space. The projection of the data on the resulting vector gives a good separation of classes.

For two-class classification problems, i.e., $R = 2$, the optimal vector is determined as a solution of the following problem

$$\frac{w^T S_B w}{w^T S_W w} \rightarrow \max_{w \in \mathbb{R}^n},$$

where

$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \in \mathbb{R}^{n \times n},$$

is the between-class covariance matrix,

$$S_W = \sum_{r=1}^2 \sum_{i:y_i=L_r} (\mu_r - x_i)(\mu_r - x_i)^T \in \mathbb{R}^{n \times n},$$

is the within-class covariance matrix, μ_r is an arithmetic mean of sample vectors of the class L_r ,

$$\mu_r = \frac{1}{N_r} \sum_{i:y_i=L_r} x_i,$$

where N_r is a number of samples in class L_r .

It can be shown that the resulting vector

$$w \propto S_W^{-1}(\mu_1 - \mu_2).$$

Similarly to the previous examples, kernel trick can be applied to the given model (Mika *et al.*, 1999). Suppose, there is a mapping between the initial data space and a Hilbert space: $\Phi : \mathbb{R}^n \rightarrow \mathcal{H}$, where

$$\langle \Phi(x), \Phi(t) \rangle_{\mathcal{H}} = K(x, t).$$

In this Hilbert space, the sample vectors class means are calculated as

$$\mu_r^\Phi = \frac{1}{N_r} \sum_{i:y_i=L_r} \Phi(x_i).$$

Therefore, an inner product of the mean with the map of any arbitrary vector $x \in \mathbb{R}^n$ is

$$\langle \Phi(x), \mu_r^\Phi \rangle_{\mathcal{H}} = \frac{1}{N_r} \sum_{i:y_i=L_r} \langle \Phi(x), \Phi(x_i) \rangle_{\mathcal{H}} = \frac{1}{N_r} \sum_{i:y_i=L_r} K(x, x_i).$$

Definition of the covariance matrix in the mapped feature space is more complex, as it defines a linear operator in \mathcal{H} . However, we can determine the linear operator by its values over the vectors from the span of Φ . For example, Hilbert space within-class covariance matrix is “schematically” defined as

$$S_W^\Phi = \sum_{r=1}^2 \sum_{i:y_i=L_r} \left[(\Phi(x_i) - \mu_r^\Phi) (\Phi(x_i) - \mu_r^\Phi)^T \right].$$

Suppose that

$$w^\Phi = \sum_{i=1}^N \alpha_i \Phi(x_i),$$

then

$$S_W^\Phi w^\Phi = \sum_{r=1}^2 \sum_{j:y_j=L_r} \langle \Phi(x_j) - \mu_r^\Phi, w^\Phi \rangle_{\mathcal{H}} [\Phi(x_j) - \mu_r^\Phi].$$

The coefficient $\langle \Phi(x_j) - \mu_r^\Phi, w^\Phi \rangle_{\mathcal{H}}$ can be easily computed using the properties of the inner product,

$$\langle \Phi(x_j) - \mu_r^\Phi, w^\Phi \rangle_{\mathcal{H}} = \sum_{i=1}^N \alpha_i [K(x_j, x_i) - \langle \Phi(x_i), \mu_r^\Phi \rangle_{\mathcal{H}}].$$

The result is a weighted sum of elements of \mathcal{H} . In a similar way a second-order function

$\langle w^\Phi, S_W^\Phi w^\Phi \rangle_{\mathcal{H}}$ can be calculated. The optimal values of α_i , $i = 1, \dots, N$, are still to be defined. Therefore, the resulting Kernel LDA optimization problem is to find maximum

$$\frac{\langle w^\Phi, S_B^\Phi w^\Phi \rangle_{\mathcal{H}}}{\langle w^\Phi, S_W^\Phi w^\Phi \rangle_{\mathcal{H}}} \rightarrow \max_{\alpha_i \in \mathbb{R}, i=1, \dots, N},$$

w.r.t.

$$w^\Phi = \sum_{i=1}^N \alpha_i \Phi(x_i).$$

Optimal solution of the stated Kernel LDA problem is

$$\alpha \propto Q^{-1}(M_1 - M_2),$$

where

$$M_r \in \mathbb{R}^N, \quad (M_r)_i = \frac{1}{N_r} \sum_{j:y_j=L_r} K(x_j, x_i),$$

$$Q = \sum_{r=1}^2 K_r (\mathbf{I} - \mathbf{E}_{N_r}) K_r^T,$$

$\mathbf{E}_{N_r} \in \mathbb{R}^{N_r \times N_r}$ is a matrix with elements equal to $\frac{1}{N_r}$,

$K_r \in \mathbb{R}^{N \times N_r}$, $(K_r)_{ij} = K(x_i, x_j)$, x_j – j th vector in the r th class.

As it was pointed previously, projection of the data on the optimal vector leads to the optimal separation. In terms of Kernel LDA, the projection of the $\Phi(x)$ on the vector w^Φ is defined by the inner product

$$\langle w^\Phi, \Phi(x) \rangle_{\mathcal{H}} = \sum_{i=1}^N \alpha_i K(x, x_i).$$

Separating property of the Kernel LDA method is illustrated in [Figure 13.6](#).

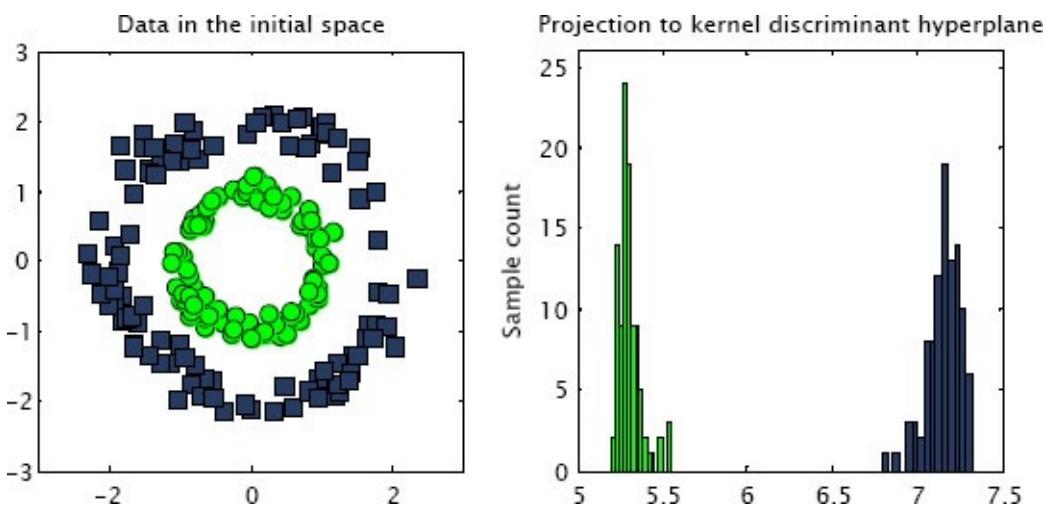


Figure 13.6: Kernel LDA projection example. Chart on the left demonstrates the distribution of the data in the initial space. Right chart demonstrates the histogram of the data samples projected on optimal vector in the feature map space.

13.3.5. Kernelization of an Arbitrary Linear Model

In fact, any linear model can be “kernelized” by replacing inner products $\langle x, t \rangle$ with kernel

values $K(\mathbf{x}, \mathbf{t})$. Here, we show the idea on an example of the ridge regression model. Recall the classical ridge regression with target function

$$\sum_{i=1}^N (y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle)^2 + \lambda \|\mathbf{w}\|^2 \rightarrow \min_{\mathbf{w}}.$$

Here, pairs (\mathbf{x}_i, y_i) belong to the training sample \mathbf{X}_N . This optimization problem can be solved analytically. With the help of linear algebra, we get

$$\mathbf{w} = \left(\sum_{i=1}^N \mathbf{x}_i \mathbf{x}_i^T + \lambda \mathbf{I} \right)^{-1} \left(\sum_{i=1}^N y_i \mathbf{x}_i \right).$$

Now, with an appropriate kernel we perform mapping $\mathbf{x}_i \rightarrow \Phi(\mathbf{x}_i)$. Therefore,

$$\sum_{i=1}^N (y_i - \langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle_{\mathcal{H}})^2 + \lambda \|\mathbf{w}\|^2 \rightarrow \min_{\mathbf{w} \in \mathcal{H}}.$$

Assuming that $\mathbf{w} = \sum_{i=1}^N \alpha_i \Phi(\mathbf{x}_i)$, we can reformulate the optimization problem as

$$\sum_{i=1}^N \left(y_i - \sum_{j=1}^N \alpha_j \langle \Phi(\mathbf{x}_j), \Phi(\mathbf{x}_i) \rangle_{\mathcal{H}} \right)^2 + \lambda \alpha^T K(\mathbf{X}_N, \mathbf{X}_N) \alpha \rightarrow \min_{\alpha \in \mathbb{R}^N}.$$

Here, we denote matrix of pairwise inner products of the training sample as $K(\mathbf{X}_N, \mathbf{X}_N)$,

$$[K(\mathbf{X}_N, \mathbf{X}_N)]_{ij} = K(\mathbf{x}_i, \mathbf{x}_j).$$

Then, the optimal solution of the stated problem is given by

$$\alpha = (K(\mathbf{X}_N, \mathbf{X}_N) + \lambda \mathbf{I})^{-1} \bar{\mathbf{y}}.$$

This result can be interpreted as a special case of the representer theorem application.

Again, for calculating y value for an arbitrary \mathbf{x} the exact form of the mapping Φ is not needed, we just have to know the corresponding kernel function.

$$y = \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle = \alpha^T K(\mathbf{X}_N, \mathbf{x}) = \sum_{i=1}^N \alpha_i K(\mathbf{x}_i, \mathbf{x}).$$

Vector $K(\mathbf{X}_N, \mathbf{x})$ consists of inner products of the training sample objects and the object being evaluated.

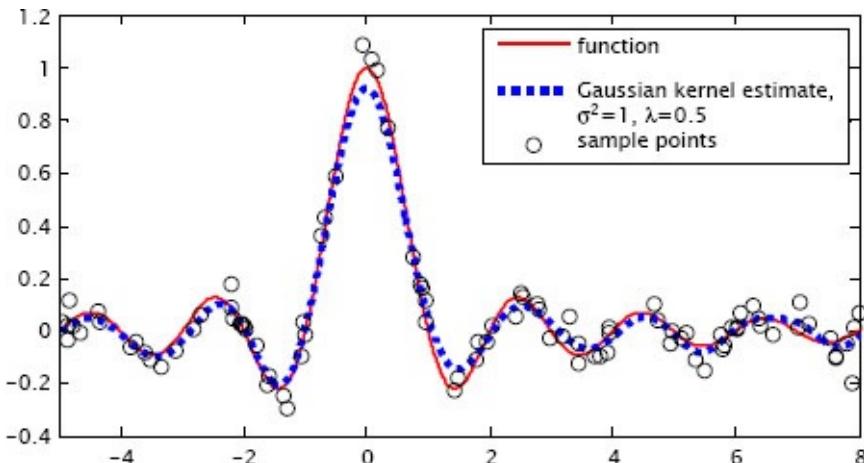


Figure 13.7: Kernel ridge regression with Gaussian kernel, $\sigma^2 = 1, \lambda = 0.5$.

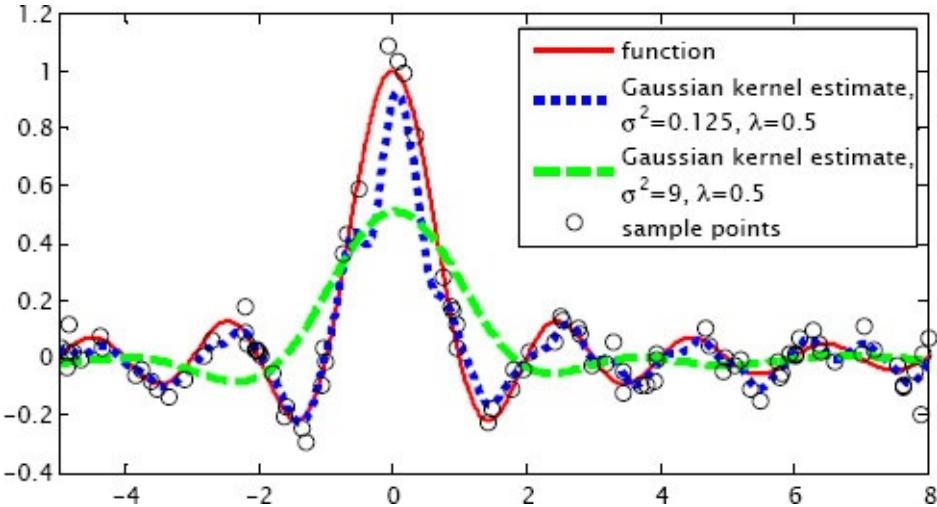


Figure 13.8: Kernel ridge regression with Gaussian kernel and different σ^2 .

Figure 13.7 shows an example of kernel ridge regression for a model

$$y = \frac{\sin \pi x}{\pi x} + \varepsilon,$$

where $\varepsilon \sim \mathcal{N}(0, s^2)$ is a normally distributed noise with zero mean and variance s^2 imposed on signal.

Kernel ridge regression significantly depends on the selection of kernel and its parameters. In the case of Gaussian kernel (Hastie *et al.*, 2003; Vapnik, 1995), too large values of spread σ^2 lead to an oversmoothed solution, while too small σ^2 results in an overfitting. These effects are shown in Figure 13.8.

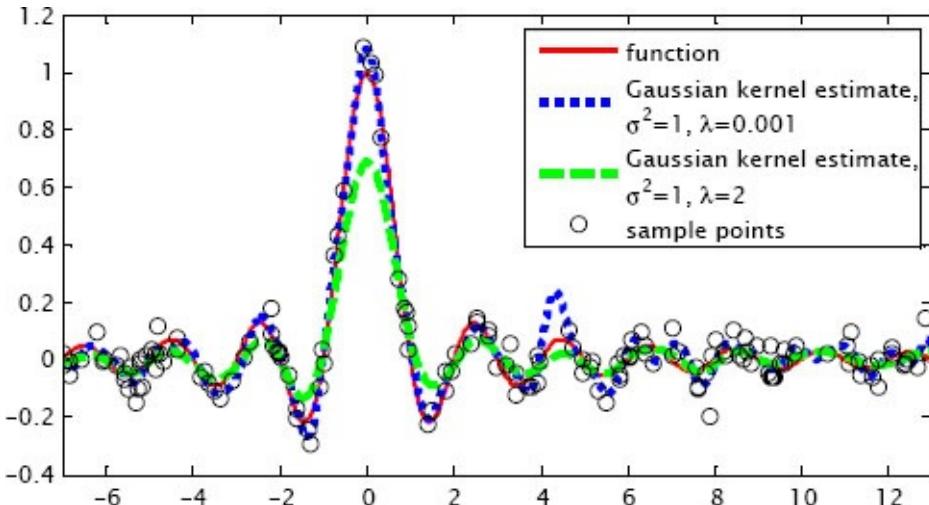


Figure 13.9: Kernel ridge regression with Gaussian kernel and different λ .

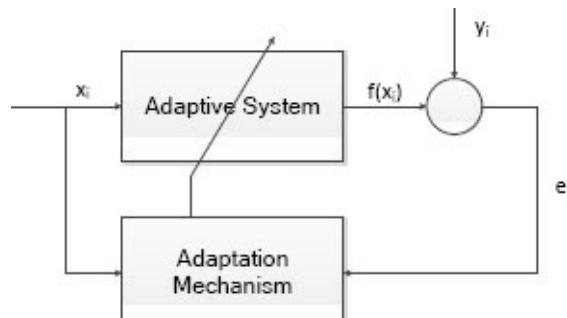


Figure 13.10: Adaptive system.

The same is with regression parameter λ : its growth will lead to smoothing and its decrease will lead to a very sensitive model, as it can be seen in [Figure 13.9](#). To tune the parameters of the kernel ridge regression, cross-validation technique may be applied (Exterkate *et al.*, 2013).

13.3.6. Adaptive Kernel Filtering

Described kernel models are applicable to adaptive filter design (Hayes, 1996). Adaptive kernel filtering admits online learning of the nonlinear filters. Learning procedure is computationally cheap, as the optimization is convex.

Consider a system that takes as input streaming data vectors $x_i \in \mathbb{R}^n$ drawn independently from an unknown probability distribution and produces output $f(x_i)$ aiming to predict corresponding outcome y_i . One of the possible representations of such a system is shown in [Figure 13.10](#).

The learning procedure is performed in online way. In case of the linear model, the output can be represented as an inner product

$$f(x_i) = \langle x_i, w \rangle.$$

Learning can be performed using any online fitting procedure, such as stochastic gradient descent (Hayes, 1996), Recursive Least Squares (Bottou, 1998), etc. But that approach limits the resulting filter output to a linear combination of the data vector components, i.e., a linear function.

This limitation can be overcome by the kernel trick. Hereafter, we use least squares functional

$$E(X_N, f) = \sum_{i=1}^N (f(x_i) - y_i)^2.$$

Consider stochastic gradient descent as a learning algorithm. Suppose that the target function is an inner product in the Hilbert space \mathcal{H} :

$$f(x) = \langle \Phi(x), w \rangle_{\mathcal{H}},$$

where $\Phi : \mathbb{R}^n \rightarrow \mathcal{H}$ is a map from the data space to a Hilbert space. Similar to the previous examples, we assume that

$$\forall x, t \in \mathbb{R}^n \langle \Phi(x), \Phi(t) \rangle_{\mathcal{H}} = K(x, t).$$

As a new learning pair (x_i, y_i) comes, parameter vector w_{i-1} is updated according to gradient of $(\langle \Phi(x_i), w_{i-1} \rangle_{\mathcal{H}} - y_i)^2$:

$$w_i = w_{i-1} - v(\langle \Phi(x_i), w_{i-1} \rangle_{\mathcal{H}} - y_i)\Phi(x_i) = w_{i-1} + \alpha_i \Phi(x_i).$$

Here, v is a gradient descent step. Therefore, the resulting function f can be expressed as

$$f(\mathbf{x}) = \sum_{i=1}^N \alpha_i K(\mathbf{x}, \mathbf{x}_i).$$

The overall algorithm is presented in [Table 13.1](#).

Parameter ν usually affects convergence and, specifically, convergence speed. The required condition for convergence is

$$\nu < \frac{N}{\xi_{\max}} \implies \nu < \frac{N}{\sum_{i=1}^N K(\mathbf{x}_i, \mathbf{x}_i)} = \frac{N}{\text{tr}(K(\mathbf{X}_N, \mathbf{X}_N))} < \frac{N}{\xi_{\max}}.$$

Table 13.1: Adaptive kernel filtering algorithm.

Kernel Least Mean Squares Algorithm

Input: $\mathbf{X}_N = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$, $\mathbf{x}_i \in \mathbb{R}^n$, $y_i \in \mathbb{R}$

Output: $\alpha_1, \dots, \alpha_N$

1. $\mathbf{w}_0 = \mathbf{0} \in \mathcal{H}$ — zero vector in the Hilbert space

2. for $\alpha_1, \dots, \alpha_N$

2.1. $e_i = (\langle \Phi(\mathbf{x}_i), \mathbf{w}_{i-1} \rangle_{\mathcal{H}} - y_i) = \left(\sum_{j=1}^{i-1} \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) - y_i \right)$

2.2 $\alpha_i = -\nu e_i$

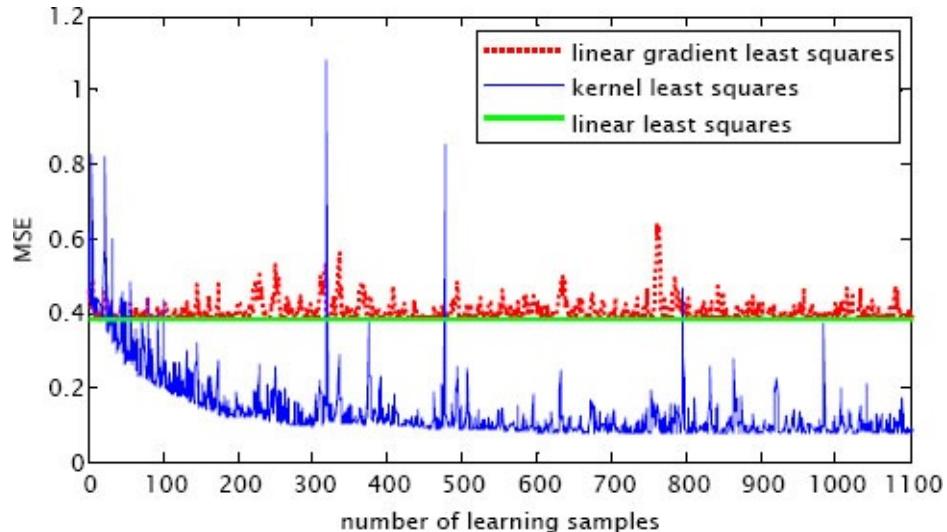


Figure 13.11: Adaptive kernel filtering.

Here, ξ_{\max} is the maximal eigenvalue of the matrix $K(\mathbf{X}_N, \mathbf{X}_N) \in \mathbb{R}^{N \times N}$ of samples' Hilbert space inner products, i.e.,

$$[K(\mathbf{X}_N, \mathbf{X}_N)]_{ij} = K(\mathbf{x}_i, \mathbf{x}_j).$$

An example of kernel least mean squares performance is presented in [Figure 13.11](#). The input of the system was generated from uniform distribution over unit 10-dimentional cube. The output was defined as the sine of the squared norm of the corresponding input vector, i.e.,

$$\mathbf{x}_i \in \mathbb{R}^{10}, \quad y_i = \sin(\|\mathbf{x}_i\|^2).$$

Thin and dotted lines represent the mean squared error (MSE) over the testing set depending on the number of learning samples used by an adaptive filter for kernel LMS

and linear stochastic LMS correspondingly. Thick line represents the MSE for the linear least squares algorithm, which is a theoretical minimum for the linear model. Kernel model converges slower, but significantly outperforms the linear model in terms of MSE.

However, one of the most significant drawbacks of the system is the order of the filter, which increases with each new accumulated learning data sample. This can be overcome by a number of approaches limiting the order of the filter (Liu *et al.*, 2010).

Kernel methods are widely applied in the area of adaptive filtering. There exist techniques for regularized regression fitting, kernel extended least squares algorithms and many other kernel data mining models.

13.3.7. Spectral Clustering

Spectral clustering (Balakrishnan *et al.*, 2011; Dhillon *et al.*, 2004; Luxburg, 2007; Ng *et al.*, 2001) is a clustering technique based on analysis of the data similarity matrix spectrum. Now it is considered to be one of the most popular clustering methods as it can be simply implemented and efficiently solved by standard linear algebra methods and outperforms many other clustering approaches in different domains (Balakrishnan *et al.*, 2011; Luxburg, 2007). Originally, the idea comes from weighted graph analysis, but it can be shown that it has connection to kernel methods.

The starting point of spectral clustering is building a similarity $N \times N$ matrix S of dataset $\{x_1, \dots, x_N\}$, where $s_{ij} > 0$ is similarity between x_i and x_j . These relationships can be represented in form of the similarity graph $G = (V, E)$, where vertex $v_i \in V$ corresponds to x_i and edge $e_{ij} \in E$ is weighted by s_{ij} . This graph can be fully connected, i.e., all $s_{ij} > 0$, or each vertex can be connected to K nearest neighbors only, i.e., all other edges have $s_{ij} = 0$, resulting in a sparse matrix (Luxburg, 2007). The problem of clustering then can be reformulated in terms of partitioning of the graph. And edges of the partition between different groups are expected to have low weights.

Following the graph theory, we denote a degree of a vertex v_i as

$$d_i = \sum_{j=1}^N s_{ij}.$$

The degree matrix D is a diagonal matrix with d_1, \dots, d_N on the diagonal.

Spectral clustering algorithms are based on eigenvalue decomposition of graph Laplacian matrix. There are several ways to calculate it.

1. The unnormalized graph Laplacian matrix (Mohar, 1991; Mohar *et al.*, 1997) is defined as

$$L = D - S.$$

2. The symmetric normalized graph Laplacian matrix (Chung, 1997) is defined as

$$L = D^{-\frac{1}{2}}(D - S)D^{-\frac{1}{2}} = \mathbf{I} - D^{-\frac{1}{2}}SD^{-\frac{1}{2}}.$$

3. The random walk normalized graph Laplacian matrix (Chung, 1997) is defined as

$$L = D^{-1}(D - S) = \mathbf{I} - D^{-1}S.$$

4. In (Ng *et al.*, 2001), the authors define graph Laplacian matrix as

$$L = D^{-\frac{1}{2}}SD^{-\frac{1}{2}},$$

which has the same eigenvectors as the symmetric normalized graph Laplacian matrix and eigenvalues λ_i , corresponding to $1 - \lambda_i$ of the latter.

After calculating the graph Laplacian matrix of the data, eigenvalue decomposition is performed. Ordinary decomposition may be replaced with a generalized eigenvalue decomposition as it is in Shi and Malik spectral clustering (Shi and Malik, 2000). Eigenvectors, which are vector columns, corresponding to K smallest eigenvalues form matrix $V^{N \times k}$ containing new descriptions of the original objects as vector rows.

Finally, these row vectors are clustered, for instance, using simple k -means algorithm. Ng *et al.* (2001) suggest normalizing vector rows to have norm 1.

Spectral clustering can be easily implemented and solved by standard linear algebra methods. Numerous experiments show that spectral clustering algorithms, though there is no theoretical proof, can be applied to clusters with arbitrary shapes (Luxburg, 2007). Intuitively, it follows from the fact that spectral clustering explores the spectrum of the similarity matrix. Thus, an appropriate similarity metric will lead to a reasonably good result.

Two examples of spectral clustering performance on datasets of complicated structures are given in [Figure 13.12](#).

In the end of the section, we show that spectral clustering has relationship with the kernel k -means,

$$\sum_{j=1}^k \sum_{x \in \pi_j} w(x) \|\Phi(x) - \mu_j\|^2 \rightarrow \min_{\pi_j, \mu_j},$$

where $\pi = \{\pi_j\}_{j=1}^k$ is a partitioning of data into clusters π_j . For fixed partition, cluster centers are calculated similarly to standard k -means,

$$\mu_j = \arg \min_{\mu} \sum_{x \in \pi_j} w(x) \|\Phi(x) - \mu\|^2$$

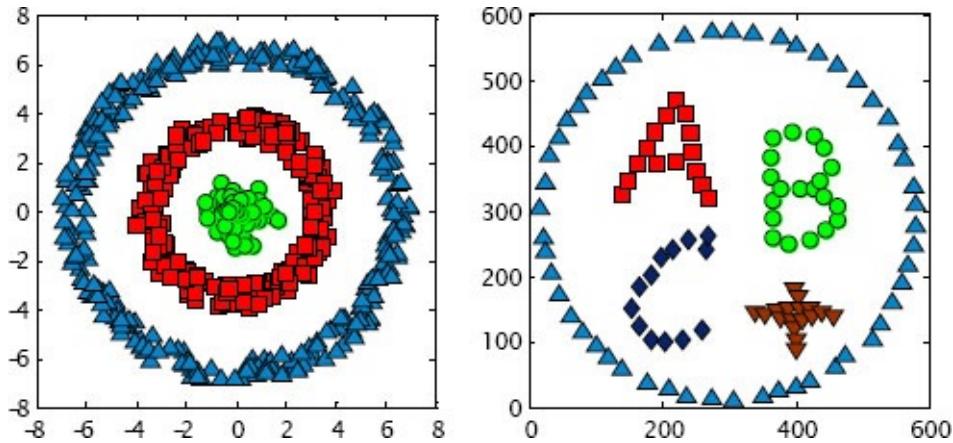


Figure 13.12: Spectral clustering result on different datasets.

and $\mathbf{x} \in X_N$. Function $w(\mathbf{x})$ assigns some nonnegative weight to each object \mathbf{x} . Mapping Φ is determined by some kernel function K . Note that there is no need in computing this mapping as $\|\cdot\|$ depends only on inner products computed by K .

Normalized spectral clustering, as having close connection to normalized graph cut problem, can be expressed as the matrix trace maximization problem (Dhillon *et al.*, 2004),

$$\text{trace}(V^T L V) \rightarrow \max_{V \in \mathbb{R}^{N \times k}, V^T V = \mathbf{I}},$$

where $L = D^{-\frac{1}{2}} S D^{-\frac{1}{2}}$.

Denoting diagonal matrix formed by weights $w(x_i), i = 1, \dots, N$, as W , weighted kernel k -means optimization problem takes the form of

$$\text{trace} \left(V^T W^{\frac{1}{2}} K(X_N, X_N) W^{\frac{1}{2}} V \right) \rightarrow \max_{V \in \mathbb{R}^{N \times k}, V^T V = \mathbf{I}},$$

which is close to that in normalized spectral clustering.

13.4. Support Vector Machines

Support vector machines are one of the most pervasive techniques for solving data mining problems, including classification, regression, anomaly detection, and metric learning. Initially invented as a linear classification problem solver in the mid-1960s by Soviet scientists Vapnik and Chervonenkis (1964), it was further developed to a framework of statistical methods with common optimization approaches. SVM plays a significant role in the scope of kernel machines due to the “sparsity” of the method, because the size of the learning set is the main limitation for such types of algorithms. Subsequent part of the chapter is composed as follows: we describe in more detail the SVM classifier (Vapnik, 1995) and its relation to kernel machines, introduce several types of support vector machines for binary classification, regression, and multiclass classification. Finally, we provide some approaches for anomaly detection, incremental SVM learning and metric fitting.

13.4.1. Support Vector Classifier

As it was pointed in the first chapter, the main idea behind the SVM is to build a maximal margin classifier. Suppose that we have a labelled sample $X_N = \{(x_1, y_1), \dots, (x_N, y_N)\}$, $x_i \in \mathbb{R}^n$, $y_i \in \{1, -1\}$, and we need to solve a two class classification problem. Assume that linear classification is considered, i.e., the algorithm’s decision rule can be presented as

$$f(x) = \text{sign}(\langle w, x \rangle - \rho),$$

where $w \in \mathbb{R}^n$, $\rho \in \mathbb{R}$. Therefore, the problem can be stated as finding of w and ρ , or defining a hyperplane that separates the objects of different classes. It is also supposed that the sample set is linearly separable meaning that

$$\exists w \in \mathbb{R}^n, \quad \exists \rho \in \mathbb{R} : \sum_{i=1}^N I[(\langle w, x_i \rangle - \rho)y_i \leq 0] = 0,$$

i.e., all the objects are classified correctly. Margin function is defined as

$$M(x_i, y_i) = (\langle w, x_i \rangle - \rho)y_i \begin{cases} \leq 0, & \text{if } f(x_i) \neq y_i, \\ > 0, & \text{if } f(x_i) = y_i. \end{cases}$$

However, there could be infinitely many possible ways to define the hyperplane as illustrated in [Figure 13.13](#) where two possible separation lines for two-dimensional classification problem are shown.

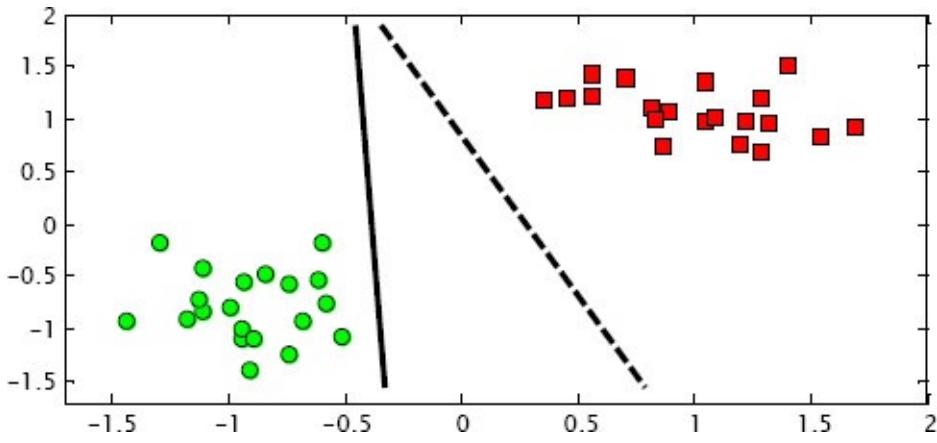


Figure 13.13: Different ways to define a hyperplane for the dataset.

Vladimir Vapnik *et al.* suggested “maximum margin” approach (Boser *et al.*, 1992), according to which the hyperplane is placed as far as possible from the closest objects of both classes. So, linear two-class SVM finds a “hyper-band” between two classes with maximal width.

Consider the maximal bandwidth problem in more detail. Actually, distance from a given point $x_0 \in \mathbb{R}^n$ to the plane $P(w, \rho)$, defined by the equation

$$\langle x, w \rangle - \rho = 0,$$

can be calculated as

$$d(P(w, \rho), x_0) = \frac{|\langle x_0, w \rangle - \rho|}{\|w\|}.$$

In order to find the best separating bandwidth, we maximize the distance from the closest data point in the learning set taking into account class separation constraints. Therefore, the optimization problem for SVM is stated as

$$\begin{aligned} & \max_{w, \rho} \min_{x_i \in X_N} \frac{|\langle x_i, w \rangle - \rho|}{\|w\|}, \\ & \text{w.r.t.} \\ & y_i(\langle x_i, w \rangle - \rho) > 0, \quad \forall (x_i, y_i) \in X_N. \end{aligned}$$

Note that for all $\alpha > 0$ terms αw , $\alpha \rho$ define the same separating hyperplane. So, we can strengthen the constraints of the problem:

$$\forall (x_i, y_i) \in X_N (\langle w, x_i \rangle - \rho) y_i \geq 1,$$

i.e., for the objects closest to the hyperplane the inequality turns into equality. It is possible to show that there should be at least two closest objects (with the same distance) from both of the classes. Suppose $P(w^*, \rho^*) \equiv P^*$ is the optimal. Let the closest vector from the class “−1” be denoted as x_{-1} , and one from the class “1” as x_{+1} . Without loss of generality, let us suppose that

$$d(P(w^*, \rho^*), x_{+1}) < d(P(w^*, \rho^*), x_{-1}).$$

Denote $\varepsilon = d(P^*, x_{-1}) - d(P^*, x_{+1})$ and consider a plane

$$P \left(w^*, \left(\rho^* - \frac{\varepsilon}{2} \right) \right) = P_\varepsilon^*.$$

This hyperplane fits to the learning set constraints (all the vectors are correctly classified), but the distance of each “−1” class sample is reduced by $\frac{\varepsilon}{2}$, the distance of each “+1” class sample is enlarged by $\frac{\varepsilon}{2}$ comparing to P^* . It is obvious that the object which is the closest to the hyperplane P_ε^* is still x_{+1} , but

$$d(P^*, x_{+1}) < d(P^*, x_{+1}) + \frac{\varepsilon}{2} = d(P_\varepsilon^*, x_{+1}),$$

that contradicts to “maximum distance” statement. Thus, objects from classes “1” and “−1”, closest to the hyperplane, are equidistant to it.

Therefore, the separation bandwidth is

$$h = \left\langle x_{+1} - x_{-1}, \frac{w}{\|w\|} \right\rangle = \frac{(\langle x_{+1}, w \rangle - \rho) - (\langle x_{-1}, w \rangle - \rho)}{\|w\|} = \frac{2}{\|w\|}.$$

This value is to be maximized, i.e.,

$$\frac{2}{\|w\|} \rightarrow \max_w.$$

The formulation is easily converted to the standard SVM optimization problem

$$\begin{aligned} \frac{1}{2} \|w\|^2 &\rightarrow \min_{w, \rho} \\ \text{w.r.t.} \\ y_i(\langle w, x_i \rangle - \rho) &\geq 1, \quad i = 1, \dots, N. \end{aligned}$$

However, it does not allow solving classification problem for linearly non-separable classes. From the mathematical perspective, it comes from the point that the set defined by constraints $y_i(\langle w, x_i \rangle - \rho) \geq 1$ may be empty. In order to overcome this issue, the so-called “slack variables” ξ_i are introduced.

$$\begin{aligned} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i &\rightarrow \min_{w, \rho, \xi} \\ \text{w.r.t.} \\ y_i(\langle w, x_i \rangle - \rho) &\geq 1 - \xi_i, \\ \xi_i &\geq 0, \quad i = 1, \dots, N. \end{aligned}$$

In this case, the set of constraints defines an open set (because $\forall w, \rho$ there exist slack variables ξ_i high enough for constraints to be met). The problem is defined as quadratic-programming optimization. Usually it is solved in its dual formulation. If we write down the Lagrangian and derive the Karush–Kuhn–Tucker (KKT) conditions we get the dual optimization problem,

$$-\frac{1}{2}\alpha Q\alpha^T + \sum_{i=1}^N \alpha_i \rightarrow \max_{\alpha}$$

w.r.t.

$$\begin{cases} 0 \leq \alpha_i \leq C \quad \forall i = 1, \dots, N, \\ \sum_{i=1}^N \alpha_i y_i = 0, \end{cases}$$

where $\alpha \in \mathbb{R}^N$ is a vector of Lagrangian multipliers, $Q_{ij} = y_i y_j \langle x_i, x_j \rangle$.

KKT conditions play an essential role in solving optimization problems and explaining the properties of the approach. We focus on two main consequences of KKT conditions for the SVM.

The first KKT condition leads to an explicit representation of vector w ,

$$w = \sum_{i=1}^N y_i \alpha_i x_i,$$

which can be interpreted as special case of the representer theorem with Lagrangian regarded as a regularized loss function. One of the key properties of the SVM is that most of the α_i will be reduced to zero. Only a relatively small subset of samples has corresponding nonzero alpha entries. Such samples are called *support vectors*.

High number of zero entries in α can be explained by second consequence of KKT conditions. It can be shown that vectors in the learning set can be split into three categories

$$\begin{cases} M(x_i, y_i) > 1, \quad \alpha_i = 0, & \xi_i = 0 : C - \text{group}, \\ M(x_i, y_i) = 1, \quad 0 < \alpha_i < C, & \xi_i = 0 : M - \text{group}, \\ M(x_i, y_i) < 1, \quad \alpha_i = C, & \xi_i > 0 : E - \text{group}, \end{cases}$$

where $M(x_i, y_i) = (\langle w, x_i \rangle - \rho)y_i$. *C-group (correct)* defines a subset of vectors from X_N lying beyond the boundaries of the separation band and, thus, are well-separated. *M-margin* and *E-error* groups are the subsets of the vectors lying on the border and within the separation band, respectively. These groups are related to less separable vectors (*E-group* may contain misclassified vectors).

Therefore, all the vectors that lie beyond the boundaries of the separation band (*C-group*) have zero alpha entries. Taking into account that the algorithm aims to minimize the sum of ξ_i , the number of vectors in *M-* and *E-groups* should be relatively small. Otherwise it could be a sign of poor classification model selection..

Other explanation for the sparsity of the SVM can be referred as “geometrical”. Constraints of the primal optimization problem define a set, limited by a hyper-polyline (maybe, hyper-polygon) in the space of w . Each vector x_i in the sample set determines a linear function in the space of w

$$f_i(\mathbf{w}, \rho) = y_i \langle \mathbf{x}_i, \mathbf{w} \rangle - y_i \rho - 1,$$

defining a half-space $f_i(\mathbf{w}, \rho) \geq 0$, but not each of linear functions is involved into forming the broken hyperplane, which represents a border of the set

$$\{\mathbf{w}, \rho | f_i(\mathbf{w}, \rho) \geq 0 \quad \forall i = 1, \dots, N\}.$$

An example with a sample set of five vectors is illustrated in [Figure 13.14](#).

It is assumed that each hyperplane defines a half-space above it. Only three lines of five construct the border polyline.

13.4.2. Kernel Trick: Examples

Let us now consider application of kernels to SVM more carefully. The so-called “*kernel trick*” (Aizerman *et al.*, 1964) is a powerful tool that introduces nonlinearity into a model allowing to separate data that cannot be separated linearly.

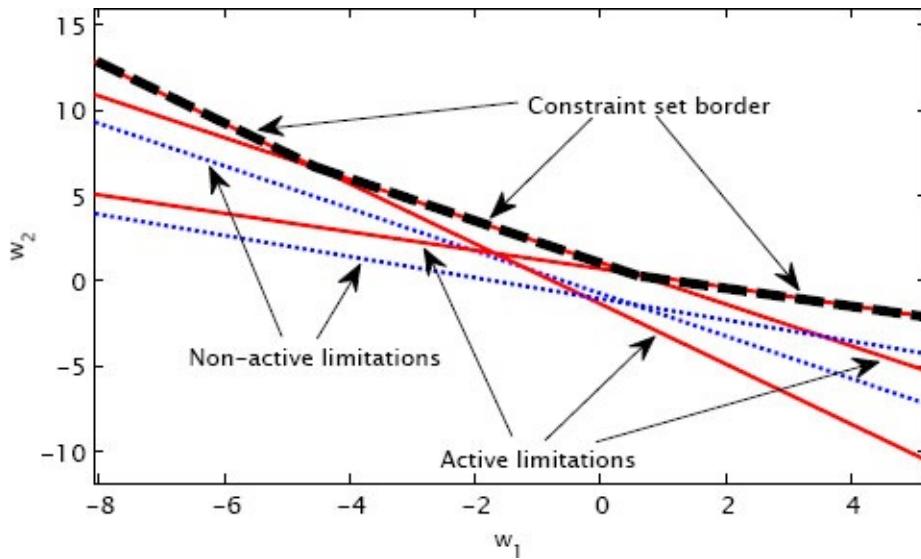


Figure 13.14: Active and nonactive constraints.

In linear SVM, we have

$$\begin{aligned} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i &\rightarrow \min_{\mathbf{w}, \xi, \rho}, \\ \text{w.r.t.} \quad y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle - \rho) &\geq 1 - \xi_i, \\ \xi_i &\geq 0, \quad i = 1, \dots, N. \end{aligned}$$

Slack variables ξ_i make the problem consistent even for linearly non-separable cases, but still linear hyperplane can have poor discriminative ability with classes being separated by a nonlinear surface. We may hope that with a map $\Phi : \mathcal{X} \rightarrow \mathcal{H}$ the problem will become linearly separable in the feature space \mathcal{H} . Due to the fact different feature space that target function and conditions in SVM depend only on inner products thus they can be calculated in \mathcal{H} using the corresponding kernel function K without the evaluation of map Φ ,

$$K(\mathbf{x}, \mathbf{t}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{t}) \rangle_{\mathcal{H}}.$$

In other words, we are trying to find such a feature space \mathcal{H} with corresponding

reproducing kernel K so that the dividing function f in the input space can be represented as a linear combination

$$f(\mathbf{x}) = \sum_{i=1}^N \hat{\alpha}_i K(\mathbf{x}, \mathbf{x}_i).$$

The dual form of the linear SVM provides an elegant solution for determining linear coefficients $\hat{\alpha}_i$. Actually, from KKT conditions, we have

$$\mathbf{w} = \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i.$$

The image of \mathbf{w} to the Hilbert space can be written as

$$\mathbf{w}_{\mathcal{H}} = \sum_{i=1}^N \alpha_i y_i \Phi(\mathbf{x}_i).$$

Then the inner product $\langle \mathbf{w}, \mathbf{x} \rangle$ that determines a linear separating plane in the initial space is replaced by a nonlinear separating surface determined by function

$$f(\mathbf{x}) = \langle \mathbf{w}_{\mathcal{H}}, \Phi(\mathbf{x}) \rangle_{\mathcal{H}} = \sum_{i=1}^N \alpha_i y_i \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}_i) \rangle_{\mathcal{H}} = \sum_{i=1}^N \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i).$$

Then the following quadratic program is to be solved instead of the initial one.

$$\begin{aligned} & \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \rightarrow \max_{\alpha}, \\ & \text{w.r.t.} \\ & 0 \leq \alpha_i \leq C, \\ & \sum_{i=1}^N \alpha_i y_i = 0. \end{aligned}$$

The computation effectiveness of the kernel SVM arises from a relatively small number of non-zero α_i , corresponding to support vectors. It dramatically reduces the number of kernel function calculations.

Here, we illustrate the kernel SVM technique by two examples. The first one in [Figure 13.15](#) shows the ability of polynomial kernel

$$K(\mathbf{x}, \mathbf{t}) = (\langle \mathbf{x}, \mathbf{t} \rangle + 1)^2,$$

to restore quadratic discriminant line. True class labels of the data are marked with different color and form. Quadratic discriminant line of the kernel SVM is in red. Filled points are support vectors.

The second example in [Figure 13.16](#) illustrates performance of Radial Basis Function SVM on data come from the mixture of Gaussians. Once again, true class

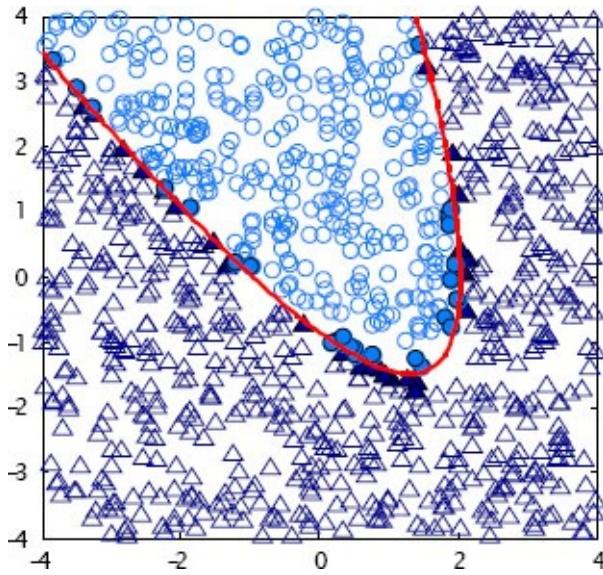


Figure 13.15: Polynomial kernel SVM example.

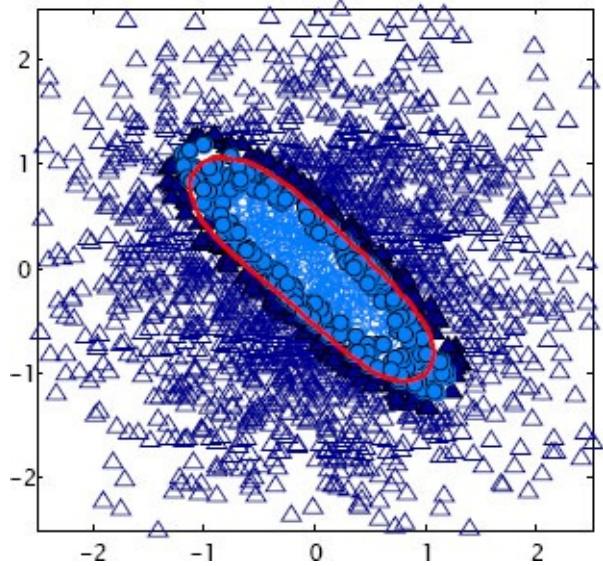


Figure 13.16: RBF SVM example.

labels of the data are marked with color and form while discriminant line of RBF SVM is in red filled points represent support vectors.

13.4.3. v-SVM and C-SVM

As we mentioned before, we can state the optimization problem to find an optimal margin in the form

$$\begin{aligned} \frac{1}{2} \|w\|^2 &\rightarrow \min_{w, \rho}, \\ y_i(\langle w, x_i \rangle - \rho) &\geq 1, \quad i = 1, \dots, N. \end{aligned}$$

This statement, however, does not allow data to be misclassified, but the dataset is not guaranteed to be exactly divisible in the proposed model.

For this purpose, we relax the constraints to allow data to be misclassified. We introduce slack variables, penalizing misclassifications,

$$\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i \rightarrow \min_{\mathbf{w}, \rho, \xi},$$

w.r.t.

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle - \rho) \geq 1 - \xi_i,$$

$$\xi_i \geq 0, \quad i = 1, \dots, N,$$

where C is a parameter, controlling the value of misclassification penalty. The constraints imposed on each of the dataset elements are often referred as *soft margin constraints*, in contrast to *hard margin constraints*, imposed on the problem.

Then we can write the Lagrangian of the problem as

$$L(\mathbf{w}, \rho, \xi, \alpha, \beta)$$

$$= \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \alpha_i (y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle - \rho) - 1 + \xi_i) - \sum_{i=1}^N \beta_i \xi_i,$$

where α_i, β_i are Lagrangian multipliers, with corresponding set of KKT conditions:

$$\alpha_i (y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle - \rho) - 1 + \xi_i) = 0,$$

$$\beta_i \xi_i = 0,$$

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle - \rho) - 1 + \xi_i \geq 0,$$

$$\alpha_i \geq 0, \quad \beta_i \geq 0, \quad \xi_i \geq 0.$$

Differentiating the Lagrangian, we obtain

$$\frac{\partial L(\mathbf{w}, \rho, \xi, \alpha, \beta)}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i,$$

$$\frac{\partial L(\mathbf{w}, \rho, \xi, \alpha, \beta)}{\partial \rho} = 0 \Rightarrow \sum_{i=1}^N \alpha_i y_i = 0,$$

$$\frac{\partial L(\mathbf{w}, \rho, \xi, \alpha, \beta)}{\partial \xi_i} = 0 \Rightarrow C - \beta_i = \alpha_i,$$

and then we switch to the dual problem

$$\tilde{L}(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \rightarrow \max_{\alpha},$$

w.r.t.

$$0 \leq \alpha_i \leq C, \quad i = 1, \dots, N,$$

$$\sum_{i=1}^N \alpha_i y_i = 0.$$

This approach is named *C-SVM* (Cortes and Vapnik, 1995). The dual representation gives a rise to an alternative technique to tackle with misclassification, *v-SVM* (Schölkopf *et al.*, 2000). For this approach, the primal problem is stated as

$$\frac{1}{2} \|\mathbf{w}\|^2 - v\gamma + \frac{1}{N} \sum_{i=1}^N \xi_i \rightarrow \min_{\mathbf{w}, \rho, \gamma}$$

w.r.t.

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle - \rho) \geq \gamma - \xi_i,$$

$$\xi_i \geq 0, \quad i = 1, \dots, N,$$

and the dual problem is

$$\tilde{L}(\alpha) = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \rightarrow \max_{\alpha}$$

w.r.t.

$$0 \leq \alpha_i \leq \frac{1}{N}, \quad i = 1, \dots, N,$$

$$\sum_{i=1}^N \alpha_i y_i = 0, \quad \sum_{i=1}^N \alpha_i \geq v.$$

Here, the parameter v gives a lower bound for the fraction of support vectors, as well as an upper bound for the fraction of margin errors in case of $\gamma > 0$. Actually, the first statement can be justified by noting that the maximum contribution to the last KKT inequality of each of the vectors is $\frac{1}{N}$, thus the overall count of support vectors cannot be lower than vN . The last statement can be proven using the KKT conditions of the primal problem (Schölkopf *et al.*, 2000). This formulation of the task can be proved to be equivalent to C-SVM with $C = \frac{1}{Nv}$, if $\gamma > 0$.

13.4.4. ϵ -SVR and v -SVR

The support vector machines do not restrict only to classification problems, but can be addressed to regression problems as well. This family of methods is referred as Support Vector Regression (SVR).

Given a learning dataset $X_N = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$, $y_i \in \mathbb{R}$, we aim to find a regression function $\hat{f}(\mathbf{x})$, approximating an unknown target function $f: X \rightarrow \mathbb{R}$, in the form

$$\hat{f}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle - \rho.$$

We state the problem as minimization of the regularized error function

$$E_1(\hat{f}, \mathbf{w}) = \frac{1}{2} \sum_{i=1}^N E\{\hat{f}(\mathbf{x}_i) - y_i\} + \frac{\lambda \|\mathbf{w}\|^2}{2}, \quad \lambda \geq 0.$$

Here, we do not use a standard error function

$$E\{\hat{f}(\mathbf{x}_i) - y_i\} = (\hat{f}(\mathbf{x}_i) - y_i)^2,$$

as we aim to obtain a sparse solution, but state

$$E\{\hat{f}(\mathbf{x}_i) - y_i\} = \begin{cases} |\hat{f}(\mathbf{x}_i) - y_i| - \epsilon, & \text{if } |\hat{f}(\mathbf{x}_i) - y_i| > \epsilon, \\ 0, & \text{else.} \end{cases}$$

Again, the statement is augmented by slack variables, penalizing the exit from

$$|\hat{f}(x_i) - y_i| < \varepsilon$$

Introducing slack variables, we turn the problem into the form

$$\begin{aligned} E(\xi^1, \xi^2, w) &= C \sum_{i=1}^N \{\xi_i^1 + \xi_i^2\} + \frac{\|w\|^2}{2} \rightarrow \min_{\xi^1, \xi^2, w}, \\ \hat{f}(x_i) - y_i &\geq -\varepsilon - \xi_i^1, \quad \xi_i^1 \geq 0 \quad \forall i \in 1, \dots, N, \\ \hat{f}(x_i) - y_i &\leq \varepsilon + \xi_i^2, \quad \xi_i^2 \geq 0 \quad \forall i \in 1, \dots, N. \end{aligned}$$

Here, $C = \frac{1}{2}\lambda^{-1}$.

As before, we introduce Lagrangian to turn the problem into dual representation

$$\begin{aligned} L(w, \xi^1, \xi^2, \beta^1, \beta^2, \alpha^1, \alpha^2, \rho), \\ = C \sum_{i=1}^N \{\xi_i^1 + \xi_i^2\} + \frac{\|w\|^2}{2} - \sum_{i=1}^N \{\beta_i^1 \xi_i^1 + \beta_i^2 \xi_i^2\} \\ - \sum_{i=1}^N \alpha_i^1 \{\varepsilon + \xi_i^1 + \hat{f}(x_i) - y_i\} - \sum_{i=1}^N \alpha_i^2 \{\varepsilon + \xi_i^2 - \hat{f}(x_i) + y_i\}. \end{aligned}$$

Differentiating, we obtain

$$\begin{aligned} \frac{\partial L}{\partial w} = \mathbf{0} &\Rightarrow w = \sum_{i=1}^N x_i (\alpha_i^1 - \alpha_i^2), \\ \frac{\partial L}{\partial \xi_i^1} = 0 &\Rightarrow C = \alpha_i^1 + \beta_i^1, \\ \frac{\partial L}{\partial \xi_i^2} = 0 &\Rightarrow C = \alpha_i^2 + \beta_i^2, \\ \frac{\partial L}{\partial \rho} = 0 &\Rightarrow \sum_{i=1}^N (\alpha_i^1 - \alpha_i^2) = 0. \end{aligned}$$

Finally, utilizing the KKT conditions, we can obtain the final expression of the dual problem:

$$\begin{aligned} \tilde{L}(\alpha^1, \alpha^2) &= -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N (\alpha_i^1 - \alpha_i^2)(\alpha_j^1 - \alpha_j^2) \langle x_i, x_j \rangle \\ &\quad - \varepsilon \sum_{i=1}^N (\alpha_i^1 + \alpha_i^2) + \sum_{i=1}^N (\alpha_i^1 - \alpha_i^2) y_i \rightarrow \max_{\alpha^1, \alpha^2}, \\ \alpha_i^1 &\geq 0, \quad \alpha_i^1 \leq C, \quad \alpha_i^2 \geq 0, \quad \alpha_i^2 \leq C. \end{aligned}$$

Also it should be mentioned, that as the Lagrangian is dependent of scalar products, one can apply standard kernel trick.

The described approach is referred as ε -SVR (Drucker *et al.*, 1996).

Similar to v -SVM, we can define the v -SVR method (Schölkopf *et al.*, 2000). The primal problem for it is formulated as follows:

$$\begin{aligned}
E(\xi^1, \xi^2, w, \varepsilon) &= \frac{C}{N} \sum_{i=1}^N \{\xi_i^1 + \xi_i^2\} + C\varepsilon v + \frac{\|w\|^2}{2} \rightarrow \min_{\xi^1, \xi^2, w, \varepsilon}, \\
\hat{f}(x_i) - y_i &\geq -\varepsilon - \xi_i^1, \quad \xi_i^1 \geq 0 \quad \forall i \in 1, \dots, N, \\
\hat{f}(x_i) - y_i &\leq \varepsilon + \xi_i^2, \quad \xi_i^2 \geq 0 \quad \forall i \in 1, \dots, N, \\
\varepsilon &\geq 0.
\end{aligned}$$

The dual problem in this case is written as

$$\begin{aligned}
L(\alpha^1, \alpha^2) &= -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N (\alpha_i^1 - \alpha_i^2)(\alpha_j^1 - \alpha_j^2) \langle x_i, x_j \rangle \rightarrow \max_{\alpha^1, \alpha^2}, \\
0 \leq \alpha_i^1 &\leq \frac{C}{N}, \quad 0 \leq \alpha_i^2 \leq \frac{C}{N}, \\
\sum_{i=1}^N (\alpha_i^1 - \alpha_i^2) &= 0, \quad \sum_{i=1}^N (\alpha_i^1 + \alpha_i^2) \leq vC.
\end{aligned}$$

ε -SVR regression for the function $y = x^2$ is shown in [Figures 13.17–13.19](#). We compare the models for different count of the learning points. The support vectors are marked by black boxes around learning points. The Gaussian radial basis kernel is chosen for this example. Prediction gets better while increasing the number of the points. Here, we can see that sparsity of the solution is exposed again. In [Figure 13.19](#), we have 30 learning points, but only 18 of them are support vectors.

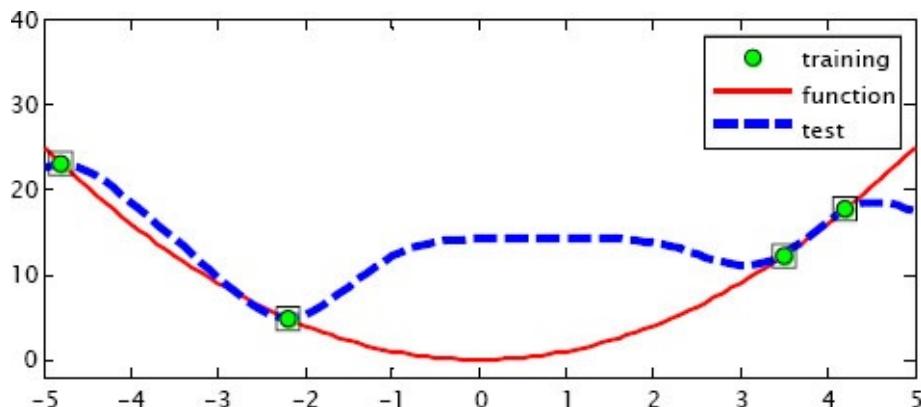


Figure 13.17: ε -SVR regression of the function $y = x^2$, 4 learning points.

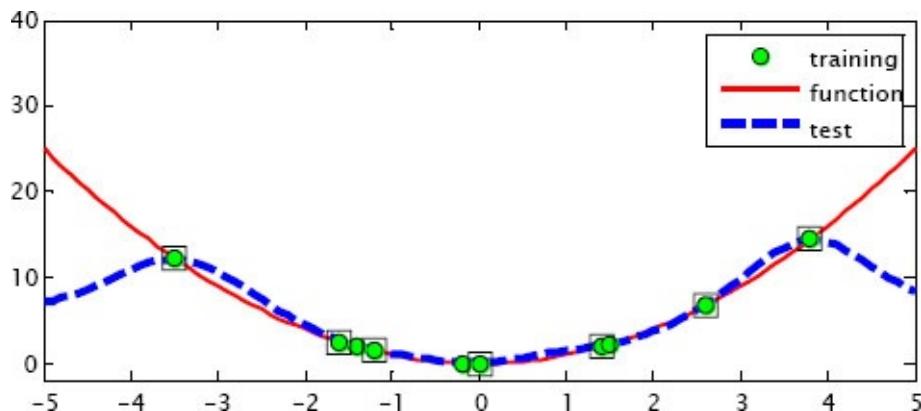


Figure 13.18: ε -SVR regression of the function $y = x^2$, 10 learning points.

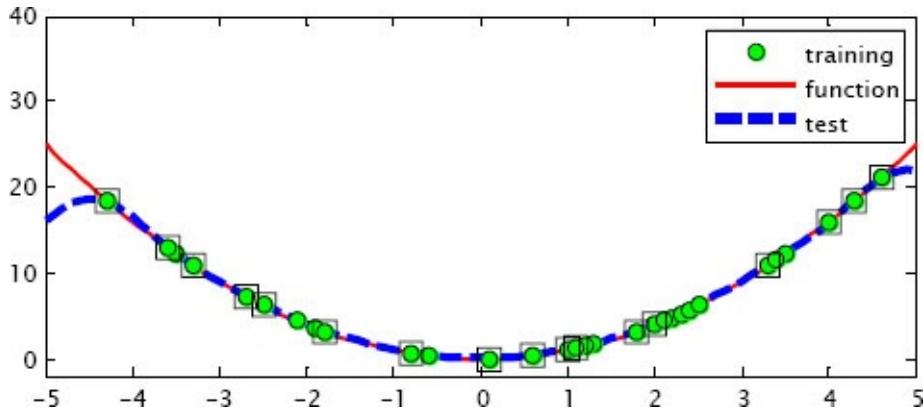


Figure 13.19: ϵ -SVR regression of the function $y = x^2$, 30 learning points.

13.4.5. Multiclass SVM

Multiclass SVM classifier is assumed to assign labels to the input objects, where the set of labels is finite and can consist of two or, generally, more elements. Formally, the classifier is trained over the sample set

$$X_N = \{(x_1, y_1), \dots, (x_N, y_N)\}, \quad x_i \in \mathbb{R}^n, \quad y_i \in \{L_1, \dots, L_R\}, \quad R \geq 2.$$

SVM classification model was initially adapted for two-class problems, and there is no common approach for multiclass classification problems.

We present three main approaches for SVM adaptation for multiclass classification problems.

- One-versus-all approach: For each of the R classes, two-class SVM classifier is trained to separate the given class from all the rest classes. If we denote the model separating r th class as

$$y_r(x) = \text{sign}(f_r(x)) = \text{sign}\left(\sum_{i=1}^N y_i^r \alpha_i^r K(x, x_i) + \rho_r\right),$$

$$y_i^r = \begin{cases} 1, & \text{if } y_i = L_r, \\ -1, & \text{otherwise,} \end{cases}$$

the classification of the input vector x is performed using winner-takes-all strategy:

$$y(x) = L_{r^*}, \quad r^* = \arg \max_{r=1, \dots, R} [f_r(x)].$$

The proposed method requires training of R classifiers. However, this architecture suffers from the so-called imbalanced learning problem, which appears when the training set for one of the class has much less samples, than all the rest in sum.

- One-versus-one strategy: For each unordered pair of classes $(L_r, L_{r'})$, a two-class SVM classifier is trained, which separates objects from class L_r from objects from $L_{r'}$, disregarding all the rest classes. Let us denote such pairwise classifiers by

$$y_{rr'}(x), \quad y_{rr'} : \mathbb{R}^n \rightarrow \{L_r, L_{r'}\}.$$

In order to classify a new incoming object x , the following procedure is performed.

First, we need to calculate how many times each class occurs as a winner in pair classification:

$$n_r(x) = |\{L_{r'} | y_{rr'}(x) = L_r\}|.$$

Then, a class with maximal “pairwise wins” is selected:

$$y(x) = L_{r^*}, \quad r^* = \arg \max_{r=1, \dots, R} [n_r(x)].$$

Such approach requires $\frac{R(R-1)}{2}$ classifiers to be trained, which is computationally complex for large R . This architecture can be exceptionally useful to tackle an imbalanced learning problem, where the samples count from all the classes but one have a clear majority over this one class. To cope with it, weighted voting procedure can be applied with some pre-defined preference matrix with confidence weights for different pairs of classifiers (Hüllermeier and Brinker, 2008). Comparison between different strategies is given by Lughofe and Buchtala (2013).

- Directed acyclic graph classification (Platt *et al.*, 2000). The approach is similar to the one-versus-one strategy, because a classifier for each pair of classes should be fitted. However, the final decision of the input label is produced not by applying all the $\frac{R(R-1)}{2}$ classifiers, but just $R - 1$. For classification, a decision tree is produced, so that each class corresponds to one leaf. Each node of the graph is linked to some of $\frac{R(R-1)}{2}$ classifiers, each outgoing edge is linked to corresponding classifier’s result. One of the nodes is selected as a root. Starting from the root, the classification process is descending to one of the leaves. An example of DAG for four classes is [Figure 13.20](#).

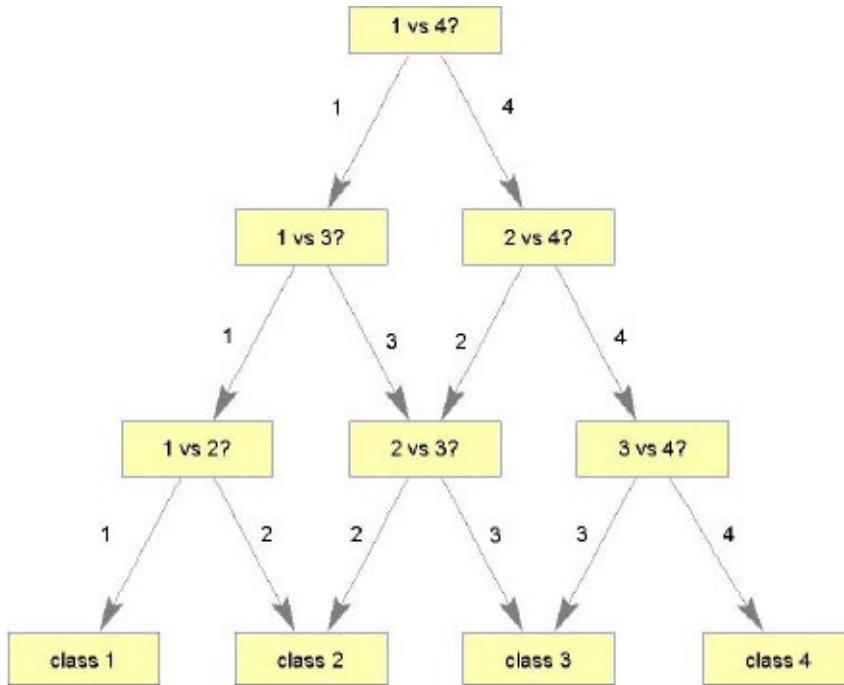


Figure 13.20: Directed acyclic graph (DAG) classification.

The main advantage of the approach is its learning simplicity against one-versus-one strategy. However, usually the classification results are not so stable as only a subpart of classifiers is engaged into the decision-making.

13.4.6. One-Class SVM

A special case of classification problems is one-class classification. This classification problem aims to distinguish between “normal” data and outliers. Such problems emerge with the training set consisting of one class examples only and an assumption of other classes existing. Either one class may be well structured and described while other being unstructured and unclear. This problem can be seen as a kind of probability distribution estimation or anomaly detection as well.

Schölkopf *et al.* (2001) showed that SVM-based approach is applicable to this class of problems. The idea of one-class SVM is to map the data into a feature space corresponding to a given kernel and to separate it from the origin with maximum margin. The quadratic program, implementing this approach, is given as follows:

$$\begin{aligned} \frac{1}{2} \|w\|^2 + \frac{1}{vN} \sum_{i=1}^N \xi_i - \rho &\rightarrow \min_{w, \xi, \rho}, \\ \text{w.r.t.} \\ \langle w, \Phi(x_i) \rangle &\geq \rho - \xi_i, \\ \xi_i &\geq 0, \quad i = 1, \dots, N. \end{aligned}$$

The decision function

$$f(x) = \text{sign}(\langle w, \Phi(x) \rangle - \rho),$$

describes the desired class (i.e., probability distribution or membership function, normal state opposite to anomalies), i.e., is positive for most examples in the training set.

To transfer to the dual form of the problem, we express the weights via KKT conditions:

$$\begin{aligned} w &= \sum_{i=1}^N \alpha_i \Phi(x_i), \\ 0 \leq \alpha_i \leq \frac{1}{vN}, \quad \sum_{i=1}^N \alpha_i &= 1. \end{aligned}$$

Then the decision function can be rewritten as

$$f(x) = \text{sign}\left(\sum_{i=1}^N \alpha_i K(x, x_i) - \rho\right).$$

Those objects with $\alpha_i > 0$ are Support Vectors. It can be proven that for any x_i with $0 < \alpha_i < \frac{1}{vN}$ it holds $\langle w, \Phi(x_i) \rangle = \rho$ at the optimum and

$$\rho = \langle w, \Phi(x_i) \rangle = \sum_{j=1}^N \alpha_j K(x_i, x_j).$$

Finally, the dual problem is given as

$$\frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j K(x_i, x_j) \rightarrow \min_{\alpha}$$

w.r.t.

$$0 \leq \alpha_i \leq \frac{1}{vN}, \quad \sum_{i=1}^N \alpha_i = 1.$$

Parameter v is of a special interest as it possesses the following theoretically proven v -property. If the solution of one-class SVM problem satisfies $\rho \neq 0$, then

- (1) v is an upper bound on the fraction of outliers in the learning set (vectors x_i for which $\alpha_i = \frac{1}{vN}$);
- (2) v is a lower bound on the fraction of SVs in the learning set (vectors x_i for which $\alpha_i > 0$);
- (3) if, additionally, data is generated independently from distribution which does not contain discrete components and the kernel is analytic and non-constant then with probability 1, asymptotically, v equals both the fraction of SVs and the fraction of outliers.

[Figure 13.21](#) shows isolines provided by one-class SVM learned on a set of two-dimensional points come from Gaussian mixture

$$p(x) = 0.5 \times p(x|\mu_1, \Sigma_1) + 0.5 \times p(x|\mu_2, \Sigma_2),$$

where $\mu_1 = (3, 3)^T$, Σ_1 is an identity matrix, $\mu_2 = (-3, -3)^T$,

$$\Sigma_2 = \begin{pmatrix} 1 & -0.5 \\ -0.5 & 2 \end{pmatrix}.$$

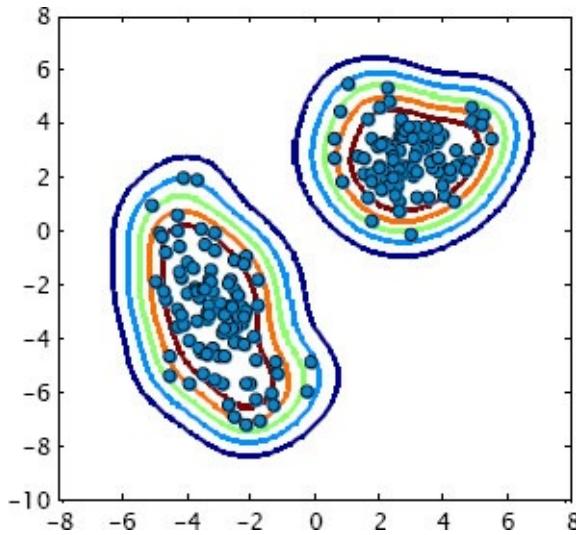


Figure 13.21: Density estimation by one-class SVM.

13.4.7. Incremental SVM Learning

Most of the SVM-type models admit incremental learning (Cauwenberghs and Poggio, 2001). It means that the model can be fitted sample-per-sample or in batch mode, but the model requires all the previous learning set to be stored. However, such algorithms could

be useful in the cases when the learning data is not provided at one moment, but is split into batches coming sequentially, for instance in streaming data analysis.

In this part, we provide the algorithm for incremental SVM learning, i.e., adding one more sample to the learning set for SVM classifier. Suppose that we have a two-class C -SVM classifier defined by its parameters (\mathbf{w}, ρ) , estimated for the kernel $K(\mathbf{x}, \mathbf{t})$ over learning sample \mathbf{X}_N .

Assume that we need to add the new sample $(\mathbf{x}_{N+1}, y_{N+1})$ to the dataset. Therefore, we need to assign a corresponding value α_{N+1} , and to update all previous values α_i , $i = 1, \dots, N$, and ρ . These parameters should be changed so that KKT conditions are met, which is necessary and sufficient condition for the optimality.

Before we discuss the algorithm itself, we should refer once again to the KKT conditions for the two-class SVM, defining three groups of samples from \mathbf{X}_N and the constraints to be met.

- (1) Margin group (M -group, $\mathbf{M} \subset \mathbf{X}_N$) — subset of vectors \mathbf{x}_m at the border of the dividing band, the corresponding KKT condition for \mathbf{x}_m is

$$M(\mathbf{x}_m) = \sum_{i=1}^N \alpha_i y_m y_i K(\mathbf{x}_i, \mathbf{x}_m) + \rho y_m = 1 \Leftrightarrow \alpha_m \in (0, C).$$

- (2) Correct group (C -group, $\mathbf{C} \subset \mathbf{X}_N$) — subset of vectors \mathbf{x}_c correctly classified by the model,

$$M(\mathbf{x}_c) = \sum_{i=1}^N \alpha_i y_c y_i K(\mathbf{x}_i, \mathbf{x}_c) + \rho y_c > 1 \Leftrightarrow \alpha_c = 0.$$

- (3) Error-group (E -group, $\mathbf{E} \subset \mathbf{X}_N$) — subset of vectors lying within the dividing band or even misclassified,

$$\mathbf{x}_e: M(\mathbf{x}_e) = \sum_{i=1}^N \alpha_i y_e y_i K(\mathbf{x}_i, \mathbf{x}_e) + \rho y_e < 1 \Leftrightarrow \alpha_e = C.$$

Additional KKT condition that should be considered is

$$\sum_{i=1}^N \alpha_i y_i = 0.$$

Here, we should mention that KKT conditions are necessary and sufficient for a point to be a solution of the dual optimization problem.

Initially, we assign $\alpha_{N+1} = 0$, leaving all the rest values $\alpha \in \mathbb{R}^N$, $\rho \in \mathbb{R}$ without changes. If margin $M(\mathbf{x}_{N+1}) > 1$ (see C -group), then all KKT conditions are met and, therefore, learning is finished.

If margin $M(\mathbf{x}_{N+1}) \leq 1$, then KKT conditions are violated, we increase α_{N+1} so that

i. margins of all points from M -group stay equal to 1,

$$\forall x_m \in M \quad M(x_m) = 1;$$

ii. $\sum_{i=1}^N \alpha_i y_i + \alpha_{N+1} y_{N+1} = 0$;

iii. margins of all points from C -group stay above 1,

$$\forall x_c \in C \quad M(x_c) > 1;$$

iv. margins of all points from E -group stay below 1,

$$\forall x_e \in E \quad M(x_e) < 1.$$

Therefore, during the change of all the parameters (α, ρ) only α_{N+1} violates the KKT conditions.

We denote the i th vector from C -, E -, and M -group x_i^c, x_i^e, x_i^m with labels y_i^c, y_i^e, y_i^m respectively.

Consider first two limitations. Assume that at the previous iteration we have the model $\alpha_{\text{old}}, \rho_{\text{old}}$. Suppose that $\alpha_{\text{old}}^m \in \mathbb{R}^{|M|}$ is a sub-vector of α_{old} related to the vectors from M -group. Once again, as the new learning data x_{N+1} arrives, we have to update all the previous parameters of the classifier, and calculate α_{N+1} iteratively. First, we assign $\alpha_{N+1} = 0$, that can violate KKT conditions, as mentioned before. Therefore, we have to change the α values. First, we fix the values of α for C -group to 0, and for E -group to C . Hence, we change only α^m values. If we increase α_{N+1} by $\Delta\alpha_{N+1} > 0$, we can calculate corresponding $\Delta\alpha^m, \Delta\rho$ while holding i and ii.

$$F^m \begin{bmatrix} \alpha_{\text{old}}^m + \Delta\alpha^m \\ \rho + \Delta\rho \end{bmatrix} + k_{N+1}^m (\alpha_{N+1} + \Delta\alpha_{N+1}) = 1,$$

where

$$F^m = \begin{bmatrix} (y^m)^T & 0 \\ Q^{mm} & y^m \end{bmatrix} \in \mathbb{R}^{|M|+1 \times |M|+1},$$

$$y^m = [y_1^m, \dots, y_i^m, \dots, y_{|M|}^m]^T,$$

$$k_{N+1}^m \in \mathbb{R}^{|M|+1}, \quad k_{N+1}^m = \begin{bmatrix} y_{N+1} \\ y_{N+1} y_1^m \langle x_1^m, x_{N+1} \rangle \\ \vdots \\ y_{N+1} y_i^m \langle x_i^m, x_{N+1} \rangle \\ \vdots \\ y_{N+1} y_{|M|}^m \langle x_{|M|}^m, x_{N+1} \rangle \end{bmatrix},$$

$$Q_{ij}^{mm} = y_i^m y_j^m \langle x_i^m, x_j^m \rangle, \quad i, j = 1, \dots, |M|.$$

Q^{mm} is a matrix of inner products of the objects in the M -group.

Therefore, taking into account that conditions i–ii are met for $\alpha_{\text{old}}, \rho_{\text{old}}$ and α_{N+1} , we can derive that

$$\begin{bmatrix} \Delta\alpha^m \\ \Delta\rho \end{bmatrix} = -(F^m)^{-1} k_{N+1}^m \Delta\alpha_{N+1}.$$

However, taking into account the conditions i–iv, we can claim that $\Delta\alpha_{N+1}$ is bounded and we increase it until one of the following conditions is met:

- (1) For one of the data vectors in the C -group, the value of $M(x_i^c)$ reaches 1. Then this data vector is moved to the M -group with corresponding α_i^c set to 0, α_{N+1} is increased by corresponding $\Delta\alpha_{N+1}$, procedure continues with the new structure of the data subsets.
- (2) For one of the data vectors in the E -group, the value of $M(x_i^e)$ reaches 1. Then this data vector is moved to the M -group with corresponding α_i^e set to 0, α_{N+1} is increased by corresponding $\Delta\alpha_{N+1}$, procedure continues with the new structure of the data subsets.
- (3) For one of the data vectors in the M -group, the value of α_i^m reaches value $\alpha_i^m = C$. Then this data vector is moved to the E -group with corresponding α_i^m set to C , α_{N+1} is increased by corresponding $\Delta\alpha_{N+1}$, procedure continues with the new structure of the data subsets.
- (4) For one of the data vectors in the M -group, the value of α_i^m gets value 0. Then this data vector is moved to the C -group with corresponding α_i^m set to 0, α_{N+1} is increased by corresponding $\Delta\alpha_{N+1}$, procedure continues with the new structure of the data subsets.
- (5) $\alpha_{N+1} = C$. Then the new data vector is moved to the E -group, and the algorithm terminates.
- (6) $M(x_{N+1}) = 1$. Then the new data vector is moved to the M -group with the current value of α_{N+1} , and the algorithm terminates.

If conditions (1)–(4) are met, then the structure of the groups E , C , and M should be re-arranged. It means that vector α^m should be increased by $\Delta\alpha^m$ and augmented/reduced by the corresponding α of the migrating vector. Matrix F^m should be recalculated. α_{N+1} is increased by $\Delta\alpha_{N+1}$. Then a new iteration of the described procedure is performed taking into account new structures of the sets C , E , M . Iterations are repeated until KKT conditions are met, which is equivalent to conditions (5) and (6).

For each condition from (1)–(4), the effective computational technique exists, which allows the inverse inner product matrix $(F^m)^{-1}$ for the M -group to be calculated iteratively using Woodbury formula. However, usually M -group is relatively small.

13.4.8. Metric Learning

At a first glance, the problem of Metric Learning stands far from the area of SVM applications. But it appears to be that SVM approach can be fruitful for learning a metric.

The problem of Metric Learning is to find a metric that pulls objects of the same class closer and pushes objects from different class as far as possible from one another. It is

common to find this metric in a family of Mahalanobis metrics parameterized by positive semidefinite matrix $M \in \mathbb{R}^{n \times n}$,

$$\rho_M(x, t) = (x - t)^T M (x - t), \quad M = M^T, \quad M \geq 0.$$

Though Mahalanobis metric was initially used to measure distance between two random vectors, it can be seen as a Euclidean distance in a linearly transformed space, i.e., for any Mahalanobis metric ρ_M there exists a linear map specified by matrix $L \in \mathbb{R}^{k \times n}$ so that

$$\rho_M(x, t) = (x - t)^T M (x - t) = (x - t)^T L^T L (x - t) = \|L(x - t)\|^2.$$

Starting from this point, most Metric Learning methods try to find the best (with some respect) linear map L .

It may be desired that in the new space for each object its distance to all neighbors of the same class is smaller than its distance to any neighbors of other classes. This leads to the following constraints on Mahalanobis metric ρ_M :

$$\min_{\substack{x_j \in \mathcal{N}(x_i) \\ y_j \neq y_i}} \rho_M(x_i, x_j) \geq \max_{\substack{x_j \in \mathcal{N}(x_i) \\ y_j = y_i}} \rho_M(x_i, x_j) + 1, \quad \forall (x_i, y_i) \in X,$$

where $\mathcal{N}(x_i)$ is a neighborhood of x_i determined by Euclidean distance.

Finding Mahalanobis metric with the smallest possible Frobenius norm results in the MLSVM model (Nguyen and Guo, 2008).

$$\frac{\lambda}{2} \|M\|_F^2 + \frac{1}{N} \sum_{i=1}^N \xi_i \rightarrow \min_{M \geq 0, \xi \geq 0},$$

w.r.t.

$$\min_{\substack{x_j \in \mathcal{N}(x_i) \\ y_j \neq y_i}} \rho_M(x_i, x_j) \geq \max_{\substack{x_j \in \mathcal{N}(x_i) \\ y_j = y_i}} \rho_M(x_i, x_j) + 1 - \xi_i, \\ \xi_i \geq 0.$$

Note that ρ_M depends linearly on the matrix M . This leads to the quadratic programming problem linking MLSVM to SVM approach. Slack variables ξ_i are introduced as it is in SVM to obtain the soft margin constraints. This optimization problem can be solved by a modified Pegasos method (Nguyen and Guo, 2008), without transforming it into a dual form.

Another approach for Metric Learning is to find such a linear transformation L that in the transformed target space neighborhood $\mathcal{N}(x_i)$ consists of objects of the same class as x_i only. Objects x_j in this neighborhood are called target neighbors and denoted as

$$j \sim i.$$

Notice that this motivation leads to better performance of KNN algorithm in the transformed space. In the original space, some objects of different classes may be closer (in Euclidean distance sense) to x_i than target neighbors. These objects are impostors x_k and defined as

$$\|L(\mathbf{x}_i - \mathbf{x}_k)\|^2 \leq \|L(\mathbf{x}_i - \mathbf{x}_j)\|^2 + 1, \quad j \sim i, \quad y_i \neq y_k.$$

Loss function in this LMNN model (Weinberger and Saul, 2009) differs from one in MLSVM and is represented as a sum of two terms, one pulling target neighbors closer together and the other pushing impostors apart. Denoting

$$y_{ij} = \begin{cases} 1, & y_i = y_j, \\ 0, & y_i \neq y_j, \end{cases}$$

we obtain the following optimization problem

$$\begin{aligned} & (1 - \mu) \sum_{j \sim i} \|L(\mathbf{x}_i - \mathbf{x}_j)\|^2 \\ & + \mu \sum_{i, j \sim i} \sum_k (1 - y_{ik}) [1 + \|L(\mathbf{x}_i - \mathbf{x}_j)\|^2 - \|L(\mathbf{x}_i - \mathbf{x}_k)\|^2]_+ \\ & \rightarrow \min_L. \end{aligned}$$

This optimization problem can be converted into a SVM-like form.

$$(1 - \mu) \sum_{j \sim i} \|L(\mathbf{x}_i - \mathbf{x}_j)\|^2 + \mu \sum_{i, j \sim i} \sum_k (1 - y_{ik}) \xi_{ijk} \rightarrow \min_{L, \xi},$$

w.r.t.

$$\begin{aligned} & \|L(\mathbf{x}_i - \mathbf{x}_k)\|^2 - \|L(\mathbf{x}_i - \mathbf{x}_j)\|^2 \geq 1 - \xi_{ijk}, \quad \forall i, j \sim i, k: y_i \neq y_k, \\ & \xi_{ijk} \geq 0, \quad M = L^T L \geq 0. \end{aligned}$$

Notice that all functions in both models depend on the input objects only through inner products. This allows incorporating kernel trick and nonlinearity into Metric Learning.

13.4.9. Structured SVM

Here we expand classification problem, relaxing the restrictions for the labels set in the way it is described by Aizerman *et al.* (1964). As usual, we consider a sample set $X \subseteq \mathbb{R}^n$ and associate each of its elements with some labels from finite set Y . In the algorithms we referred before, it is supposed that label $y \in Y \subseteq \mathbb{R}$, but here we abandon this constraint and allow labels to be any arbitrary objects (e.g., strings, graphs, etc.), often referred as “structured”. As before, we define a map $f: X \rightarrow Y$, a training set

$$X_N = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\},$$

and we are trying to learn a map $\hat{f}: X \rightarrow Y$, giving an approximation of f .

In order to tackle with such complex structured labels, we also define a discriminant function

$$F(\mathbf{x}, y|\mathbf{w}): X \times Y \rightarrow \mathbb{R},$$

where \mathbf{w} is a vector of parameters. This function gives the maximum for the most “suitable” pairs of \mathbf{x} and y , so it should represent the map \hat{f} in the form

$$\hat{f}(\mathbf{x}|\mathbf{w}) = \arg \max_{y \in Y} F(\mathbf{x}, y|\mathbf{w}).$$

In order to utilize common optimization schemes, we restrict the function $F(\mathbf{x}, y|\mathbf{w})$ to depend linearly on the parameters:

$$F(\mathbf{x}, y|\mathbf{w}) = \langle \mathbf{w}, \Psi(\mathbf{x}, y) \rangle.$$

Here Ψ is a function, dependent of the data samples, referred as the “joint feature map”, $\Psi : X \times Y \rightarrow \mathbb{R}^K$, $\mathbf{w} \in \mathbb{R}^K$. If, for example, $Y = \{1, \dots, R\} \subset \mathbb{N}$, and

$$\Psi(\mathbf{x}, y) = \{[y=1]\mathbf{x}, [y=2]\mathbf{x}, \dots, [y=R]\mathbf{x}\} \in \mathbb{R}^{R \times n},$$

then the method performs as a multiclass SVM.

The maximization problem can be reformulated (Tsochantaridis *et al.*, 2005) as a maximum margin problem

$$\frac{1}{2} \|\mathbf{w}\|^2 \rightarrow \min_{\mathbf{w}},$$

with respect to constraints defining that maximum is reached only on the given labeling \hat{f} . These constraints can be written for all $\mathbf{x}_i, y_i = f(\mathbf{x}_i)$, as

$$\max_{y \in Y \setminus y_i} \{\langle \mathbf{w}, \Psi(\mathbf{x}_i, y) \rangle\} < \langle \mathbf{w}, \Psi(\mathbf{x}_i, y_i) \rangle.$$

These constraints ensure us to have an exact labeling. However, they are nonlinear, but they can be replaced by $N(|Y| - 1)$ linear constraints

$$\begin{aligned} \forall (\mathbf{x}_i, y_i) \in X_N, \quad \forall y \in Y \setminus y_i \\ \langle \mathbf{w}, \Delta(\mathbf{x}_i, y) \rangle > 0, \end{aligned}$$

where $\Delta(\mathbf{x}_i, y) = (\Psi(\mathbf{x}_i, y_i) - \Psi(\mathbf{x}_i, y))$.

As in ordinary SVM, \mathbf{w} can be rescaled so that it fulfills the following conditions:

$$\forall \mathbf{x}_i, y_i, y \in Y \setminus y_i, \quad \langle \mathbf{w}, \Delta(\mathbf{x}_i, y) \rangle \geq 1.$$

To allow classifier errors on the dataset, slack variables ξ_i are introduced:

$$\begin{aligned} \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{N} \sum_{i=1}^N \xi_i \rightarrow \min_{\mathbf{w}, \xi} \\ \text{w.r.t.} \\ \forall \mathbf{x}_i, y_i, y \in Y \setminus y_i, \quad \langle \mathbf{w}, \Delta(\mathbf{x}_i, y) \rangle \geq 1 - \xi_i, \\ \xi_i \geq 0, \quad 1 \leq i \leq N. \end{aligned}$$

The dual version of the problem can be derived by introducing Lagrangian multipliers:

$$\begin{aligned} L(\mathbf{w}, \xi, \alpha, \beta) &= \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{N} \sum_{i=1}^N \xi_i \\ &\quad - \sum_{i=1}^N \beta_i \xi_i - \sum_{i=1}^N \sum_{y \in Y \setminus y_i} \alpha_{iy} \{ \langle \mathbf{w}, \Delta(\mathbf{x}_i, y) \rangle - 1 + \xi_i \}, \\ \frac{\partial L}{\partial \mathbf{w}} &= \mathbf{w} - \sum_{i=1}^N \sum_{y \in Y \setminus y_i} \alpha_{iy} \Delta(\mathbf{x}_i, y) = 0, \\ \frac{\partial L}{\partial \xi_i} &= \frac{C}{N} - \beta_i - \sum_{y \in Y \setminus y_i} \alpha_{iy} = 0. \end{aligned}$$

The kernel trick can be applied and the dual optimization problem should be written as

$$\sum_{i=1, y \in Y \setminus y_i}^N \alpha_{iy} - \frac{1}{2} \sum_{i=1, y \in Y \setminus y_i}^N \sum_{j=1, y' \in Y \setminus y_j}^N \alpha_{iy} \alpha_{jy'} K(\Delta(x_i, y), \Delta(x_j, y')) \rightarrow \max_{\alpha},$$

$$\alpha_{iy} \geq 0, \quad \sum_{i, y \in Y \setminus y_i} \alpha_{iy} \leq \frac{C}{N} \quad \forall i = 1, \dots, N.$$

There are several alternative formulations of the problem, different in the way of defining slack variables, but they are not ideologically different and can be found in reference (Aizerman *et al.*, 1964).

A very important problem with the described method is the large amount of constraints, leading to exponential complexity of the optimization algorithm depending on “dimensionality” of the space Y . To avoid this, cutting planes relaxation technique is utilized, which provides polynomial complexity, but takes into account only some of the constraints, giving the most severe restrictions. This means the constraints could be violated by a value not more than some pre-specified value ε . To do this, the optimization is provided within a loop, while the constraint list is complemented until all the constraints are violated not more than on ε .

The structured SVM approach can be used to solve various tasks once a correct feature-specific joint feature map Ψ is chosen, taking contributions both from input and output. Given some problem-specific function Ψ , the structured SVM can be utilized for various image segmentation (Lucchi *et al.*, 2012) and object detection problems (Tsochantaridis *et al.*, 2004), protein alignments structure identification (Joachims *et al.*, 2009) and many others.

For the case of natural grammar analysis, we have a grammar, incorporating grammar rules G_i . String parsing can be seen as a process of grammar rule application. The hierarchy of rule applications is organized into a tree. The function Ψ then represents the histogram of grammar rules application event frequency (Blaschko and Lampert, 2008). Then, the structure $y \in Y$, found by optimization of the function $F(x, y|w)$, represents a parse tree prediction.

Let us consider in more detail a problem of image segmentation as structured SVM application. For each of the pixels of the image I , we should assign an integer value $Y = \{1 \dots K\}$, referred as segment labels. Let us define a graph (V, E) , with a set of vertices V , each corresponding to one pixel of the image I , and a set of edges E , giving a neighborhood relation. The assignment of the segment labels can be done within the Conditional Random Field (CRF) framework (Blaschko and Lampert, 2008). Alternatively, it can be interpreted as graph labelling problem for the graph (V, E) with the following energy functional:

$$E_w(Y) = \sum_{v_i \in V} U_i(v_i, y_i) + \sum_{(v_i, v_j) \in E} D_{ij}(v_i, v_j, y_i, y_j) \rightarrow \min_Y.$$

Such type of optimization problems are usually solved by algorithms like graph-cuts or α -expansion (Boykov and Kolmogorov, 2004; Boykov *et al.*, 2001).

Here, U is referred as unary potential, and D is pairwise potential. However, there remains a problem to define these potentials. Then, imagine that we have a learning set of the images of the same size with corresponding “ground truth” segmentations, namely

$$\{(I_1, Y_1), (I_2, Y_2), (I_3, Y_3), \dots, (I_N, Y_N)\}, \quad I_i \in \mathbb{R}^{n \times m}, \quad Y_i \in \{1 \dots K\}^{n \times m}.$$

Structured SVM enables us to define it as an optimization problem aimed to find a functional, providing the images to be segmented closely to the “ground truth”.

For each of the pixels p_{ik} of the image I_k , we associate some feature vector $x_{ik} \in \mathbb{R}^l$, where l is the feature vector dimensionality. Here, x_{ik} can be either SIFT descriptor or simply a color value (RGB), or any other feature.

First, let us define the unary potential for the image I_k as

$$U_i^k(v_i, y_i) = \langle w_{y_i}, x_{ik} \rangle, \quad w_{y_i} \in \mathbb{R}^l.$$

Here, we should mention, that the graph is the same for all the images, as they have the same size. Let us define

$$\psi_i^k(y_i) = (I(y_i = 1)x_{ik}^T, \dots, I(y_i = K)x_{ik}^T)^T.$$

If we denote $w^U = (w_1, w_2, \dots, w_K)^T$, then

$$U_i^k(v_i, y_i) = \langle w^U, \psi_i^k(y_i) \rangle = \langle w_{y_i}, x_{ik} \rangle.$$

Second, define the pairwise potential as $D_{ij}^k(v_i, v_j, y_i, y_j) = w_{y_i, y_j} \in \mathbb{R}$.

Let us define

$$\begin{aligned} \psi_{ij}(y_i, y_j) \\ = (I(y_i = 1, y_j = 1), \dots, I(y_i = p, y_j = r), \dots, I(y_i = K, y_j = K)). \end{aligned}$$

Also, suppose

$$w^V = (w_{ij})_{(i,j) \in \{1 \dots K\}^2}.$$

Then,

$$D_{ij}^k(v_i, v_j, y_i, y_j) = \langle w^V, \psi_{ij}(y_i, y_j) \rangle = w_{y_i, y_j}.$$

We can represent optimized energy function for the image I_k in the following way:

$$\begin{aligned} E_w^k(Y) &= \sum_{v_i \in V} U_i^k(v_i, y_i) + \sum_{(v_i, v_j) \in E} D_{ij}^k(v_i, v_j, y_i, y_j) \\ &= \sum_{v_i \in V} \langle w^U, \psi_i^k(y_i) \rangle + \sum_{(v_i, v_j) \in E} \langle w^V, \psi_{ij}(y_i, y_j) \rangle. \end{aligned}$$

Therefore, $E_w^k(Y)$ is a linear function of $w \equiv (w^U, w^V)$. If we define

$$\Psi(I_k, Y) = (\Psi^D(I_k, Y), \Psi^V(Y)) = \left(\sum_{v_i \in V} \psi_i^k(y_i), \sum_{(v_i, v_j) \in E} \psi_{ij}(y_i, y_j) \right),$$

then

$$E_w^k(Y) = \langle w, \Psi(I_k, Y) \rangle.$$

We can state structured SVM problem for the weights w , with $\Psi(I_k, Y)$ as a joint feature map. One should pay attention that for each of the constraints the evaluation of the maximum operator is equivalent to the optimization problem for the energy function $E_w^k(Y)$ which can be a difficult one.

13.5. Conclusion

Kernel methods provide powerful, theoretically proven tools for a broad class of machine learning problems.

The problems we address in this chapter are namely classification, clustering, and regression. While first two problems are aimed to find a discriminant function to divide the dataset to some groups, the last is designated for estimating the function for variables relationship. The question is how accurately these functions can be approximated using information contained in the training set.

Here we use an assumption that such functions can be taken from a Reproducing Kernel Hilbert Space with a corresponding kernel, and evaluated on the given training set. Some of the kernels, called universal kernels, can accurately approximate any function. Mercer's theorem shows kernels as inner products in RKHS. All these developments pave the way to the technique of "kernel trick", which allows replacing inner products in the linear model with kernel function values. The representer theorem presents a parametrically linear model of generally nonlinear function, which is an optimal solution for a very wide range of machine learning problems.

Nowadays, the kernel techniques in machine learning are pervasive. In fact, any linear model based on inner products can be kernelized as it happens with kernel filtering or kernel linear discriminant analysis which is not "linear" at all. We have highlighted usage of kernels for Nadaraya–Watson regression, for density estimation problems. To reveal nonlinear dependencies in data, kernels can be introduced into Principal Component Analysis. Kernelizing of k -means leads to spectral clustering linking kernel methods with graph theory.

Kernels provide an extension for a widely used and deservedly praised Support Vector Machine approach. For the linear kernel, SVM searches the best discriminant hyperplane that divides two classes with the largest possible margin between them. Replacing inner products in SVM model with appropriate kernel values results in implicit mapping of input vectors to RKHS where they can be (with a successful kernel choice) linearly separated. In the input space this results in a nonlinear dividing surface.

SVM itself has many interesting extensions. Its model allows control over misclassification rate which is implicit in C-SVM and explicit in ν -SVM where the parameter ν gives an upper bound for the margin errors fraction. SVM can be extended to multiclass problems with one-versus-one or one-versus-all approaches. It can be modified to solve regression tasks resulting in ϵ -SVR and ν -SVR, or to estimate densities resulting in one-class SVM. Following tendency towards online learning methods, SVM models can be incrementally updated. SVM idea of maximizing margin between classes is applicable in adjacent areas such as metric learning. Finally, the idea of supervised learning can be expanded to composite, structured objects, such as graphs, inaddition to

regularly used real valued labels, that are supported by structured SVM models.

References

- Aizerman, M., Braverman, E. and Rozonoer, L. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automat. Rem. Contr.*, 25, pp. 821–837.
- Aronszajn, N. (1950). Theory of Reproducing Kernels. *Trans. Amer. Math. Soc.*, 68(3), pp. 337–404.
- Atkeson, C. G., Moore, A. W. and Schaal, S. (1997). Locally weighted learning. *Artif. Intell. Rev.*, 11, pp. 11–73.
- Balakrishnan, S., Xu, M., Krishnamurthy, A. and Singh, A. (2011). Noise thresholds for spectral clustering. *NIPS*, pp. 954–962.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag.
- Blaschko, M. B. and Lampert, C. H. (2008). Learning to localize objects with structured output regression. *Computer Vision: ECCV (2008)*, Berlin, Heidelberg: Springer-Verlag, pp. 2–15.
- Boser, B. E., Guyon, I. M. and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proc. 5th. Ann. Workshop on Comput. Learning Theory*, New York:ACM Press, pp. 144–152.
- Bottou, L. (1998). Online algorithms and stochastic approximations. *Online Learning and Neural Networks, Edited by David Saad*. Cambridge, UK: Cambridge University Press.
- Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization, First Edition*. New York, NY, USA: Cambridge University Press.
- Boykov Y. and Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9), 1124–1137.
- Boykov, Y., Veksler, O. and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11), pp. 1222–1239.
- Cauwenberghs G. and Poggio, T. (2001). Incremental and decremental support vector machine learning. In *Adv. Neural Inform. Processing Syst. (NIPS'2000)*, 13.
- Chung, F. (1997). Spectral graph theory. CMBS, Washington.
- Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Mach. Learning*, 20, pp. 273–297.
- Cuturi, M. (2010). Positive definite kernels in machine learning.
- Dhillon, I. S., Guan, Y. and Kulis, B. (2004). Kernel k-means: Spectral clustering and normalized cuts. In *Proc. Tenth. ACM SIGKDD. Int. Conf. Knowl. Discov. Data Min. (KDD)*, pp. 551–556.
- Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A. J. and Vapnik, V. N. (1997). Support Vector Regression Machines. In *Adv. Neural Inform. Processing Syst. 9(NIPS 1996)*, pp. 155–161.
- Exterkate, P., Groenen, P. J. F., Heij, C. and van Dijk, D. (2013). *Nonlinear Forecasting With Many Predictors Using Kernel Ridge Regression*. CREATES research papers 2013-16. University of Aarhus, Denmark: School of Economics and Management.
- Hastie, T., Tibshirani, R. and Friedman, J. (2003). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction in the Elements of Statistical Learning*, Second Edition. New York: Springer-Verlag.
- Hayes, M. H. (1996). *Statistical Digital Signal Processing and Modeling, First Edition*. New York, NY, USA: John Wiley & Sons, Inc.
- Hüllermeier, E. and Brinker, K. (2008). Learning valued preference structures for solving classification problems. *Fuzzy Sets Syst.*, pp. 2337–2352.
- Joachims, T., Hofmann, T., Yue, Y. and Yu, C.-N. (2009). Predicting structured objects with support vector machines. *Communications of the ACM, Research Highlight*, 52(11), pp. 97–104.
- Jones, M. C., Davies, S. J. and Park, B. U. (1994). Versions of kernel-type regression estimators, *J. Am. Stat. Assoc.*, 89(427), pp. 825–832.
- Jones, M. C., Marron, J. S. and Sheather, S. J. (1996). A brief survey of bandwidth selection for density estimation. *J. Am. Stat. Assoc.*, 91(433), pp. 401–407.

- Liu, W., Príncipe, J. C. and Haykin, S. (2010). Appendix B: Approximate linear dependency and system stability. In *Kernel Adaptive Filtering: A Comprehensive Introduction*. Hoboken, NJ, USA: John Wiley & Sons, Inc.
- Lucchi, A., Li, Y., Smith, K. and Fua, P. (2012). Structured image segmentation using kernelized features. *Computer Vision–ECCV*, pp. 400–413.
- Lughofer, E. and Buchtala, O. (2013). Reliable all-pairs evolving fuzzy classifiers. *IEEE Trans. Fuzzy Syst.*, 21(4), pp. 625–641.
- Luxburg, U. (2007). A tutorial on spectral clustering. *Stat. Comput.*, 17(4), pp. 395–416.
- Mercer, J. (1909). Functions of positive and negative type, and their connection with the theory of integral equations. *Phil. Trans. R. Soc. Lond.*, 209, pp. 415–446.
- Micchelli, C. A., Xu, Y. and Zhang, H. (2006). Universal Kernels. *J. Mach. Learn. Res.*, 6, pp. 2651–2667.
- Mika, S., Ratsch, G., Weston, J., Schölkopf, B. and Müller, K. (1999). Fisher discriminant analysis with Kernels. In Hu, Y.-H., Larsen, J., Wilson, E. and Douglas, S. (eds.), *Neural Networks for Signal Processing*. Piscataway, NJ: IEEE Press, pp. 41–48.
- Minh, H. Q., Niyogi, P. and Yao, Y. (2006). *Mercers Theorem, Feature Maps, and Smoothing*. New York: Springer, pp. 154–168.
- Mohar, B. (1991). The Laplacian spectrum of graphs. In *COMBGRAPH*, 2, pp. 871–898.
- Mohar, B. (1997). Some applications of Laplace eigenvalues of graphs. In Hahn, G. and Sabidussi, G. (eds.), *Graph Symmetry: Algebraic Methods and Applications, NATO ASI Series*. Netherlands: Springer, pp. 225–275.
- Müller, H.-G. (1987). Weighted local regression and Kernel methods for nonparametric curve fitting. *J. Am. Stat. Assoc.*, 82(397), pp. 231–238.
- Nadaraya, E. A. (1964). On estimating regression. *Theor. Probab. Appl.*, 9(1), pp. 141–142.
- Ng, A. Y., Jordan, M. I. and Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. *Adv. Neural Inform. Proc. Syst.*, 14, pp. 849–856.
- Nguyen, N. and Guo, Y. (2008). Metric learning: A support vector approach. In *Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases*. Berlin, Heidelberg: Springer-Verlag, pp. 125–136.
- Park, B. U. and Marron, J. S. (1990). Comparison of data-driven bandwidth selectors. *J. Am. Statist. Assoc.*, 85(409), pp. 66–72.
- Park, B. U. and Turlach, B. A. (1992). Practical performance of several data driven bandwidth selectors (with discussion). *Comput. Statistics*, 7, pp. 251–270.
- Parzen, E. (1962). On estimation of a probability density function and mode. *Ann. Math. Statist.*, 33(3), pp. 1065–1076.
- Platt, J. C., Cristianini, N. and Shawe-Taylor, J. (2000). Large Margin DAGs for Multiclass Classification. In *Proc. of Neural Inform. Processing Syst.*, (NIPS'99), pp. 547–553.
- Rosenblatt, M. (1956). Remarks on some nonparametric estimates of a density function. *Ann. Math. Statist.*, 27(3), pp. 832–837.
- Schölkopf, B., Burges, C. J. C. and Smola, A. J. (1999). *Advances in Kernel Methods: Support Vector Learning*. Cambridge, MA, USA: MIT Press.
- Schölkopf, B., Herbrich, R. and Smola, A. J. (2001). A generalized representer theorem. In *Computational Learning Theory: Lecture Notes in Computer Science*, 2111, pp. 416–426.
- Schölkopf, B., Smola, A., Williamson, R. C. and Bartlett, P. L. (2000). New support vector algorithms. *Neural Comput.*, 12, pp. 1207–1245.
- Schölkopf, B., Smola, A. J. and Müller, K.-R. (1999). Kernel principal component analysis, advances in Kernel methods — *Support Vector Learning*. Schölkopf, B., Burges, C. J. C. and Smola, A. J. (eds.). Cambridge, MA: MIT Press, pp. 327–352.
- Schölkopf, B., Platt, J., Shawe-Taylor, J., Smola, A. and Williamson, R. (2001). Estimating the support of a high-dimensional distribution. *Neural Comput.*, 13(7), pp. 1443–1472.
- Sheather, S. J. and Jones, M. C. (1991). A reliable data-based bandwidth selection method for kernel density estimation.

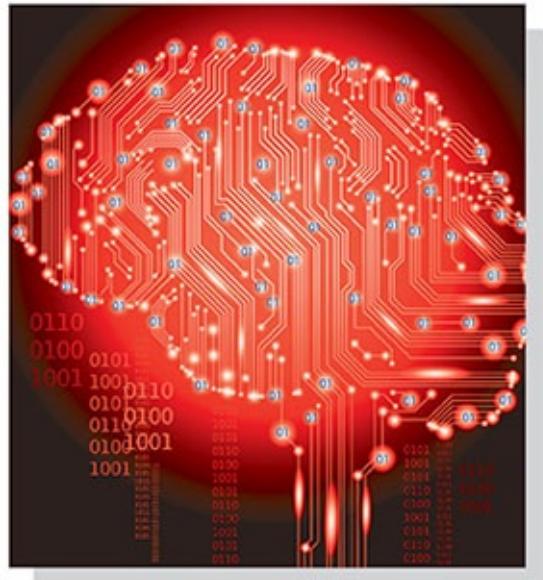
J. Roy. Stat. Soc., 53(Series B), pp. 683–690.

- Sheather, S. J. (1992). The performance of six popular bandwidth selection methods on some real datasets (with discussion). *Comput. Statistics*, 7, pp. 225–281.
- Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8), pp. 888–905.
- Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. London: Chapman & Hall.
- Tsochantaridis, I., Hofmann, T., Joachims, T. and Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *Int. Conf. Mach. Learning* (ICML), ACM, Banff, Alberta, Canada, p. 104.
- Tsochantaridis, I., Joachims, T., Hofmann, T. and Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *J. Mach. Learning Res.*, pp. 1453–1484.
- Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer-Verlag.
- Vapnik, V. and Chervonenkis, A. (1964). A note on one class of perceptrons. *Automat. Rem. Contr.*, 25, pp. 112–120.
- Watson, G. S. (1964). Smooth regression analysis. *Sankhya*, 26(15), pp. 175–184.
- Weinberger, K. Q. and Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *J. Mach. Learning Res.*, 10, pp. 207–244.
- Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G., Ng, A., Liu, B., Yu, P., Zhou, Z.-H., Steinbach, M., Hand, D. and Steinberg, D. (2008). Top 10 algorithms in data mining. *Knowl. Inform. Syst.*, 14(1), pp. 1–37.

HANDBOOK ON COMPUTATIONAL INTELLIGENCE

Volume 2

Plamen Parvanov Angelov Editor



 **World Scientific**

HANDBOOK ON COMPUTATIONAL INTELLIGENCE

**Volume 2: Evolutionary Computation,
Hybrid Systems, and Applications**

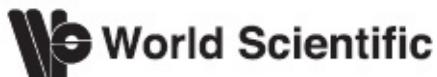
HANDBOOK ON COMPUTATIONAL INTELLIGENCE

**Volume 2: Evolutionary Computation,
Hybrid Systems, and Applications**

Editor

Plamen Parvanov Angelov

Lancaster University, UK



NEW JERSEY • LONDON • SINGAPORE • BEIJING • SHANGHAI • HONG KONG • TAIPEI • CHENNAI • TOKYO

Published by

World Scientific Publishing Co. Pte. Ltd.

5 Toh Tuck Link, Singapore 596224

USA office: 27 Warren Street, Suite 401-402, Hackensack, NJ 07601

UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library.

HANDBOOK ON COMPUTATIONAL INTELLIGENCE

In 2 Volumes

Volume 1: Fuzzy Logic, Systems, Artificial Neural Networks, and Learning Systems

Volume 2: Evolutionary Computation, Hybrid Systems, and Applications

Copyright © 2016 by World Scientific Publishing Co. Pte. Ltd.

All rights reserved. This book, or parts thereof, may not be reproduced in any form or by any means, electronic or mechanical, including photocopying, recording or any information storage and retrieval system now known or to be invented, without written permission from the publisher

For photocopying of material in this volume, please pay a copying fee through the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, USA. In this case permission to photocopy is not required from the publisher.

ISBN 978-981-4675-00-0 (Set)

ISBN 978-981-4675-03-1 (Vol. 1)

ISBN 978-981-4675-04-8 (Vol. 2)

In-house Editors: Dipasri Sardar/Amanda Yun

Typeset by Stallion Press

Email: enquiries@stallionpress.com

Printed in Singapore

Dedication

Computational Intelligence is only possible if we have a *Computer*. This work is dedicated to the inventor of the digital computer, John Vincent Atanasoff.

Contents

Introduction by the Editor

About the Editor

Acknowledgments

Prologue

Volume 1: Fuzzy Logic, Systems, Artificial Neural Networks, and Learning Systems

Part I: Fuzzy Logic and Systems

1. Fundamentals of Fuzzy Set Theory

Fernando Gomide

2. Granular Computing

Andrzej Bargiela and Witold Pedrycz

3. Evolving Fuzzy Systems — Fundamentals, Reliability, Interpretability, Usability, Applications

Edwin Lughofer

4. Modeling Fuzzy Rule-Based Systems

Rashmi Dutta Baruah and Diganta Baruah

5. Fuzzy Classifiers

Abdelhamid Bouchachia

6. Fuzzy Model-Based Control — Predictive and Adaptive Approaches

Igor Škrjanc and Sašo Blažič

7. Fuzzy Fault Detection and Diagnosis

Bruno Sielly Jales Costa

Part II: Artificial Neural Networks and Learning Systems

8. The ANN and Learning Systems in Brains and Machines

Leonid Perlovsky

9. Introduction to Cognitive Systems

Péter Érdi and Mihály Bányai

10. A New View on Economics with Recurrent Neural Networks

Hans-Georg Zimmermann, Ralph Grothmann and Christoph Tietz

11. Evolving Connectionist Systems for Adaptive Learning and Pattern Recognition: From

Neuro-Fuzzy-, to Spiking- and Neurogenetic

Nikola Kasabov

12. Reinforcement Learning with Applications in Automation Decision and Feedback Control

Kyriakos G. Vamvoudakis, Frank L. Lewis and Draguna Vrabie

13. Kernel Models and Support Vector Machines

Denis Kolev, Mikhail Suvorov and Dmitry Kangin

Volume 2: Evolutionary Computation, Hybrid Systems, and Applications

Part III: Evolutionary Computation

14. History and Philosophy of Evolutionary Computation

Carlos A. Coello Coello, Carlos Segura and Gara Miranda

15. A Survey of Recent Works in Artificial Immune Systems

Guilherme Costa Silva and Dipankar Dasgupta

16. Swarm Intelligence: An Introduction, History and Applications

Fevrier Valdez

17. Memetic Algorithms

Qiangfu Zhao and Yong Liu

Part IV: Hybrid Systems

18. Multi-Objective Evolutionary Design of Fuzzy Rule-Based Systems

Michela Antonelli, Pietro Ducange and Francesco Marcelloni

19. Bio-Inspired Optimization of Interval Type-2 Fuzzy Controllers

Oscar Castillo

20. Nature-Inspired Optimization of Fuzzy Controllers and Fuzzy Models

Radu-Emil Precup and Radu-Codruț David

21. Genetic Optimization of Modular Neural Networks for Pattern Recognition with a Granular Approach

Patricia Melin

22. Hybrid Evolutionary-, Constructive- and Evolving Fuzzy Neural Networks

Michael J. Watts and Nikola Kasabov

Part V: Applications

23. Applications of Computational Intelligence to Decision-Making: Modeling Human Reasoning/Agreement

Simon Miller, Christian Wagner and Jonathan Garibaldi

24. Applications of Computational Intelligence to Process Industry

Jose Macias Hernández

25. Applications of Computational Intelligence to Robotics and Autonomous Systems

Adham Atyabi and Samia Nefti-Meziani

26. Selected Automotive Applications of Computational Intelligence

Mahmoud Abou-Nasr, Fazal Syed and Dimitar Filev

Index

Introduction by the Editor

The term *Computational Intelligence* was coined more recently (at the end of the last century when a series of high profile conferences were organized by the Institute of Electrical and Electronics Engineers (IEEE) leading to the formation of the Computational Intelligence Society within the IEEE), however, the disciplines and problems it covers have been in existence for a much longer period of time. The very idea of developing systems, devices, algorithms, techniques that possess characteristics of “*intelligence*” and are computational (not just conceptual) dates back to the middle of the 20th century or even earlier and is broadly associated with the so-called “*artificial intelligence*”. However, “*artificial intelligence*” is nowadays rather linked with logic, cognition, natural language processing, induction and so on, while “*computational intelligence*” has been developed in a direction that can be described as “*nature-inspired*” alternatives to the conventional/traditional computing approaches. This includes, but is not limited to:

- Fuzzy logic (as a more human-oriented approach to reasoning);
- Artificial neural networks (mimicking the human brain);
- Evolutionary algorithms (mimicking the population-based genetic evolution), and
- Dynamically evolving systems based on the above.

Some authors also attribute other areas of research such as belief-based Dempster–Shafer theory, chaos theory, swarm and collective intelligence, etc. on the margins of *Computational Intelligence*. It is also often the case that the application areas such as pattern recognition, image processing, business and video analytics and so on are also attributed or linked closely to *Computational Intelligence*; areas of research that are closer to Statistical Learning (e.g., Support Vector Machines), probability theory, Bayesian, Markov models etc. are also sometimes considered to be a part of *Computational Intelligence*.

In this handbook, while not closing the door to all possible methods and alternatives, we keep clear the picture of *Computational Intelligence* as a distinct area of research that is based on the above mentioned pillars and we assume that the other areas are either applications of *Computational Intelligence* methods, techniques or approaches or research areas that gravitate around Statistical Learning.

The primary goal of the area of *Computational Intelligence* is to provide efficient computational solutions to the existing open problems from theoretical and application points of view in understanding, representation, modeling, visualization, reasoning, decision, prediction, classification, analysis and control of physical objects, environmental or social processes and phenomena to which the traditional methods, techniques and

theories (primarily, so-called “first principles”, deterministic, often expressed as differential equations and stemming from mass-and energy-balance) cannot provide a valid or useful/practical solution.

Another specific feature of *Computational Intelligence* is that it offers solutions that bear characteristics of “intelligence” which is usually attributed to humans only. This has to be considered broadly rather than literally as in the area of “artificial intelligence”. For example, fuzzy logic systems can make decisions like humans do. This is in a stark contrast with the deterministic type expert systems as well as with the probabilistic associative rules. For artificial neural networks, one can argue that they process the data in a manner which is more like what human Brain does. In evolutionary computation, the population of candidate solutions “evolves” towards the optimum in a manner similar to the way species and living organisms evolve in Nature. Finally, in dynamically evolving systems, self-development is very much like in the real life, where we, humans, learn individually from experience in a supervised (from parents, teachers, peers etc.) or unsupervised (self-learning) manner. Learning from experience, we can develop a rule-base starting from “scratch” and constantly update this rule-base by adding new rules or removing the outdated ones; or similarly, the strengths of the links between neurons in our brain can dynamically evolve and new areas of the brain can be activated or deactivated (thus, dynamically evolving neural networks).

In this handbook, a mix of leading academics and practitioners in this area as well as some of the younger generation researchers contributed to cover the main aspects of the exciting area of *Computational Intelligence*. The overall structure and invitations to a much broader circle of contributors were set up initially. After a careful peer review process, a selected set of contributions was organized in a coherent end product aiming to cover the main areas of *Computational Intelligence*; it also aims to cover both, the theoretical basis and, at the same time, to give a flavor of the possible efficient applications.

This handbook is composed of two volumes and five parts which contain 26 chapters.

[Volume 1](#) includes [Part I](#) (Fuzzy Logic) and [Part II](#) (Artificial Neural Networks and Learning Systems).

[Volume 2](#) includes [Part III](#) (Evolutionary Computation), [Part IV](#) (Hybrid Systems) and [Part V](#) (Applications).

In [Part I](#), the readers can find seven chapters, including:

- Fundamentals of Fuzzy Set Theory

This chapter sets the tone with a thorough, step by step introduction to the theory of fuzzy sets. It is written by one of the foremost experts in this area, Dr. Fernando Gomide, Professor at University of Campinas, Campinas, Brazil.

- Granular Computing

Granular computing became a cornerstone of *Computational Intelligence* and the chapter offers a thorough review of the problems and solutions. It is written by two of the leading experts in this area, Dr. Andrzej Bargiela, Professor at Nottingham University, UK (now in Christchurch University, New Zealand) and Dr. Witold Pedrycz, Professor at University of Alberta, Canada.

- Evolving Fuzzy Systems — Fundamentals, Reliability, Interpretability, Usability, Applications

Since its introduction around the turn of the centuries by the Editor, the area of evolving fuzzy systems is constantly developing and this chapter offers a review of the problems and some of the solutions. It is written by Dr. Edwin Lughofer, Key Researcher at Johannes Kepler University, Linz, Austria, who quickly became one of the leading experts in this area following an exchange of research visits with Lancaster University, UK.

- Modeling of Fuzzy Rule-based Systems

This chapter covers the important problem of designing fuzzy systems from data. It is written by Dr. Rashmi Dutta Baruah of Indian Institute of Technology, Guwahati, India and Diganta Baruah of Sikkim Manipal Institute of Technology, Sikkim, India. Rashmi recently obtained a PhD degree from Lancaster University, UK in the area of evolving fuzzy systems.

- Fuzzy Classifiers

This chapter covers the very important problem of fuzzy rule-based classifiers and is written by the expert in the field, Dr. Hamid Bouchachia, Associate Professor at Bournemouth University, UK.

- Fuzzy Model based Control: Predictive and Adaptive Approach

This chapter covers the problems of fuzzy control and is written by the experts in the area, Dr. Igor Škrjanc, Professor and Dr. Sašo Blažič, Professor at the University of Ljubljana, Slovenia.

- Fuzzy Fault Detection and Diagnosis

This chapter is written by Dr. Bruno Costa, Professor at IFRN, Natal, Brazil who specialized recently in Lancaster, UK.

Part II consists of six chapters, which cover:

- The ANN and Learning Systems in Brains and Machines

This chapter is written by Dr. Leonid Perlovsky from Harvard University, USA.

- Introduction to Cognitive Systems

This chapter is written by Dr. Peter Erdi, Professor at Kalamazoo College, USA, co-authored by Dr. Mihaly Banyai (leading author) from Wigner RCP, Hungarian Academy of Sciences.

- A New View on Economics with Recurrent Neural Networks

This chapter offers a rather specific view on the recurrent neural networks from the point of view of their importance for modeling economic processes and is written by a team of industry-based researchers from Siemens, Germany including Drs. Hans Georg Zimmermann, Ralph Grothmann and Christoph Tietz.

- Evolving Connectionist Systems for Adaptive Learning and Pattern Recognition: From Neuro-Fuzzy, to Spiking and Neurogenetic

This chapter offers a review of one of the cornerstones of *Computational Intelligence*, namely, the evolving connectionist systems, and is written by the pioneer in this area Dr. Nikola Kasabov, Professor at Auckland University of Technology, New Zealand.

- Reinforcement Learning with Applications in Automation Decision and Feedback Control

This chapter offers a thorough and advanced analysis of the reinforcement learning from the perspective of decision-making and control. It is written by one of the world's leading experts in this area, Dr. Frank L. Lewis, Professor at The University of Texas, co-authored by Dr. Kyriakos Vamvoudakis from the same University (the leading author) and Dr. Draguna Vrabie from the United Technologies Research Centre, USA.

- Kernel Models and Support Vector Machines

This chapter offers a very skilful review of one of the hottest topics in research and applications linked to classification and related problems. It is written by a team of young Russian researchers who are finishing their PhD studies at Lancaster University, UK (Denis Kolev and Dmitry Kangin) and by Mikhail Suvorov. All three graduated from leading Moscow Universities (Moscow State University and Bauman Moscow State Technical University).

Part III consists of four chapters:

- History and Philosophy of the Evolutionary Computation

This chapter lays the basis for one of the pillars of *Computational Intelligence*, covering its history and basic principles. It is written by one of the well-known experts in this area, Dr. Carlos A. Coello-Coello from CINVESTAV, Mexico and co-authored by Dr. Carlos Segura from the Centre of Research in Mathematics, Mexico and Dr. Gara Miranda from the University of La Laguna, Tenerife, Spain.

- A Survey of Recent Works in Artificial Immune Systems

This chapter covers one of the important aspects of *Computational Intelligence* which is

associated with the Evolutionary Computation. It is written by the pioneer in the area Dr. Dipankar Dasgupta, Professor at The University of Memphis, USA and is co-authored by Dr. Guilherme Costa Silva from the same University who is the leading author.

- **Swarm Intelligence: An Introduction, History and Applications**

This chapter covers another important aspect of Evolutionary Computation and is written by Dr. Fevrier Valdez from The Institute of Technology, Tijuana, Mexico.

- **Memetic Algorithms**

This chapter reviews another important type of methods and algorithms which are associated with the Evolutionary Computation and is written by a team of authors from the University of Aizu, Japan led by Dr. Qiangfu Zhao who is a well-known expert in the area of *Computational Intelligence*. The team also includes Drs. Yong Liu and Yan Pei.

Part IV consists of five chapters:

- **Multi-objective Evolutionary Design of Fuzzy Rule-Based Systems**

This chapter covers one of the areas of hybridization where Evolutionary Computation is used as an optimization tool for automatic design of fuzzy rule-based systems from data. It is written by the well-known expert in this area, Dr. Francesco Marcelloni, Professor at the University of Pisa, Italy, supported by Dr. Michaela Antonelli and Dr. Pietro Ducange from the same University.

- **Bio-inspired Optimization of Type-2 Fuzzy Controllers**

The chapter offers a hybrid system where a fuzzy controller of the so-called type-2 is being optimized using a bio-inspired approach. It is written by one of the leading experts in type-2 fuzzy systems, Dr. Oscar Castillo, Professor at The Institute of Technology, Tijuana Mexico.

- **Nature-inspired Optimization of Fuzzy Controllers and Fuzzy Models This**

chapter also offers a hybrid system in which fuzzy models and controllers are being optimized using nature-inspired optimization methods. It is written by the well-known expert in the area of fuzzy control, Dr. Radu-Emil Precup, Professor at The Polytechnic University of Timisoara, Romania and co-authored by Dr. Radu Codrut David.

- **Genetic Optimization of Modular Neural Networks for Pattern Recognition with a Granular Approach**

This chapter describes a hybrid system whereby modular neural networks using a granular approach are optimized by a genetic algorithm and applied for pattern recognition. It is written by Dr. Patricia Melin, Professor at The Institute of Technology, Tijuana, Mexico who is well-known through her work in the area of hybrid systems.

- Hybrid Evolutionary-, Constructive-, and Evolving Fuzzy Neural Networks

This is another chapter by the pioneer of evolving neural networks, Professor Dr. Nikola Kasabov, co-authored by Dr. Michael Watts (leading author), both from Auckland, New Zealand.

Part V includes four chapters:

- Applications of Computational Intelligence to Decision-Making: Modeling Human Reasoning/Agreement

This chapter covers the use of *Computational Intelligence* in decision-making applications, in particular, modeling human reasoning and agreement. It is authored by the leading expert in this field, Dr. Jonathan Garibaldi, Professor at The Nottingham University, UK and co-authored by Drs. Simon Miller (leading author) and Christian Wagner from the same University.

- Applications of Computational Intelligence to Process Industry

This chapter offers the industry-based researcher's point of view. Dr. Jose Juan Macias Hernandez is leading a busy Department of Process Control at the largest oil refinery on the Canary Islands, Spain and is also Associate Professor at the local University of La Laguna, Tenerife.

- Applications of Computational Intelligence to Robotics and Autonomous Systems

This chapter describes applications of *Computational Intelligence* to the area of Robotics and Autonomous Systems and is written by Dr. Adham Atyabi and Professor Dr. Samia Nefti-Meziani, both from Salford University, UK.

- Selected Automotive Applications of Computational Intelligence

Last, but not least, the chapter by the pioneer of fuzzy systems area Dr. Dimitar Filev, co-authored by his colleagues, Dr. Mahmoud Abou-Nasr (leading author) and Dr. Fazal Sayed (all based at Ford Motors Co., Dearborn, MI, USA) offers the industry-based leaders' point of view.

In conclusion, this Handbook is composed with care aiming to cover all main aspects of *Computational Intelligence* area of research offering solid background knowledge as well as end-point applications from the leaders in the area supported by younger researchers in the field. It is designed to be a one-stop-shop for interested readers, but by no means aims to completely replace all other sources in this dynamically evolving area of research.

Enjoy reading it.

Plamen Angelov

Lancaster, UK

About the Editor



Professor Plamen Angelov (www.lancs.ac.uk/staff/angelov) holds a Personal Chair in Intelligent Systems and leads the Data Science Group at Lancaster University, UK. He has PhD (1993) and Doctor of Sciences (DSc, 2015) degrees and is a Fellow of both the IEEE and IET, as well as a Senior Member of the International Neural Networks Society (INNS). He is also a member of the Boards of Governors of both bodies for the period 2014–2017. He also chairs the Technical Committee (TC) on Evolving Intelligent Systems within the Systems, Man and Cybernetics Society, IEEE and is a member of the TCs on Neural Networks and on Fuzzy Systems within the Computational Intelligence Society, IEEE. He has authored or co-authored over 200 peer-reviewed publications in leading journals, peer-reviewed conference proceedings, five patents and a dozen books, including two research monographs by Springer (2002) and Wiley (2012), respectively. He has an active research portfolio in the area of data science, computational intelligence and autonomous machine learning and internationally recognized results into online and evolving learning and algorithms for knowledge extraction in the form of human-intelligible fuzzy rule-based systems. Prof. Angelov leads numerous projects funded by UK research councils, EU, industry, UK Ministry of Defence, The Royal Society, etc. His research was recognized by ‘The Engineer Innovation and Technology 2008 Special Award’ and ‘For outstanding Services’ (2013) by IEEE and INNS. In 2014, he was awarded a Chair of Excellence at Carlos III University, Spain sponsored by Santander Bank. Prof. Angelov is the founding Editor-in-Chief of Springer’s journal on *Evolving Systems* and Associate Editor of the leading international scientific journals in this area, including *IEEE Transactions on Cybernetics*, *IEEE Transactions on Fuzzy Systems* and half a dozen others. He was General, Program or Technical co-Chair of prime IEEE conferences (IJCNN-2013, Dallas; SSCI2014, Orlando, WCCI2014, Beijing; IS’14, Warsaw; IJCNN2015, Killarney; IJCNN/ WCCI2016, Vancouver; UKCI 2016, Lancaster; WCCI2018, Rio de Janeiro) and founding General co-Chair of a series of annual IEEE conferences on Evolving and Adaptive Intelligent Systems. He was a Visiting Professor in Brazil, Germany, Spain, France, Bulgaria. Prof. Angelov regularly gives invited and plenary talks at leading conferences, universities and companies.

Acknowledgments

The Editor would like to acknowledge the support of the Chair of Excellence programme of Carlos III University, Madrid, Spain.

The Editor would also like to acknowledge the unwavering support of his family (Rosi, Lachezar and Mariela).

Prologue

The *Handbook on Computational Intelligence* aims to be a one-stop-shop for the various aspects of the broad research area of Computational Intelligence. The *Handbook* is organized into five parts over two volumes:

- (1) Fuzzy Sets and Systems (Vol. 1)
- (2) Artificial Neural Networks and Learning Systems (Vol. 1)
- (3) Evolutionary Computation (Vol. 2)
- (4) Hybrid Systems (Vol. 2)
- (5) Applications (Vol. 2)

In total, 26 chapters detail various aspects of the theory, methodology and applications of *Computational Intelligence*. The authors of the different chapters are leading researchers in their respective fields or “rising stars” (promising early career researchers). This mix of experience and energy provides an invaluable source of information that is easy to read, rich in detail, and wide in spectrum. In total, over 50 authors from 16 different countries including USA, UK, Japan, Germany, Canada, Italy, Spain, Austria, Bulgaria, Brazil, Russia, India, New Zealand, Hungary, Slovenia, Mexico, and Romania contributed to this collaborative effort. The scope of the *Handbook* covers practically all important aspects of the topic of Computational Intelligence and has several chapters dedicated to particular applications written by leading industry-based or industry-linked researchers.

Preparing, compiling and editing this Handbook was an enjoyable and inspirational experience. I hope you will also enjoy reading it and will find answers to your questions and will use this book in your everyday work.

Plamen Angelov
Lancaster, UK

Volume 2

Part III

Evolutionary Computation

Chapter 14

History and Philosophy of Evolutionary Computation

Carlos A. Coello Coello, Carlos Segura and Gara Miranda

This chapter provides a general overview of evolutionary computation (EC), including a short review of natural evolution theories and some basic concepts from natural evolution and genetics that have been adopted in EC. Then, a historical review of EC is presented, including its three main paradigms: evolution strategies, evolutionary programming and genetic algorithms. In the final part of the chapter, a unified view of EC is provided, together with a short introduction to the design of evolutionary algorithms that includes a practical example.

14.1. Introduction

Evolutionary computation (EC) is a research area within computer science that studies the properties of a set of algorithms — usually called evolutionary algorithms (EAS) — that draw their inspiration from natural evolution. Initially, several algorithms intended to better understand the population dynamics present in evolution were designed (Crosby, 1967). However, although the design of the EC schemes is based on drawing inspiration from natural evolution, a faithful modeling of biologic processes is not usually incorporated. Thus, since EAS are usually overly simplistic versions of their biological counterparts, using EAS to model population dynamics is not too widely accepted by the evolutionary biology community (Mitchell, 1998). Instead, the advances achieved in recent decades have shown that the main strength of EC is that it can be successfully applied to numerous practical problems that appear in several areas such as process control, machine learning and function optimization (Fogel, 1995). For this reason, EC should be seen as a general and robust evolution-inspired framework devoted to problem solving.

Although it is not easy to classify the kinds of problems that can be solved with EC, some taxonomies have been proposed. For instance, Everett distinguished between two main uses of EAS (Everett, 2000): Optimizing the performance of operating systems, and the testing and fitting of quantitative models. Another view was proposed by Eiben and Smith (2003) by providing an analogy between problems and systems. In working systems, the three main components are: inputs, outputs, and the internal model connecting these two. They distinguished between three kinds of problems that can be solved with EC depending on which of the three components is unknown. In any case, the important fact is that EC has been successfully used in a huge number of applications such as numerical optimization, design of expert systems, training of neural networks, data mining and many others. In general, EC might be potentially applied to any problem where we can identify a set of candidate solutions and a quality level associated with each of them.

This chapter is devoted to presenting the history and philosophy behind the use of EC. In [Section 14.2](#), a brief review of natural evolution theories is provided. In addition, some of the main concepts from the fields of evolution and genetics that have inspired developments in EC are introduced. Then, [Section 14.3](#) presents the history and philosophy of the first EC approaches ever developed. Some of the latest developments, as well as the current unifying view of EC, are summarized in [Section 14.4](#). [Section 14.5](#) offers some observations on the design of EAS and describes some of the most popular components that have been used when tackling optimization problems with EC. Finally, some concluding remarks are given in [Section 14.6](#).

14.2. The Fundamentals of Evolution

The term “*evolution*” is defined by the Oxford Dictionary as *the process by which different kinds of living organisms are thought to have developed and diversified from earlier forms during the history of the Earth*. A much more general definition of this term is *the gradual development of something, especially from a simple to a more complex form*. The word “*evolution*” originated from the Latin word “*evolutio*”, which means unrolling, from the verb “*evolvere*”. Early meanings related to physical movement, first recorded in describing a tactical “*wheeling*” maneuver in the realignment of troops or ships. Current senses stem from a notion of “*opening out*” and “*unfolding*”, giving rise to a general sense of “*development*”. The term appeared a couple of centuries before Darwin wrote “*On the Origin of Species*”. In fact, Darwin did not even use the word evolution in his book until the last line:

There is grandeur in this view of life, with its several powers, having been originally breathed by the Creator into a few forms or into one; and that, whilst this planet has gone circling on according to the fixed law of gravity, from so simple a beginning endless forms most beautiful and most wonderful have been, and are being evolved.

14.2.1. A Historical Review

During the 18th century, a group of researchers, called naturalists, managed to gather a great deal of information on the flora and fauna in many different areas of our planet. In an attempt to organize and classify this remarkable amount of information, **Carl Linnaeus** (1707–1778) proposed a set of rules to assign genus and species labels to all known living beings. His taxonomy, called *System Naturae* (Linnaeus, 1735), focused solely on the morphological properties of living beings to define the classification. Since this classification criterion was purely morphological, to Linnaeus, species identified distinct groups with no relation of origin. This perspective, called *fixity of species*, considered that each species was created as it was, and individuals did not experience changes over time. Linnaeus began his study believing this concept, but later rejected it after observing interbreeding between various species. Due to his extensive work in the field, he is considered the founding father of our current taxonomic system.

The accumulation of information provided by naturalists and the progress achieved in the taxonomies led to the adoption of new approaches, different from the fixity of species, and based on the fact that some species came from other species. This idea required defining a new classification that reflected the relationships among organisms. It was called “*natural classification*”. Although **Georges-Louis Leclerc, Comte de Buffon** (1707–1788), was the first to question Linnaeus’s fixity of species, the first to propose a hypothesis for how one species could come from another was **Jean-Baptiste Pierre Antoine de Monet, Chevalier de Lamarck** (1744–1829). Lamarck, in his *Zoological Philosophy* (de Monet de Lamarck, 1809) presented a systematic description for the evolution of living beings. For Lamarck, species develop as a result of their reaction and

adaptation to the environment. These changes, therefore, must be gradual and will occur over long periods of time. Lamarck suggested that certain organs are strengthened by the use that animals make of them, mainly due to the specific nature of their environment. Other organs, in contrast, are atrophied and eventually eliminated because they fall into disuse. For Lamarck, nature developed in the following way: circumstances create a need, that need creates habits, habits produce the changes resulting from the use or disuse of the organ and the means of nature are responsible for setting these modifications. Lamarck believed that these physiological changes acquired over the life of an organism could be transmitted to offspring. This hypothesis is known as the “*inheritance of acquired characteristics*” and is also commonly referred to as *Lamarckism*. We can say, therefore, that Lamarck was the first to formulate a strictly evolutionary hypothesis, although the word “evolution” was at that time reserved for the development of the embryo, so his proposal was referred to as “*transformism*”. Despite Lamarck’s hypothesis, there was no experimental evidence for the existence of mechanisms by which individuals could transmit the alleged improvements acquired over the course of their lives. In fact, Lamarckism is now considered an obsolete theory. The principles governing the transformation of the individual characters, which are now commonly accepted by science, were first established by Darwin and Wallace. The principles governing the transmission or inheritance of these characteristics were first established by Mendel.

In 1858, **Charles Robert Darwin** (1809–1882) and **Alfred Russel Wallace** (1823–1913) gave a presentation at the Linnean Society of London on a theory of the evolution of species by means of “*natural selection*” (Darwin and Wallace, 1858). One year later, Darwin published *On the Origin of Species* (Darwin, 1859), a work in which he provided a detailed explanation of his theory supported by numerous experiments and observations of nature. Darwin’s theory — usually known as *Darwinism* — is based on a set of related ideas which try to explain different features of biological reality:

- Living beings are not static; they change continuously: Some new organisms are created and some die out. This process of change is gradual, slow and continuous, with no abrupt or discontinuous steps.
- Species diversify, by adapting to different environments or ways of life, thus branching out. The implication of this phenomenon is that all species are somehow related — to varying degrees — and ultimately all species have a single origin in one common and remote ancestor. That is, all living organisms can be traced back to a single and common origin of life.
- Natural selection is the key to the system. It is conceived as a result of two factors: The inherited natural variability of the individuals of a species, and the selection through survival in the struggle for life, i.e., the fittest individuals, who were born with favorable spontaneous modifications better suited to the environment, and are thus more likely to survive, reproduce, and leave offspring with these advantages. This implies that each

slight variation, if useful, is preserved.

Although Darwin knew that there should be a mechanism for transmitting these characteristics from parents to offspring, he was unable to discover the transmission mechanism. It was **Gregor Johann Mendel** (1822–1884) who suggested a number of hypotheses which would set the basic underlying principles of heredity. Mendel's research (Mendel, 1865) focused on plants (peas) and, especially on individual features, all unequivocally different from one other and having the peculiarity of not being expressed in a graduated form, i.e., the feature is only present or not present. This research led Mendel to three important conclusions:

- The inheritance of each trait is determined by “units” or “factors” that are passed on to descendants unchanged. These units are now called “*genes*”.
- An individual inherits one such unit from each parent for each trait.
- A trait may not show up in an individual but can still be passed on to the next generation. In this sense, the term “*phenotype*” refers to the set of physical or observable characteristics of an organism, while the term “*genotype*” refers to the individual's complete collection of genes. The difference between genotype and phenotype is that the genotype can be distinguished by looking at the deoxyribonucleic acid (DNA) of the cells and the phenotype can be established from observing the external appearance of an organism.

Mendel's laws are the foundation of modern genetics and abolished the idea that characteristics are transmitted from parents to children through bodily fluids so that, when mixed, cannot be separated, thus causing the offspring to have characteristics that will mix the characteristics of the parents. This theory is called “*pangenesis*” and was mainly based on observations such as how crossing plants with red flowers with plants with white flowers produces plants with pink flowers. Starting from Mendel's advances in genetics and the concept of natural selection, in the thirties and forties the “*modern evolutionary synthesis*” was established (Fisher, 1930; Haldane, 1932; Wright, 1931). Basically, this theory (Huxley, 1942) served as a link between the unity of evolution (“the gene”) and the mechanism of evolution (“the selection”), i.e., gradual changes and natural selection in populations are the primary mechanism of evolution. According to this theory, genetic variation in populations arises mainly by chance through mutation (alteration or change in the genetic information — genotype — of a living being) and recombination (the mixing of chromosomes produced at meiosis).

14.2.2. Main Concepts

In the above description, some important definitions related to evolution and genetics were introduced. This section is devoted to describing some other terms that have been used in some popular books and papers on EC. A complete list of the terms contained in the papers

on EC would be too extensive to be included. For this reason, we have selected the most broadly used terms:

- *DNA*: Nucleic acid that consists of two long chains of nucleotides twisted into a double helix and joined by hydrogen bonds between the complementary bases adenine and thymine or cytosine and guanine. It is the main constituent of the chromosome and carries the genes as segments along its strands.
- *Chromosome*: Structure within the nucleus of eukaryotic cells that bears the genetic material as a thread-like linear strand of DNA.
- *Gene*: As seen in the previous section, Mendel regarded “genes” as inherited factors which determine the external features on living beings. In modern genetics, a “gene” is defined as a sequence of DNA that occupies a specific location on a chromosome.
- *Diploid cell*: Cell that contains two sets of paired chromosomes. For example, humans have 2 sets of 23 chromosomes, for a total of 46 chromosomes. Exact replicas of diploid cells are generated through a process called mitosis.
- *Haploid cell*: Cell that contains only one complete set of chromosomes. The genesis of a haploid cell can occur by meiosis of diploid cells or by mitosis of haploid cells. A haploid cell will merge with another haploid cell at fertilization, i.e., sperm and ova (also known as “gametes”).
- *Locus*: Specific position of the chromosome where a gene or other DNA sequence is located.
- *Genetic linkage*: Tendency of genes that are located proximal to each other on a chromosome to be inherited together.
- *Alleles*: Variant or alternative forms of a gene which are located at the same position — locus — on a chromosome.
- *Genotype*: Genetic information of an organism. This information — contained in the chromosomes of the organism — may or may not be manifested or observed in the individual.
- *Phenotype*: Property observed in the organism, such as morphology, development, or behavior. It is the expression of the genotype in relation to a particular environment.
- *Epistasis*: Type of interaction between genes located at different loci on the same chromosome consisting of one gene masking or suppressing the expression of the other.

14.3. History of EC

EC is a field of research with a fascinating but complex history. In its origins, several independent researchers proposed numerous approaches with the common feature of using natural evolution to inspire their implementations. Some of the developments were also inspired from other, related fields such as artificial life (Conrad and Pattee, 1970) and other areas of artificial intelligence. EC had many independent beginnings, so different terms have been used to refer to concepts that are broadly similar. Given the similarities of the different schemes proposed, the term EC was invented in the early 1990s in an effort to unify the field of evolution-inspired algorithms (Fogel, 1995).

Determining the merits of the various authors is not easy, which is why several different versions of the history of EC have been told (Fogel, 1998). In this section, we review the origins of EC, focusing on the most important milestones. Many of the bases of EC were developed in the 1960s and 1970s. During that period, three different popular schemes inspired by natural evolution were devised: Evolutionary Programming (EP), Evolution Strategies (ESS), and Genetic Algorithms (GAS). These schemes were inspired by the same ideas but some details, like the nature of the representation and the operators applied, were different. In addition, there were several earlier attempts that highly influenced the advances in this area. Some of these proposals can be considered to be the first designed EAs. In other cases, the schemes were substantially different from what is currently known as an EA. In any case, they are historically important because they had a significant impact on the subsequent studies carried out in the field. This section reviews the origins along with the three aforementioned types of EAS.

14.3.1. Origins

Wright and Cannon (Cannon, 1932; Wright, 1932) were probably the first researchers to influence the subsequent development of EC. They viewed natural evolution as a learning process where the genetic information of species is continuously changed through a trial and error mechanism. In this way, evolution can be regarded as a method whose aim is to maximize the adaptiveness of species to their environment. Wright went further and introduced the concept of fitness landscape as a representation of the space of all possible genotypes along with their fitness values. This concept is widely used today to study the performance of EAS (Richter, 2014). In addition, he developed one of the first studies where selection and mutation were linked, concluding that a proper balance between them is required to proceed successfully. These ideas were extended by Campbell (1960). He claimed that a blind-variation-and-selective-survival process is one of the main underlying principles of evolution, and hypothesized that “in all processes leading to expansions of knowledge, a blind-variation-and-selective-survival process is involved”. This is one of the principles behind Universal Darwinism, a proposal that extends the applicability of Darwin’s theory to other areas beyond natural evolution.

Turing can also be considered another EC pioneer. In Turing (1950), Turing recognized a connection between machine learning and evolution. Specifically, Turing claimed that in order to develop an intelligent computer that passes the famous Turing test, a learning scheme based on evolution might be used. In this regard, the structure of the evolved machine is related to hereditary material, changes made to the machine are related to mutation and the use of an experimenter to evaluate the generated machines are related to natural selection. In addition, Turing had previously suggested that these kinds of methods might be used to train a set of networks which were akin to current neural networks (Turing, 1948). Turing used the term “genetical search” to refer to these kinds of schemes. However, this paper was not published until 1968 (Evans and Robertson, 1968) because his supervisor considered it to be a “schoolboy essay” (Burgin and Eberbach, 2013). By the time of its publication, other EAS had already appeared and the term “genetical search” was never adopted by the EC community.

Some important advances were made in the late 1950s and early 1960s, coinciding with the period in which electronic digital computers became more readily available. In this period, several analyses of the population dynamics appearing in evolution were carried out. A survey of the application of digital computers in this field was presented in (Crosby, 1967). Crosby identified three kinds of schemes:

- Methods based on studying the mathematical formulations that emerge in the deterministic analysis of population dynamics. These schemes usually assume an infinite population size and analyze the distributions of genes under different circumstances. An example of this type of scheme is the one developed by Lewontin (1964), where the interactions between selection and linkage are analyzed.
- Approaches that simulate evolution but do not explicitly represent a population (Crosby, 1960). In these methods, the frequencies of the different potential genotypes are stored for each generation, resulting in a non-scalable scheme in terms of the number of genes. In addition, a faithful mathematical representation of the evolutionary system is required.
- Schemes that simulate evolution by explicitly maintaining a representation of the population. Fraser and some of his colleagues pioneered these kinds of methods and they analyzed this approach in a set of papers published over the course of a decade (Fraser, 1957; Fraser and Burnell, 1967). The main conclusions drawn from these studies were gathered in their seminal book (Fraser and Burnell, 1970). From its inception, this model was widely accepted, and several researchers soon adopted it (Gill, 1965; Young, 1966). In fact, even though there are several implementation details that differ from those used in current EAS, several authors regard the works of Fraser as representing the first invention of a GA (Fogel, 1995). As for the implementation details, the main difference with respect to GAS is that individuals are diploid instead of haploid. However, in our opinion, the key difference is not the implementation but the

philosophy behind the use of the scheme. Fraser used its algorithms as a way to analyze population dynamics and not to solve problems, which is the typical current application of EAS.

There were other authors who proposed schemes resembling current EAS. Among them, the works of Friedman, Box, and Friedberg are particularly important and often-cited papers. Friedman (1956) hypothesized on the automatic design of control circuits by using a scheme inspired by natural evolution that was termed “selective feedback”. His proposals were very different from current ones. For instance, there was no notion of population and/or generations. Moreover, Friedman did not implement his method, and some of his assumptions seem to be overly optimistic (Fogel, 1998). In any event, his research can be considered as the first efforts in the field of evolvable hardware.

Box proposed a technique called “Evolutionary Operation” (EVOP) (Box, 1957) to optimize the management processes of the chemical industry. His technique essentially used a single parent to generate multiple offspring by modifying a few production parameters at a time. Then, one individual was selected to survive to the next generation by considering certain statistical evidence. The system was not autonomous. Specifically, the variation and selection processes required human intervention, so it can be considered as the first interactive EA.

The most important contribution of Friedberg *et al.* to the field of EC was their attempt to automatically generate computer programs using an evolutionary scheme (Friedberg, 1958; Friedberg *et al.*, 1959). In the previous papers, Friedberg *et al.* did not identify any relationship between their proposal and natural evolution. However, in subsequent publications by his coauthors, such a relationship was explicitly identified (Dunham *et al.*, 1963). In their first attempts, the aim was to automatically generate small programs with some simple functionalities that relied on a tailor-made instruction set. In order to direct the search to promising regions, problem-dependent information was considered by defining a variation scheme specifically tailored to the problem at hand. Even with this addition, their success was limited. Even so, some of the analyses they carried out were particularly important to subsequent achievements in EC. Among their contributions, some of the most important are the following:

- The idea of dividing candidate solutions into classes is a precursor of the notions of *intrinsic parallelism* and *schema*, proposed years later by Holland (1975).
- Several instruction sets were tested, showing, for the first time, the influence of genotype-to-phenotype mapping.
- A credit assignment algorithm was used to measure the influence of the single instructions. This idea is closely related to the work of Holland on GAS and classifier systems.

There are several more papers that did not elicit much attention when they were first

published, but that proposed techniques very similar to others that were later reinvented. For instance, the work by Reed, Toombs, and Barricelli (Reed *et al.*, 1967) provides several innovations closely related to self-adaptation, crossover, and coevolution.¹

14.3.2. EP

EP was devised by Fogel (1962) while he was engaged in basic research on artificial intelligence for the National Science Foundation. At the time, most attempts to generate intelligent behavior used man as a model (Fogel, 1999). However, Fogel realized that since evolution had been able to create humans and other intelligent creatures, a model mimicking evolution might be successful. Note that, as previously described, some research into this topic had already been published by then. The basic ideas adopted by Fogel are similar: Use variation and selection to evolve candidate solutions better adapted to the given goals. However, Fogel was not aware of the existing research and his proposals differed substantially from them, meaning that his models can be considered as a reinvention of evolution-based schemes.

Philosophically, the coding structures utilized in EP are an abstraction of the phenotype of different species (Fogel, 1994). As a result, the encoding of candidate solutions can be freely adapted to better support the requirements of the problems at hand. Since there is no sexual communication between different species, this view of candidate solutions also justifies the lack of recombination operators in EP. Thus, the non-application of recombination operators is due to a conceptual rather than a technical view. The lack of recombination, the freedom to adapt the encoding and the use of a probabilistic survivor selection operator — not used in the initial versions of EP — might be considered the main features that distinguished EP from other EC paradigms (Fogel and Chellapilla, 1998).

In his first designs, Fogel reasoned that one of the main features that characterizes intelligent behavior is the capacity to predict one's environment, coupled with a translation of said predictions into an adequate response so as to accomplish a given objective. A finite state transducer is a finite state machine (FSM) that allows a sequence of input symbols to be transformed into a sequence of output symbols. Depending on the states and given transitions, they can be used to model different situations and cause different transformations. Thus, Fogel viewed FSMS as a proper mechanism for dealing with the problem of generating intelligent behavior. In his initial experiments, the aim was to develop a FSM capable of predicting the behavior of an environment. The environment was considered to be a sequence of symbols belonging to a given input alphabet. The aim was to develop a FSM that, given the symbols previously generated by the environment, could predict the next symbol to emerge from the environment.

The initial EP proposal operates as follows. First, a population with N random FSMS is created. Then, each member of the population is mutated, creating as many offspring as parents. Five mutation modes are considered: add a state, delete a state, change the next

state of a transition, change the output symbol of a transition, or change the initial state. Mutation operators are chosen randomly — other ways were also tested — and in some cases the offspring are subjected to more than one mutation. Finally, the offspring are evaluated and the best N FSMS are selected to survive. FSMS are evaluated in light of their capacity to correctly predict the next symbols in known sequences. Initially, the fitness is calculated by considering a small number of symbols, but as the evolution progresses more symbols are attached to the training set.

In the first experiments carried out involving EP, different prediction tasks were tested: Periodic sequences of numbers, sequences with noise, non-stationary environments, etc. Subsequently, more difficult tasks were considered. Among other applications, EP was used to tackle pattern recognition, classification, and control system design. All this research resulted in the publication of the first book on EC (Fogel *et al.*, 1966). EP was also used to evolve strategies for gaming (Fogel and Burgin, 1969). This work is particularly important because it is one of the first applications of coevolution.

It is also important to remark that in most of the initial studies on EP, the amount of computational results was not ample because of the limited computational power at the time of its inception. Most of these initial experiments were recapitulated and extended in a later period (Fogel and Fogel, 1986), providing a much deeper understanding of EP.

In the 1970s, most of the research into EP was conducted under the guidance of Dearholt. One of the main contributions of his work was the application of EP to practical problems. EP was applied to pattern recognition in regular expressions (Lyle, 1972) and handwritten characters (Cornett, 1972) and to classify different types of electrocardiograms (Dearholt, 1976). These works incorporated several algorithmic novelties. Among them, the most influential were the use of several simultaneous mutation operators and the dynamic adaptation of the probabilities associated with the different mutation schemes.

Starting in the 1980s, EP diversified by using other arbitrary representations of candidate solutions in order to address different problems. The number of applications that have been addressed with EP is huge. In Fogel and Fogel (1986), EP was used to solve routing problems by considering a permutation-based encoding. In order to tackle the generation of computer programs, tree-based encoding was used in (Chellapilla, 1997). Real-valued vectors were used to deal with continuous optimization problems in (Yao *et al.*, 1999) and to train neural networks in (Porto and Fogel, 1995). During this period, the feeling was that by designing problem-specific representations with operators specifically tailored to face a given problem, more efficient searches might be performed (Fogel, 1999). In fact, most EP practitioners defended that by designing intelligent mutation schemes, the use of recombination operators might be avoided (Chellapilla, 1997; Fogel and Atmar, 1990).

Among the aforementioned topics, the application of EP to continuous optimization problems saw a large expansion in the 1990s. Over this period, great efforts were made into developing self-adaptive schemes. In self-adaptation, some of the parameters required by the algorithm are bound to each individual and evolved with the original variables. The variables of the problem are called *object parameters*, while those newly attached are called *strategy parameters*. Initial proposals adapted the scale factor of Gaussian mutation (Fogel *et al.*, 1991). In more advanced variants of EP, several mutation operators are considered simultaneously (Yao *et al.*, 1999). Since self-adaptive schemes surpassed more traditional EP variants, self-adaptation was adopted by practitioners addressing problems where real encoding was not used, becoming a common mechanism in EP (Fogel *et al.*, 1995). Self-adaptive EP for continuous optimization presents several similarities with ESS. The most important difference is that EP does not include recombination. In addition, some implementation details were also different in the first variants devised (Eiben and Smith, 2003). For instance, the order in which the object and strategy parameters were subjected to mutation was different in the two schemes, though these differences vanished over time. Moreover, it has been shown that there are cases where mutation alone can outperform schemes with recombination and vice versa (Richter *et al.*, 2008). There is also evidence that indicates that it is promising to adapt the parameters associated with crossover (Jain and Fogel, 2000), which further stretches the barriers between EP and ESS. Since self-adaptation for continuous optimization was first proposed in ESS, the history of and advances in self-adaption for continuous optimization is presented in the section devoted to ESS.

14.3.3. Evolution Strategies

In the mid-1960s, three students at the Technical University of Berlin — Bienert, Schwefel and Rechenberg — were studying practical problems that arise in fluid mechanics and some related fields. Their desire was to build robots that could autonomously solve engineering problems (Fogel, 1998). They formulated these engineering tasks as optimization problems and developed autonomous machines that were automatically modified using certain rules. They also made some initial attempts at using traditional optimization schemes. However, since the problems were noisy and multimodal, their schemes failed to provide promising solutions. In order to avoid these drawbacks, they decided to apply mutation and selection methods analogous to natural evolution. Rechenberg published the first report on ESS by applying these ideas to minimizing the total drag of a body in a wind tunnel (Rechenberg, 1965). Subsequently, other problems like the design of pipes and efficient flashing nozzles were addressed (Fogel, 1998).

Philosophically, the coding structures utilized in ESS are an abstraction of the phenotype of different individuals (Fogel, 1994). This is why the encoding of candidate

solutions can be freely adapted to better support the requirements of the problems at hand. In addition, recombination among individuals is allowed in spite of not being implemented in the initial variants of ESS.

In the first problems tested, a set of quantitative discrete variables had to be adjusted. The initial proposal was very similar to the current $(1 + 1)$ -ES and works as follows. First, a random solution is created and considered to be the current solution. Then, the current solution is mutated by slightly changing each variable. Specifically, mutation is implemented by emulating a Galton pin-board, so binomial distributions are used. The principle behind the design of this operator is to apply small mutations more frequently than large mutations, as is the case in nature. The current solution is replaced by the mutant only if the mutant solution is at least as good. This process is repeated until the stopping criterion is reached. Empirical evidence revealed the promising behavior of these kinds of schemes with respect to more traditional optimization methods.

In 1965, Schwefel first implemented an ES in a general computer (Schwefel, 1965). In this case, ESS were mainly applied to continuous optimization. The most important contribution of the new variant was the incorporation of Gaussian mutation with zero mean, which is still the most typical distribution used nowadays. The mutation operator could be controlled by tuning the standard deviations — or step size — of the Gaussian distribution. Studies on the step size resulted in the first theoretical analyses of ESS.

As previously discussed, initial ES variants only kept a single solution at a time. However, in the late 1960s, the use of populations with several individuals was introduced. The first population-based scheme tested is now known as $(\mu + 1)$ -ES. This scheme uses a population with μ individuals. These individuals are used to create a new individual through recombination and mutation. Then, the best μ individuals from the $\mu+1$ individuals are selected to survive. This scheme resembles the steady state selection that was popularized much later in the field of GAS (Whitley and Kauth, 1988).

ESS were extended further to support any number of parents and offspring. In addition, two different selection schemes and a notation that is still in use were proposed. The first selection comprises the $(\mu + \lambda)$ -ES. In this case, λ offspring are created from a population with μ individuals. Then, the best μ individuals from the union of parents and offspring are selected to survive. Schemes that consider the second kind of selection are known as (μ, λ) -ES. In this case, the best μ individuals from the λ offspring are selected to survive.

When ESS were first developed, the papers were written in German and, as a consequence, they were accessible only to a fairly small community. However, in 1981, Schwefel published the first book on ESS in English (Schwefel, 1981). Since then, several researchers have adopted ESS and the number of studies in this area has grown enormously. Schwefel's book focuses on the use of ESS for continuous optimization, which is in fact the field where ESS have been more successful. In his book, Schwefel discusses, among other

topics, the use of different kinds of recombinations and the adaptation of the Gaussian distributions adopted for mutation. Regarding recombination, some controversies and misunderstandings have appeared. The use of adaptive mutation is one of the distinguishing features of ESS, and the history of and advances in both topics are described herein. Most of the papers reviewed in this chapter involve continuous single-objective optimization problems. However, it is important to note that the application of ESS has not been limited to these kinds of problems. For instance, multi-objective (Igel *et al.*, 2007), mixed-integer (Li *et al.*, 2013) and constrained problems (Mezura-Montes and Coello Coello, 2005) have also been addressed.

14.3.3.1. Recombination

Initially, four different recombination schemes were proposed by combining two different properties (Schwefel, 1981). First, two choices were given for the number of parents taking part in creating an offspring: *bisexual* or *multisexual*. In bisexual recombination — also known as local — two parents are selected and they are used to recombine each parameter value. In the case of multisexual recombination, also known as global, different pairs of parents are selected for each parameter value. Thus, more than two individuals can potentially participate in the creation of each offspring. In addition, two different ways of combining two parameter values were proposed. In the *discrete* case, one of the values is selected randomly. In the *intermediary* case, the mean value of both parameters is calculated.

In the above recombination schemes, the number of parents taking part in each recombination cannot be specified. Rechenberg proposed a generalization that changed this restriction (Rechenberg, 1978). Specifically, a new parameter, ρ , is used to specify the number of parents taking part in each individual creation. The new schemes were called $(\mu/\rho, \lambda)$ and $(\mu/\rho + \lambda)$. Further extensions of the intermediary schemes were developed by weighting the contributions of the different individuals taking part in the recombination (Bäck and Schwefel, 1993). Different ways of assigning weights have been extensively tested. Previous schemes were not proper generalizations of the initial schemes, in the sense that the intermediary case calculates the mean of ρ individuals, and not the mean of two individuals selected from among a larger set, as was the case in the initial schemes. A more powerful generalization that allowed for both the original operators and those proposed by Rechengerg to be implemented was devised (Eiben and Bäck, 1997). Thus, using the notation proposed in this last paper is preferable.

The situation became even more complicated because of a misinterpretation of the initial crossover schemes. In the survey presented in (Bäck *et al.*, 1991), the authors showed a formula indicating that, in global recombination, one parent is chosen and held fixed while the other parent is randomly chosen anew for each component.² This new way of recombination was adopted by many authors over the following years, so when using

global recombination, the exact definition must be carefully given.

14.3.3.2. Adaptive Mutation

One of the main features that characterizes ESS is its mutation operator. In the case of continuous optimization, mutation is usually based on Gaussian distributions with zero mean. In most cases, individuals are disturbed by adding a random vector generated from a multivariate Gaussian distribution, i.e., the candidate solutions x_i are perturbed using Equation (1). In this equation, C represents a covariance matrix.

$$x'_i = x_i + N_i(0, C). \quad (1)$$

From the inception of ESS, it was clear that the features of C might have a significant effect on search performance. As a result, several methods that adapt C during the run have been developed. Many ES variants can be categorized into one of the following groups depending on how C is adapted (Hansen and Ostermeier, 2001):

- Schemes where the surfaces of equal probability density are hyper-spheres. In these cases, the only free adaptive parameter is the global step size or standard deviation.
- Schemes where the surfaces of equal probability density are axis-parallel hyper-ellipsoids. In these cases, the most typical approach is to have as many free adaptive parameters as there are dimensions.
- Schemes where the surfaces of equal probability density are arbitrarily oriented hyper-ellipsoids. In these cases, any positive-definite matrix can be used, resulting in $(n^2 + n)/2$ free adaptive parameters.

The first adaptive ES considered the global step size as the only free parameter. These schemes originated from the studies presented in (Rechenberg, 1973), which analyzed the convergence properties of ESS depending on the relative frequency of successful mutations. These analyses led to the first adaptive scheme (Schwefel, 1981), where the global step size is adjusted by an online procedure with the aim of producing successful mutations with a ratio equal to 1/5. This rule is generally referred to as the “1/5 success rule of Rechenberg”. However, this rule is not general so it does not work properly for many functions, which is why self-adaptive ESS were proposed. In the first such variants, the only strategy parameter was the global step size, and this strategy parameter was subjected to variation using a lognormal distributed mutation.

In the case of schemes where the surfaces of equal probability are axis-parallel hyperellipsoids, a larger number of methods have been devised. A direct extension of previous methods considers the self-adaptation of n strategy parameters, where n is the number of dimensions of the optimization problem at hand. In such a case, each strategy parameter represents the variance in each dimension. Another popular ES variant is the

derandomized scheme (Ostermeier *et al.*, 1994a). Ostermeier *et al.* realized that in the original self-adaptive ES, the interaction of the random elements can lead to a drop in performance. The scheme is based on reducing the number of random decisions made by ESS. Another interesting extension is termed *accumulation* (Ostermeier *et al.*, 1994b). In these variants, the adaptation is performed considering information extracted from the best mutations carried out during the whole run, and not only in the last generation. Note that these last variants favor adaptation instead of self-adaptation.

By adapting the whole covariance matrix, correlations can be induced among the mutations performed involving the different parameters, thus making ESS more suitable for dealing with non-separable functions. The first adaptive scheme where the whole matrix is adapted was presented in (Schwefel, 1981). In this case, the coordinate system in which the step size control takes place, as well as the step sizes themselves, are self-adapted. In general, this method has not been very successful because it requires very large populations to operate properly. A very efficient and popular scheme is covariance matrix adaptation (cma-es) (Hansen and Ostermeier, 2001). In this scheme, derandomization and accumulation are integrated in an adaptive scheme whose aim is to build covariance matrices that maximize the creation of mutation vectors that were successful in previous generations of the execution. This scheme is very complex and the practitioner must specify a large number of parameters. This has led to simpler schemes being devised. For instance, the covariance matrix self-adaptation (CMSA) scheme (Beyer and Sendhoff, 2008) reduces the number of parameters required by combining adaptation and self-adaptation. Many other not so popular schemes that are capable of adapting the whole covariance matrix have been devised. The reader is referred to (Rudolph, 2012) for a broader review of these kinds of schemes.

Finally, it is important to note that several schemes that favor some directions without adapting a full covariance matrix have been devised. For instance, in (Poland and Zell, 2001), the main descent direction is adapted. The main advantage of these kinds of schemes is the reduced complexity in terms of time and space. Since the time required for matrix computation is usually negligible in comparison to the function evaluation phase, these schemes have not been very popular. Still, they might be useful especially for large-scale problems, where the cost of computing the matrices is much higher. In addition, several other adaptive and self-adaptive variants that do not fit into the previous categorization have been devised. For instance, some schemes that allow the use of Gaussian distributions with a non-zero mean have been proposed (Ostermeier, 1992). In addition, some methods not based on Gaussian mutation have been designed (Yao and Liu, 1997). In this chapter, we have briefly described the fundamentals of different methods. The reader is referred to (Bäck *et al.*, 2013; Rudolph, 2012) for a discussion of the implementation details for several ES variants.

14.3.4. Genetic Algorithms

Holland (1975) identified the relationship between the adaptation process that appears in natural evolution and optimization, and conjectured that it might have critical roles in several other fields, such as learning, control and/or mathematics. He proposed an algorithmic framework inspired by these ideas, which was initially called the “Genetic Plan” and was subsequently renamed the “Genetic Algorithm”. GAS are based on progressively modifying a set of structures with the aim of adapting them to a given environment. The specific way of making these modifications is inspired by genetics and evolution. His initial aim was to formally study the phenomenon of adaptation. However, GAS were soon used as problem solvers. Specifically, GAS have been successfully applied to numerous applications, such as machine learning, control, search, and function optimization. A very popular use of GAS appeared in the development of *learning classifier systems*, which is an approach to machine learning based on the autonomous development of rule-based systems capable of generalization and inductive learning. The application of GAS as function optimizers was also soon analyzed (De Jong, 1975), defining metrics, benchmark problems, and methodologies to perform comparisons. This is probably the field where GAS have been most extensively and successfully applied.

The evolutionary schemes devised by Holland are very similar to those previously designed by Bremermann (1962) and Bledsoe (1961). In fact, “by 1962 there was nothing in Bremermann’s algorithm that would distinguish it from what later became known as genetic algorithms” (Fogel and Anderson, 2000). Moreover, they pioneered several ideas that were developed much later, like the use of multisexual recombination, the use of real-encoding evolutionary approaches and the adaptation of mutation rates. Moreover, since Bremermann’s algorithm was very similar to the schemes described earlier by Fraser, some authors regard GAS as having been reinvented at least three times (Fogel, 1995).

Philosophically, the coding structures utilized in GAS are an abstraction of the genotype of different individuals (Fogel, 1994). Specifically, each trial solution is usually coded as a vector, termed *chromosome*, and each element is denoted with the term *gene*. A candidate solution is created by assigning values to the genes. The set of values that can be assigned are termed *alleles*. In the first variants of GAS, there was an emphasis on using binary representations — although using other encodings is plausible — so even when continuous optimization problems were addressed, the preferred encoding was binary. Goldberg claimed that “GAS work with a coding of the parameter set, not the parameters themselves” (Goldberg, 1989), which is a distinguishing feature of GAS. This results in the requirement to define a mapping between genotype and phenotype.

The working principles of GAS are somewhat similar to those of EP and ESS. However, unlike other schemes, in most early forms of GAS, recombination was emphasized over mutation. The basic operation of one of the first GAS — denoted as canonical GA or simple GA — is as follows. First, a set of random candidate solutions is created. Then, the performance of each individual is evaluated and a *fitness value* is assigned to each

individual. This fitness value is a measure of each individual's quality, and in the initial GA variants it had to be a positive value. Based on the fitness values, a temporary population is created by applying *selection*. This selection process was also called *reproduction* in some of the first works on GAS (Goldberg, 1989). The best-performing individuals are selected with larger probabilities and might be included several times in the temporary population. Finally, in order to create the population of the next generation, a set of variation operators is applied. The most popular operators are crossover and mutation, but other operators such as inversion, segregation, and translocation have also been proposed (Goldberg, 1989). In most of the early research on GAS, more attention was paid to the crossover operator. The reason is that crossover was believed to be the major source of the power behind GAS, while the role of mutation was to prevent the loss of diversity (Mitchell, 1998). This first variant was soon extended by Holland's students. Some of these extensions include the development of game-playing strategies, the use of diploid representations and the use of adaptive parameters (Bäck *et al.*, 1997).

Among Holland's many contributions, the schema theorem was one of the most important. Holland developed the concepts of schemata and hyperplanes to explain the working principles of GAS. A schema is basically a template that allows grouping candidate solutions. In a problem where candidate solutions are represented with l genes, a schema consists of l symbols. Each symbol can be an allele of the corresponding gene or “*”, which is the “do not care” symbol. Each schema designates the subset of candidate solutions in which the corresponding representations match every position in the schema that is different from “*”. For instance, the candidate solutions “0 0 1 1” and “0 1 0 1” belong to the schema “0 * * 1”. Note that the candidate solutions belonging to a schema form a hyperplane in the space of solutions. Holland introduced the concept of *intrinsic parallelism* — subsequently renamed as *implicit parallelism* — to describe the fact that the evaluation of a candidate solution provides valuable information on many different schemata. He hypothesized that GAS operate by better sampling schemata with larger mean fitness values.

The concept of schema was also used to justify the use of binary encoding. Specifically, Holland showed that by using binary encoding the number of schemata sampled in each evaluation can be maximized. However, the use of binary encoding was not extensively tested in his preliminary works, and its application has been a source of controversy. One of the first problems detected with the use of binary encoding is that, depending on the length of the vector, precision or efficiency might be sacrificed. It is not always easy to strike a proper balance, so dynamic parameter encoding methods have also been proposed (Schraudolph and Belew, 1992). Goldberg weakened the binary-encoding requirement by claiming that “the user should select the smallest alphabet that permits a natural expression of the problem” (Goldberg, 1989). A strict interpretation of this means that the requirement for binary alphabets can be dropped (Bäck *et al.*, 1997). In fact, several authors have been able to obtain better results by using representations different

from the binary encoding, such as floating point representations. For instance, Michalewicz carried out an extensive comparison between binary and floating point representations (Michalewicz, 1994). He concluded that GAS operating with floating-point representations are faster, more consistent and provide a higher precision. Subsequently, Fogel and Ghozeil demonstrated that there are equivalences between any bijective representation (Fogel and Ghozeil, 1997). Thus, for GAS with bijective representations, the interactions that take place between the representation and other components are one of the keys to their success.

Holland used the above concepts to develop the *schema theorem*, which provides lower bounds to the change in the sampling rate for a single hyperplane from one generation to the next. For many years, the schema theorem was used to explain the working operation of GAS. However, several researchers have pointed out some weaknesses that call into question some of the implications of the theorem (Altenberg, 1995). Some of the most well-known weaknesses pointed out by several researchers are the following. First, the schema theorem only provides lower bounds for the next generation and not for the whole run. Second, the schema theorem does not fully explain the behavior of GAS in problems that present large degrees of inconsistencies in terms of the bit values preferred (Heckendorn *et al.*, 1996). Finally, considering the typical population sizes used by most practitioners, the number of samples belonging to high-order schemata — those with few “**” — is very low, so in these cases the theorem is not very accurate. In general, the controversies are not with the formulae, but with the implications that can be derived from them. The reader is referred to (Whitley and Sutton, 2012) for a more detailed discussion of this topic. Finally, we should note that other theoretical studies not based on the concept of schema have also been proposed to explain the behavior of GAS. In some cases, the model is simplified by assuming infinitely large population sizes (Vose and Liepins, 1991). In other cases, the Markov model has been used to analyze simple GAS with finite population sizes (Nix and Vose, 1992).

Due to the complexity of analyzing the mathematics behind GAS, there is a large amount of work focused on the practical use of GAS. For instance, the initial GA framework has been extended by considering several selection and variation operators. Regarding the former, note that initially, selection was only used to choose the individuals that were subjected to variation. This operator is now known as *parent selection* or *mating selection* (Eiben and Smith, 2003). However, in most current GAS, a *survivor selection* or *environmental selection* is also carried out. The aim of this last selection — also called *replacement* — is to choose which individuals survive to the next generation. In its most general form, survivor selection can operate on the union of parents and offspring. The number of selection operators defined in recent decades is very large, ranging from deterministic to stochastic schemes. Parent selection is usually carried out with stochastic operators. Some of the most well-known stochastic selectors are: *fitness proportional selection*, *ranking selection*, and *tournament selection*. Alternatively, survivor selection is

usually based on deterministic schemes. In the case of the survivor selection schemes, not only is the fitness used, but the age of individuals might also be considered. Some of the most well-known schemes are *age-based replacement* and *replace-worst*. Also of note is the fact that *elitism* is included in most current GAS. Elitist selectors ensure that the best individual from among parents and offspring is always selected to survive. In these kinds of schemes, and under some assumptions, asymptotic convergence is ensured (Eiben *et al.*, 1991) and, in general, there is empirical evidence of its advantages. A topic closely related to replacement is the *steady state model* (Whitley and Kauth, 1988). In the steady state model, after the creation of one or maybe a few offspring, the replacement phase is executed. Note that this model is closely related to the *generation gap* concept previously defined in (De Jong, 1975). The generation gap was introduced into GAS to permit overlapping populations.

In the same way, several crossover schemes have been proposed in the literature. The first GAS proposed by Holland operated with one-point crossover, whose operation is justified in light of the schema theorem. Specifically, Goldberg (1989) claimed that in GAS with one-point crossover, short, low-order, and highly fit schemata — which received the name of building blocks³ — are sampled, recombined, and resampled to form strings of potentially higher fitness. However, this hypothesis has been very controversial. One of the basic features of one-point crossover is that bits from a single parent that are close together in the encoding are inherited together, i.e., it suffers from *positional bias*. This idea was borrowed in part from the biological concept of *coadapted alleles*. However, several authors have questioned the importance of this property (Whitley and Sutton, 2012). In fact, Goldberg claimed that since many encoding decisions are arbitrary, it is not clear that a GA operating in this way might obtain the desired improvements (Goldberg, 1989). The inversion operator might in some way alleviate this problem by allowing the reordering of genes so that linkages between arbitrary positions can be created. Another popular attempt to identify linkages was carried out in the messy GA (Goldberg *et al.*, 1989), which explicitly manipulates schemata and allows linking non-adjacent positions. More advanced linkage learning mechanisms have been depicted recently (Chen and Lim, 2008). In addition, given the lack of a theory that fully justifies the use of one-point crossover, several different crossover operators, as well as alternative hypotheses to explain the effectiveness of crossover, have been proposed. Among these hypotheses, some of the most popular view crossover as a macro-mutation or as an adaptive mutation (Sivanandam and Deepa, 2007). Regarding alternative definitions of crossover operators, it has been shown empirically that depending on the problem, operators different from one-point crossover might be preferred. For instance, Syswerda showed the proper behavior of uniform crossover with a set of benchmark problems (Syswerda, 1989). In the case of binary encoding, some of the most well-known crossover operators are the two-point crossover, multi-point crossover, segmented crossover and uniform crossover (Eiben and Smith, 2003). Note that some multisexual crossover operators have also been

provided. In fact, well before the popularization of GAS, Bremermann had proposed their use (Fogel, 1998). For other representations, a large set of crossover operators have been devised (Eiben and Smith, 2003).

As mentioned earlier, several components and/or parameters in GAS have to be tuned. The first GA variants lacked procedures to automatically adapt these parameters and components. However, several adaptive GAS have also been proposed (Lobo *et al.*, 2007). In these schemes, some of the parameters and/or components are adapted by using the feedback obtained during the search. In addition, most current GAS do not operate on binary strings, relying instead on encodings that fit naturally with the problem at hand. As a result, several variation operators specific to different chromosome representations have been defined. The reader is referred to (Eiben and Smith, 2003) for a review of these operators.

14.4. A Unified View of EC

The three main branches of EC — (EP, GAS, and ESS) — developed quite independently of each other over the course of about 25 years, during which several conferences and workshops devoted to specific types of EAS were held, such as the Evolutionary Programming Conference. However, in the early 1990s, it was clear to some researchers that these approaches had several similarities, and that findings in one kind of approach might also be useful for the other types of EAS. Since these researchers considered that the topics covered by the conferences at the time period were too narrow, they decided to organize a workshop called “Parallel Problem Solving From Nature” that would accept papers on any type of EA, as well as papers based on other metaphors of nature. This resulted in a growth in the number of interactions and collaborations among practitioners of the various EC paradigms, and as a result the different schemes began to merge naturally. This unification process continued at the *Fourth International Conference on Genetic Algorithms*, where the creators of EP, GAS, and ESS met. At this conference, the terms EC and EA were proposed and it was decided to use these terms as the common denominators of their approaches. Basically, an EA was defined as any approach towards solving problems that mimics evolutionary principles.⁴ Similar efforts were undertaken at other conferences, such as in the *Evolutionary Programming Conference*. Finally, these unifying efforts resulted in the establishment in 1993 of the journal *Evolutionary Computation*, published by MIT press. Furthermore, the original conferences were soon replaced by more general ones, like the *Genetic and Evolutionary Computation Conference*⁵ and the *IEEE Congress on Evolutionary Computation*. In addition, other journals specifically devoted to EC have appeared, like the *IEEE Transactions on Evolutionary Computation*.

It is worth noting that since the different types of EAS were developed in an independent way, many ideas have been reinvented and explored more than once. In addition, there was no clear definition for each kind of scheme, so the boundaries between EC paradigms become blurred. Thus, for many contemporary EAs it is very difficult and even unfair to claim that they belong to one or another type of EA. For instance, a scheme that adopts proportional parent selection, self-adaptation and stochastic replacement merges ideas that were originally depicted in each of the initial types of EAS. For this reason, when combining ideas that were originated by practitioners of the different EC paradigms — which is very typical — it is preferable to use the generic terms EC and EA.

Algorithm 1.1 Pseudocode of an EA — A Unified View

- 1: **Initialization:** Generate an initial population with N individuals
- 2: **Evaluation:** Evaluate every individual in the population
- 3: **while** (not fulfilling the stopping criterion) **do**

- 4: **Mating selection:** Select parents to generate the offspring
 - 5: **Variation:** Apply variation operators to the mating pool to create a child population
 - 6: **Evaluation:** Evaluate the child population
 - 7: **Survivor selection:** Select individuals for the next generation
 - 8: **end while**
-

In recent decades, several papers comparing the differences and similarities of the different types of EAS have appeared (Bäck *et al.*, 1993). In addition, some authors have proposed unifying and general frameworks that allow for the implementation of any of these schemes (Bäck, 1996; De Jong, 2006). In fact, note that with a pseudocode as simple as the one shown in Algorithm 1.1, any of the original schemes can be implemented. Although this simple pseudocode does not fit with every contemporary EA, it does capture the main essence of EC by combining population, random variation and selection. Two interesting books where the unifying view is used are: (De Jong, 2006; Eiben and Smith, 2003).

The last two decades have seen an impressive growth in the number of EC practitioners. Thus, the amount of research that has been conducted in the area is huge. For instance, EAS have been applied to several kinds of optimization problems, such as constrained (Mezura-Montes, 2009) and multi-objective (Coello Coello *et al.*, 2006) optimization problems. Also, several efforts have been made to address the problem of premature convergence (Črepinšek *et al.*, 2013). Studies conducted on this topic include the use of specific selection schemes such as fitness sharing (Goldberg and Richardson, 1987), crowding schemes (De Jong, 1975), restarting mechanisms (Eshelman, 1991) and the application of multi-objective concepts to tackle single-objective problems (Segura *et al.*, 2013b). Other highly active topics include the design of parallel EAS (Alba, 2005) and memetic (Moscato, 1989) algorithms — related to Lamarck’s hypothesis which was discussed before — which allows hybridizing EAS with more traditional optimization schemes. Note that for most of the topics that have been studied over these past decades, there were some preliminary works that were developed in the 1960s. For instance, the island-based model — which is one of the most popular parallel models — was proposed in (Bossert, 1967), while some preliminary works on hybrid models were presented as early as 1967 (Kaufman, 1967). The aforementioned works represent only a very small cross-section of the topics that have been covered in recent years. It is beyond the scope of this chapter to present an extensive review of current research, so readers are referred to some of the latest papers published in some of the most popular conferences and journals in the field.

Interestingly, it is also remarkable that while considerable efforts have been made to

unify the different EC paradigms, some new terms for referring to specific classes of EAS have also appeared in recent decades. However, in these last cases, their distinguishing features are clear, so there is no ambiguity in the use of such terms. For instance, Differential Evolution (Storn and Price, 1995) is a special type of EA where the mutation is guided by the differences appearing in the current population. Another popular variant is Genetic Programming (Koza, 1990), which focuses on the evolution of computer programs.

Finally, we would like to state that while there has been considerable research into EC in the last decade, the number of topics that have yet to be addressed and further explored in the field is huge. Moreover, in recent years there has been a remarkable increase in the number of proposals based on alternative nature-inspired phenomena. In some cases, they have been merged with evolutionary schemes. Thus, similarly to the period where synergies were obtained by combining ideas that arose within different evolutionary paradigms, the interactions among practitioners of different nature-inspired algorithms might be beneficial for advancing this field.

14.5. Design of EAs

EC is a general framework that can be applied to a large number of practical applications. Among them, the use of EC to tackle optimization problems is probably the most popular one (Eiben and Smith, 2003). The generality of EC and the large number of different components that have been devised imply that when facing new problems, several design decisions must be made. These design decisions have an important impact on the overall performance (Lobo *et al.*, 2007), meaning they must be made very carefully. This section discusses some of the main decisions involved in the design of EAs and describes some of the most well-known components that have been used to handle optimization problems.

One of the first decisions to be made when applying EAs is the way in which individuals are represented. Let us consider a simple optimization problem such as the minimization of the function $f(x, y, z)$ [Equation (2)], where each variable is an integer number in the range $[0, 15]$. Considering some of the first GAS designed, an alternative is to represent each number in base-2 with 4 binary digits.

Values of variables	Binary representation	Integer representation
$x = 5, y = 8, z = 3$	0 1 0 1 1 0 0 0 0 0 1 1	5 8 3
$x = 7, y = 0, z = 11$	0 1 1 1 0 0 0 0 1 0 1 1	7 0 11

Figure 14.1: Individuals with binary and integer encoding.

In such a case, individuals consist of 12 binary digits or genes (based on the GAS nomenclature). Another possible choice — which is more extended nowadays — is to use a gene for each variable, where each gene can have any value between 0 and 15. [Figure 14.1](#) shows some candidate solutions considering both the binary and integer representations. Each cell represents a gene of the chromosome. The representation chosen influences other decisions. For instance, the design of the variation operators depends on the representation, which is why it must be carefully selected.

$$f(x, y, z) = (x - 1)^2 + (y - 2)^2 + (z - 7)^2 - 2. \quad (2)$$

Another important component of EAs is the fitness function. The role of the fitness function is to assign a quality measure to each individual, which is then used mainly in the selection stages. The term *fitness function* is usually associated with maximization, so it has caused some controversy when applied to minimization problems. However, changing minimization into maximization or vice versa is trivial, so when connecting selection operators and fitness functions both components must consider maximization or minimization. Depending on the selection operators used, some requirements might be fulfilled by the fitness function. For instance, some selectors cannot operate with negative fitness values. Let us consider an even simpler optimization problem, such as the minimization of the function $f(x)$ [Equation (3)], where x is a real number in the range $[0, 15]$. In this case, a plausible fitness function for the minimization of $f(x)$ that guarantees

that the fitness value of any candidate solution is positive is given by f_1 [Equation (4)]. Note that 191 is the maximum value of the function $f(x)$ in the range considered. Usually, this value is not known. However, any value greater than 191 can also be used. Alternatively, a maximization fitness function that might produce negative values is given by the simpler function $f_2(x) = -f(x)$. In this last case, estimating the maximum value of the function is not a requirement. However, with f_2 not every selection operator can be used.

$$f(x) = (x - 1)^2 - 5, \quad (3)$$

$$f_1(x) = -f(x) + 191. \quad (4)$$

The selection operator is another important component that has a large impact on the overall performance of the EAS (Blickle and Thiele, 1996). The aim of selection is to focus the search on the most promising regions. Some of the first selectors designed made their decisions considering only the individuals' fitness values. However, other factors, such as age or diversity, are usually considered in state-of-the-art EAS (Segura *et al.*, 2013a). Selection operators are applied in two different stages: parent selection and replacement. In both cases, stochastic and deterministic operators can be used, though the most popular choice is to apply stochastic operators in the parent selection stage and deterministic operators in the replacement phase (Eiben and Smith, 2003).

A very popular stochastic selection operator is the *fitness proportional* operator. In the fitness proportional selection, the total fitness F is first calculated as the sum of the fitness values of the individuals in the current population. Then, the selection probability of an individual $I_i(p_i)$ is calculated using Equation (5), where *fit* represents the fitness function. In order to apply this selection scheme, the fitness function of every individual must be positive. Let us assume that we are using the fitness function established in Equation (4) and that our current population consists of three individuals with $x = 2$, $x = 5$, and $x = 12$. [Table 14.1](#) shows the fitness value and the selection probability associated with each individual. One of the weaknesses of this operator is that it is susceptible to function transposition, i.e., its behavior changes if the fitness function of every individual is transposed by adding a constant value. This is illustrated in [Figure 14.2](#). In this figure, a population consisting of individuals A, B, and C is used. Two fitness functions are considered: the first one (f_1) is shown in Equation (4), while the second one (f_2) is generated by substituting the value 191 in Equation (4) by the value 1000. [Figure 14.2](#) shows the fitness values of the different individuals, as well as the selection probabilities associated with the individuals when f_1 and f_2 are applied. Since the selection scheme is susceptible to function transposition, these probabilities differ, as can be seen in the pie charts. In addition, note that with f_2 , the selection pressure is very low, i.e., all individuals are selected with very similar probabilities. The reason is that when the differences between the fitness values are small with respect to their absolute values, the selection

pressure vanishes. This last effect is a well-known weakness of fitness proportional selection.

Table 14.1: Fitness values and selection probabilities induced by the fitness proportional selection.

Individual	Fitness	Selection Prob.
$x = 2$	195	0.43
$x = 5$	180	0.40
$x = 12$	75	0.17

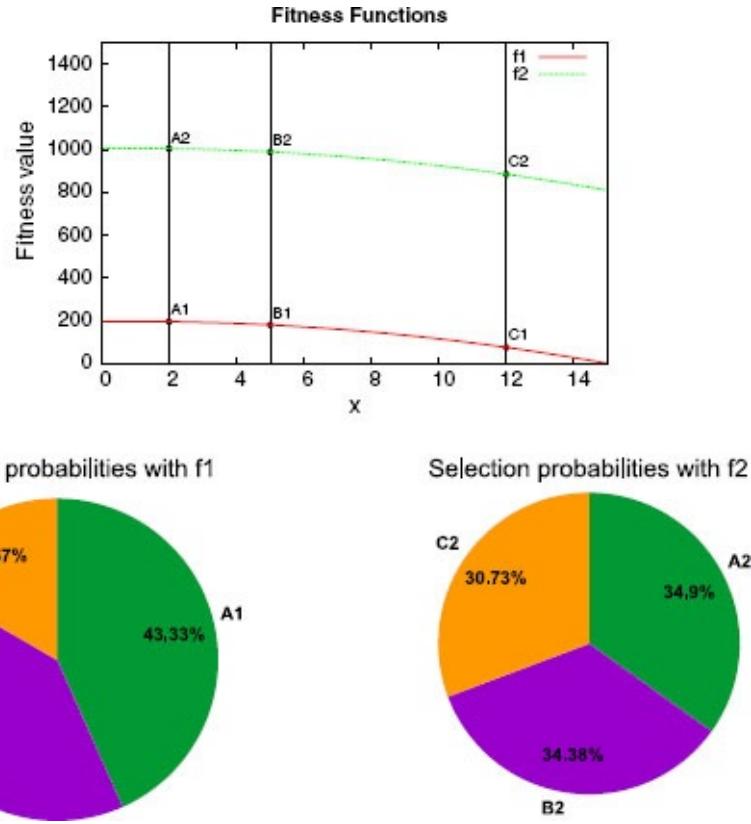


Figure 14.2: Effect of transposition on fitness proportional selection.

$$p_i = \frac{fit(I_i)}{F}. \quad (5)$$

Another quite popular stochastic selection operator is the *tournament selection*. In this case, each time that an individual must be selected, there is a competition between k randomly selected individuals, with the best one being chosen. Tournament selection has very different features than *fitness proportional* selection. For instance, it is invariant to transposition of the fitness function. However, this does not mean that it is superior to *fitness proportional* selection. For instance, when facing some fitness landscapes, *fitness proportional* selection can induce a larger focus on exploitation than *tournament selection*. Depending on the problem at hand, on the stopping criterion and on the other components applied, this might be desired or counterproductive, so it is the task of the designer to analyze the interactions between the selectors and the fitness landscape.

Finally, another quite important stage of EAS is the variation phase. Among the different operators, the crossover and mutation operators are the most popular, and a large

number of different choices have been devised for each. For instance, in (Eiben and Smith, 2003) several operators that can be applied with binary, integer, floating-point, and permutation representations are discussed. In order to better illustrate the working operation of crossover, two popular crossover operators that can be applied with binary, integer, and floating-point representations are presented: one-point and the uniform crossover. One-point crossover (Figure 14.3) operates by choosing a random gene, and then splitting both parents at this point and creating the two children by exchanging the tails. Alternatively, uniform crossover (Figure 14.4) works by treating each gene independently, so the parent associated with each gene is selected randomly. This is implemented by generating a string with L random variables from a uniform distribution over $[0, 1]$, where L is the number of genes. In each position, if the value is below a given value (usually 0.5), the gene is inherited from the first parent; otherwise, it is inherited from the second parent. The second offspring is created using inverse mapping.

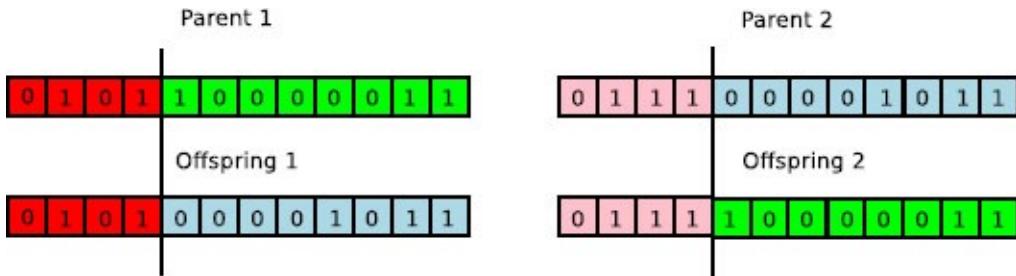


Figure 14.3: Operation of one-point crossover.

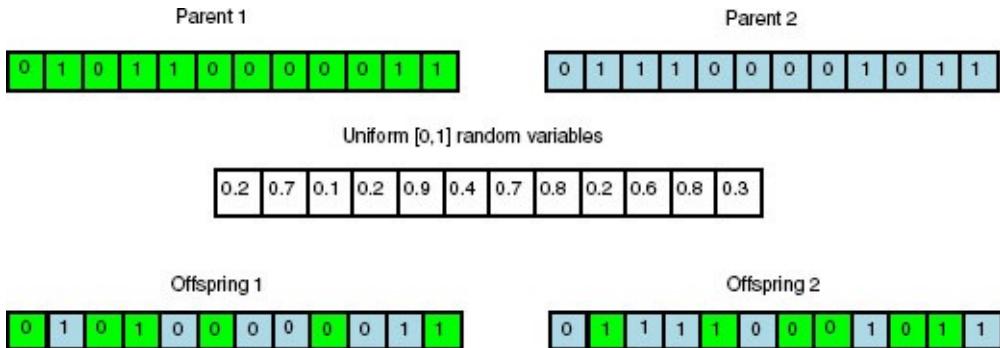


Figure 14.4: Operation of uniform crossover.

It is important to note that the crossover schemes discussed induce different linkages between the genes. For instance, in one-point crossover, genes that are close in the chromosome are usually inherited together, while this is not the case in uniform crossover. Depending on the meaning of each gene and on its position in the chromosome, introducing this kind of linkage might make sense or not. Thus, when selecting the crossover operator, the representation of individuals must be taken into account. Finally, note that as mentioned before, operators that try to learn a proper linkage between genes have been devised (Chen and Lim, 2008). These kinds of operators have provided important benefits in several cases.

Note that not all the components involved in the design of EAS have been presented in this section. In addition, the components discussed are very simple, as most of them were

devised at the time when the foundations of EC were being established. In spite of this, many of them are used in several state-of-the-art schemes, so mastering them is important. Regarding their performance, it is not possible to make any categorical assertions. The reason is that the performance depends on all the components, on the interactions that arise among them, and on the fitness landscape. Depending on the components used, different balances between exploitation and exploration can be induced. It is very important to recognize how to modify this balance because it is one of the keys to success. Thus, when designing EAS to solve new problems, several components and parameterizations are usually tested. By analyzing the performance of different components and the reasons for their good or poor performance, some alternative designs might be tested until an acceptable scheme is obtained. Moreover, note that not only do the components have to be selected, but also a set of parameters have to be adapted to the problem at hand. [Table 14.2](#) enumerates some of the most important aspects to define when designing and running EAS. It is necessary not only to define the representation of individuals, but also to specify all the variation operators and selection schemes involved in the evolution process. We must also decide the criterion for terminating the execution of the algorithm. In addition to all of these aspects, we also have to consider the internal parameters associated with many of the operators mentioned above. Since all of these decisions have an important effect on the algorithm's behavior, parameter tuning strategies should be used at this stage. It is also very important to know the features and implications of using the different components and parameter values and recognize the drawbacks that each of them can circumvent. This is because testing every parameter and component combination is not possible given how time and computationally consuming this task is. Moreover, it might also be desirable to use several components and parameterizations simultaneously. Parameter control techniques can be used for such a purpose. Thus, designing efficient EAS for new problems is a complex task that requires using a variety of tools (León *et al.*, 2009; Liefoghe *et al.*, 2011; Luke *et al.*, 2007) and knowledge that has been developed in recent decades through both experimental and theoretical studies.

Table 14.2: Some aspects to consider when designing EAS.

Component or aspect	Related decisions
Individual	Representation Evaluation (<i>fitness function</i>)
Population	Size Population sizing scheme Initialization
Evolution	Mutation operator Crossover operator Repair operator Intensification operator Probabilities for applying operators Parent selection

	Replacement scheme
Stopping criterion	Number of evaluations of individuals
	Generations
	Time
	Given signs of stagnation
	Achievement of a certain quality of solution

14.6. Concluding Remarks

EC is a very active area of research that studies the design and application of a set of algorithms that draw their inspiration from natural evolution. The first studies that influenced the development of EC date back to at least the 1930s. However, it was during the 1960s when the roots of EC were established. During the first stages of EC, several independent researchers devised different schemes that were applied to a diverse number of applications. Three different types of schemes were developed: EP, ESS, and GAS. Although each type of scheme had its own features, further developments made it clear that these schemes had more in common than not, so they began to merge naturally. In the 1990s, the terms EC and EA were proposed with the aim of unifying the terminology and they were soon widely adopted by the community. In fact, it is now very difficult to claim that a given state-of-the-art EA belongs to any of the previous classes because they usually combine components from several types of schemes. The number of studies that has been carried out in the last decades is vast. These studies have yielded significant advances in EC; as a result, the number of problems that have been addressed with EAS has grown enormously.

One of the main handicaps of EC is that mastering EAS is a very complex task. One of the reasons is that despite the huge amount of research conducted in recent years, the discipline cannot yet be considered as fully mature. For instance, we are still in a period where different researchers have opposing views with respect to several aspects of EAS, so much more research is still required to fully understand EAS and the large number of different components that have been devised by various authors. Fortunately, many tools that facilitate the use of EAS are now available and recent years have seen significant efforts made to facilitate the application of EAS. In the near future, we expect this area to grow even more as it continues to develop on its path to maturity.

Acknowledgments

The first author gratefully acknowledges the support obtained from CONACyT Project No. 221551. The second author acknowledges the financial support from CONCYTEG as part of the plan “Investigadores Jóvenes — DPP-2014” (project14-IJ-DPP-Q182-11).

References

- Alba, E. (2005). *Parallel Metaheuristics: A New Class of Algorithms*. NJ, USA: John Wiley & Sons.
- Altenberg, L. (1995). The schema theorem and Price's theorem. In Whitley, D. and Vose M. D. (eds.), *Foundations of Genetic Algorithms 3*. San Francisco: Morgan Kaufmann, pp. 23–49.
- Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford, UK: Oxford University Press.
- Bäck, T., Fogel, D. B. and Michalewicz, Z. (1997). *Handbook of Evolutionary Computation*, 1st edn. Bristol, UK: IOP Publishing Ltd.
- Bäck, T., Foussette, C. and Krause, P. (2013). *Contemporary Evolution Strategies*, Natural Computing Series. New York: Springer.
- Bäck, T., Hoffmeister, F. and Schwefel, H.-P. (1991). A survey of evolution strategies. *Proc. Fourth Int. Conf. Genetic Algorithms*, Morgan Kaufmann, pp. 2–9.
- Bäck, T., Rudolph, G. and Schwefel, H.-P. (1993). Evolutionary programming and evolution strategies: Similarities and differences. In Fogel, D. B. and Atmar, J. W. (eds.), *Second Annual Conference on Evolutionary Programming*. San Diego, CA, pp. 11–22.
- Bäck, T. and Schwefel, H.-P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evol. Comput.*, 1(1), pp. 1–23.
- Barricelli, N. A. (1962). Numerical testing of evolution theories. *Acta Biotheor.* 16(1–2), pp. 69–98.
- Beyer, H.-G. and Sendhoff, B. (2008). Covariance matrix adaptation revisited — the CMSA evolution strategy. In Rudolph, G., Jansen, T., Lucas, S., Poloni, C. and Beume, N. (eds.), *Parallel Problem Solving from Nature — PPSN X, Lecture Notes in Computer Science*, Vol. 5199. Berlin Heidelberg: Springer, pp. 123–132.
- Bledsoe, W. W. (1961). The use of biological concepts in the analytical study of systems. Tech. Rep. Technical report of talk presented to ORSA-TIMS national meeting, San Francisco, CA.
- Blickle, T. and Thiele, L. (1996). A comparison of selection schemes used in evolutionary algorithms. *Evol. Comput.*, 4(4), pp. 361–394.
- Bossert, W. (1967). Mathematical optimization: Are there abstract limits on natural selection? In Moorehead, P. S. and Kaplan, M. M. (eds.), *Mathematical Challenges to the Neo-Darwinian Interpretation of Evolution*. Philadelphia, PA: The Wistar Institute Press, pp. 35–46.
- Box, G. E. P. (1957). Evolutionary operation: A method for increasing industrial productivity. *Appl. Stat.*, 6(2), pp. 81–101.
- Bremermann, H. J. (1962). Optimization through evolution and recombination. In Yovits, M. C., Jacobi, G. T. and Golstine, G. D. (eds.), *Proceedings of the Conference on Self-Organizing Systems*. Washington, DC: Spartan Books, pp. 93–106.
- Burgin, M. and Eberbach, E. (2013). Recursively generated evolutionary turing machines and evolutionary automata. In Yang, X.-S. (ed.), *Artificial Intelligence, Evolutionary Computing and Metaheuristics, Studies in Computational Intelligence*, Vol. 427. Berlin Heidelberg: Springer, pp. 201–230.
- Campbell, D. T. (1960). Blind variation and selective survival as a general strategy in knowledge-processes. In Yovits, M. C. and Cameron, S. (eds.), *Self-Organizing Systems*. New York: Pergamon Press, pp. 205–231.
- Cannon, W. B. (1932). *The Wisdom of the Body*. W. W. Norton & Company, Inc.
- Chellapilla, K. (1997). Evolving computer programs without subtree crossover. *IEEE Trans. Evol. Comput.* 1(3), pp. 209–216.
- Chen, Y. and Lim, M. H. (2008). *Linkage in Evolutionary Computation*, Studies in Computational Intelligence. New York: Springer.
- Coello Coello, C. A., Lamont, G. B. and Veldhuizen, D. A. V. (2006). *Evolutionary Algorithms for Solving Multi-Objective Problems*, Genetic and Evolutionary Computation. Secaucus, NJ, USA: Springer-Verlag New York, Inc.
- Conrad, M. and Pattee, H. H. (1970). Evolution experiments with an artificial ecosystem. *J. Theor. Biolo.* 28(3), pp. 393–

- Cornett, F. N. (1972). *An application of evolutionary programming to pattern recognition*. Master's thesis, Las Cruces.
- Črepinský, M., Liu, S.-H. and Merník, M. (2013). Exploration and exploitation in evolutionary algorithms: A survey. *ACM Comput. Surv.*, 45(3), pp. 35:1–35:33.
- Crosby, J. L. (1960). The use of electronic computation in the study of random fluctuations in rapidly evolving populations. *Philos. T. Roy. Soc. B*, 242(697), pp. 550–572.
- Crosby, J. L. (1967). Computers in the study of evolution. *Science Progress Oxford*, 55, pp. 279–292.
- Darwin, C. (1859). *On the Origin of Species by Means of Natural Selection* (Murray), or *the Preservation of Favoured Races in the Struggle for Life*.
- Darwin, C. and Wallace, A. (1858). On the tendency of species to form varieties; and on the perpetuation of varieties and species by natural means of selection. *Zool. J. Linn. Soc.* 3, pp. 46–50.
- De Jong, K. A. (1975). An analysis of the behavior of a class of genetic adaptive systems. Ph.D. thesis, Ann Arbor, MI, USA.
- De Jong, K. A. (2006). *Evolutionary Computation — A Unified Approach*. MIT Press.
- de Monet de Lamarck, J. B. P. A. (1809). *Philosophie zoologique: Ou Exposition des considérations relative à l'histoire naturelle des animaux*.
- Dearholt, D. W. (1976). Some experiments on generalization using evolving automata. *9th Hawaii Int. Conf. System Sciences*, Western Periodicals, Honolulu, pp. 131–133.
- Dunham, B., Fridshal, D., Fridshal, R. and North, J. H. (1963). Design by natural selection. *Synthese*, 15(1), pp. 254–259.
- Eiben, A. E., Aarts, E. H. L. and Hee, K. M. (1991). Global convergence of genetic algorithms: A Markov Chain analysis. In Schwefel, H.-P. and Männer, R. (eds.), *Parallel Problem Solving from Nature, Lecture Notes in Computer Science*, Vol. 496. Berlin Heidelberg: Springer, pp. 3–12.
- Eiben, A. E. and Bäck, T. (1997). Empirical investigation of multiparent recombination operators in evolution strategies. *Evol. Comput.*, 5(3), pp. 347–365.
- Eiben, A. E. and Smith, J. E. (2003). *Introduction to Evolutionary Computing*, Natural Computing Series. New York: Springer.
- Eshelman, L. (1991). The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. *Foundations of Genetic Algorithms*, pp. 265–283.
- Evans, C. R. and Robertson, A. D. J. (eds.) (1968). *Cybernetics: Key Papers*. Baltimore, MD, USA: University Park Press.
- Everett, J. E. (2000). Model building, model testing and model fitting. In *The Practical Handbook of Genetic Algorithms: Applications*, 2nd edn., Boca Raton, FL, USA: CRC Press, Inc.
- Fisher, R. (1930). *The Genetical Theory of Natural Selection*. Oxford University Press.
- Fogel, D. B. (1994). An introduction to simulated evolutionary optimization. *IEEE Trans. Neural Netw.*, 5(1), pp. 3–14.
- Fogel, D. B. (1995). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. Piscataway, NJ, USA: IEEE Press.
- Fogel, D. B. (1998). *Evolutionary Computation: The Fossil Record*. New York: Wiley-IEEE Press.
- Fogel, D. B. and Anderson, R. W. (2000). Revisiting Bremermann's genetic algorithm. I. Simultaneous mutation of all parameters. In *2000 IEEE Congress on Evolutionary Computation*, Vol. 2, pp. 1204–1209.
- Fogel, D. B. and Atmar, J. W. (1990). Comparing genetic operators with gaussian mutations in simulated evolutionary processes using linear systems. *Biol. Cybern.*, 63(2), pp. 111–114.
- Fogel, D. B. and Chellapilla, K. (1998). Revisiting evolutionary programming, pp. 2–11.
- Fogel, D. B., Fogel, L. J. and Atmar, J. W. (1991). Meta-evolutionary programming. In *Asilomar Conference in Signals Systems and Computers*, pp. 540–545.
- Fogel, D. B. and Ghozeil, A. (1997). A note on representations and variation operators. *IEEE Trans. Evol. Comput.*, 1(2),

pp. 159–161.

- Fogel, L. J. (1962). Autonomous automata. *Industrial Research*, 4, pp. 14–19.
- Fogel, L. J. (1999). *Intelligence Through Simulated Evolution: Forty Years of Evolutionary Programming*. Wiley series on intelligent systems. New York: Wiley.
- Fogel, L. J., Angeline, P. J. and Fogel, D. B. (1995). An evolutionary programming approach to self-adaptation on finite state machines. *Annual Conference on Evolutionary Programming IV*, pp. 355–365.
- Fogel, L. J. and Burgin, G. H. (1969). Competitive goal-seeking through evolutionary programming. Tech. Rep. Contract AF 19(628)-5927, Air Force Cambridge Research Labs.
- Fogel, L. J. and Fogel, D. B. (1986). Artificial intelligence through evolutionary programming. Tech. Rep. PO-9-X56-1102C-1, U.S. Army Research Institute, San Diego, CA.
- Fogel, L. J., Owens, A. J. and Walsh, M. J. (1966). *Artificial Intelligence Through Simulated Evolution*. New York: John Wiley & Sons.
- Fraser, A. S. (1957). Simulation of genetic systems by automatic digital computers. I. Introduction. *Aust. J. Bio. Sci.*, 10, pp. 484–491.
- Fraser, A. S. and Burnell, D. (1967). Simulation of genetic systems XII. Models of inversion polymorphism. *Genetics*, 57, pp. 267–282.
- Fraser, A. S. and Burnell, D. (1970). *Computer Models in Genetics*. NY: McGraw-Hill.
- Friedberg, R. M. (1958). A learning machine: Part I. *IBM J. Res. Dev.*, 2(1), pp. 2–13.
- Friedberg, R. M., Dunham, B. and North, J. H. (1959). A learning machine: Part II. *IBM J. Res. Dev.*, 3(3), pp. 282–287.
- Friedman, G. J. (1956). Selective feedback computers for engineering synthesis and nervous system analogy. Master's thesis, University of California, Los Angeles.
- Gill, J. L. (1965). Effects of finite size on selection advance in simulated genetic populations. *Aust. J. Biol. Sci.*, 18, pp. 599–617.
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Artificial Intelligence. New York: Addison-Wesley.
- Goldberg, D. E., Korb, B. and Deb, K. (1989). Messy genetic algorithms: Motivation, Analysis, and First Results. *Complex Systems*, 3(5), pp. 493–530.
- Goldberg, D. E. and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. *Proc. Second Int. Conf. Genetic Algorithms and their Application*, L. Erlbaum Associates Inc., Hillsdale, NJ, USA, pp. 41–49.
- Haldane, J. B. S. (1932). *The Causes of Evolution*. Longman, Green and Co.
- Hansen, N. and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, 9(2), pp. 159–195.
- Heckendorn, R. B., Whitley, D. and Rana, S. (1996). Nonlinearity, Hyperplane Ranking and the Simple Genetic Algorithm. In Belew, R. K. and Vose, M. D. (eds.), *Foundations of Genetic Algorithms 4*, pp. 181–201.
- Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: University of Michigan Press.
- Huxley, J. S. (1942). *Evolution: The Modern Synthesis*. Allen and Unwin.
- Igel, C., Hansen, N. and Roth, S. (2007). Covariance matrix adaptation for multi-objective optimization. *Evol. Comput.*, 15(1), pp. 1–28.
- Jain, A. and Fogel, D. B. (2000). Case studies in applying fitness distributions in evolutionary algorithms. II. Comparing the improvements from crossover and Gaussian mutation on simple neural networks. *2000 IEEE Sympo. Combinations of Evolutionary Computation and Neural Networks*, pp. 91–97.
- Kaufman, H. (1967). An experimental investigation of process identification by competitive evolution. *IEEE Trans. Syst. Sci. Cybern.*, 3(1), pp. 11–16.
- Koza, J. R. (1990). Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems. Tech. Rep., Stanford, CA, USA.

- León, C., Miranda, G. and Segura, C. (2009). Metco: A parallel plugin-based framework for multi-objective optimization. *Int. J. Artificial Intelligence Tools*, 18(4), pp. 569–588.
- Lewontin, R. C. (1964). The interaction of selection and linkage. I. General considerations; Heterotic models. *Genetics*, 49(1), pp. 49–67.
- Li, R., Emmerich, M. T. M., Eggermont, J., Bäck, T., Schütz, M., Dijkstra, J. and Reiber, J. H. C. (2013). Mixed integer evolution strategies for parameter optimization. *Evol. Comput.*, 21(1), pp. 29–64.
- Liefooghe, A., Jourdan, L. and Talbi, E.-G. (2011). A software framework based on a conceptual unified model for evolutionary multiobjective optimization: Paradiseomoeo. *Eur. J. Oper. Res.*, 209(2), pp. 104–112.
- Linnaeus, C. (1735). *Systema Naturae, Sive Regna Tria Naturae Systematice Proposita Per Secundum Classes, Ordines, Genera, & Species, Cum Characteribus, Differentiis, Synonymis, Locis*. Theodorus Haak.
- Lobo, F. G., Lima, C. F. and Michalewicz, Z. (2007). *Parameter Setting in Evolutionary Algorithms*, 1st edn. New York: Springer Publishing Company, Inc.
- Luke, S., Panait, L., Balan, G. and Et (2007). ECJ: A Java-based Evolutionary computation research system.
- Lyle, M. R. (1972). An investigation into scoring techniques in evolutionary programming. Master's thesis, Las Cruces.
- Mendel, G. (1865). Experiments in plant hibridization. In *Brünn Natural History Society*, pp. 3–47.
- Mezura-Montes, E. (2009). *Constraint-Handling in Evolutionary Optimization*, 1st edn. New York: Springer.
- Mezura-Montes, E. and Coello Coello, C. A. (2005). A simple multimembered evolution strategy to solve constrained optimization problems. *IEEE Trans. Evol. Comput.*, 9(1), pp. 1–17.
- Michalewicz, Z. (1994). *Genetic Algorithms + Data Structures = Evolution Programs*. New York, NY, USA: Springer-Verlag New York, Inc.
- Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press.
- Moscato, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Tech. Rep. C3P Report 826, California Institute of Technology.
- Nix, A. and Vose, M. D. (1992). Modeling genetic algorithms with Markov chains. *Ann. Math. Artif. Intell.* 5(1), pp. 79–88.
- Ostermeier, A. (1992). An evolution strategy with momentum adaptation of the random number distribution. In Männer, R. and Manderick, B. (eds.), *Parallel Problem Solving from Nature 2, PPSN-II*. Brussels, Belgium: Elsevier, pp. 199–208.
- Ostermeier, A., Gawelczyk, A. and Hansen, N. (1994a). A derandomized approach to self-adaptation of evolution strategies. *Evol. Comput.*, 2(4), pp. 369–380.
- Ostermeier, A., Gawelczyk, A. and Hansen, N. (1994b). Step-size adaptation based on nonlocal use of selection information. In Davidor, Y., Schwefel, H.-P. and Männer, R. (eds.), *Parallel Problem Solving from Nature PPSN III, Lecture Notes in Computer Science*, Vol. 866. Springer: Berlin Heidelberg, pp. 189–198.
- Poland, J. and Zell, A. (2001). Main vector adaptation: A CMA variant with linear time and space complexity. In Spector, L., Goodman, E. D., Wu, A., Langdon, W. B., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. H. and Burke E. (eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*. San Francisco, California, USA: Morgan Kaufmann, pp. 1050–1055.
- Porto, V. W. and Fogel, D. B. (1995). Alternative neural network training methods [active sonar processing]. *IEEE Expert*, 10(3), pp. 16–22.
- Radcliffe, N. J. (1997). Schema processing. In Bäck, T., Fogel, D. B. and Michalewicz, Z. (eds.), *Handbook of Evolutionary Computation*. Bristol, New York: Institute of Physics Publishing and Oxford University Press, pp. 1–10.
- Rechenberg, I. (1965). Cybernetic solution path of an experimental problem. Tech. Rep. Library Translation 1122, Royal Air Force Establishment.
- Rechenberg, I. (1973). *Evolutions strategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart: Frommann-Holzboog.
- Rechenberg, I. (1978). Evolutionsstrategen. In Schneider, B. and Ranft, U. (eds.), *Simulations methoden in der Medizin*

- und Biologie*. Berlin, Germinay: Springer-Verlag, pp. 83–114.
- Reed, J., Toombs, R. and Barricelli, N. A. (1967). Simulation of biological evolution and machine learning: I. Selection of self-reproducing numeric patterns by data processing machines, effects of hereditary control, mutation type and crossing. *J. Theor. Biol.* 17(3), pp. 319–342.
- Richter, H. (2014). Fitness landscapes: From evolutionary biology to evolutionary computation. In Richter, H. and Engelbrecht, A. (eds.), *Recent Advances in the Theory and Application of Fitness Landscapes, Emergence, Complexity and Computation*, Vol. 6. Berlin Heidelberg: Springer, pp. 3–31.
- Richter, J. N., Wright, A. and Paxton, J. (2008). Ignoble trails — where crossover is provably harmful. In Rudolph, G., Jansen, T., Lucas, S., Poloni, C. and Beume, N. (eds.), *Parallel Problem Solving from Nature PPSN X, Lecture Notes in Computer Science*, Vol. 5199. Berlin Heidelberg: Springer, pp. 92–101.
- Rudolph, G. (2012). Evolutionary strategies. In Rozenberg, G., Bäck, T. and Kok, J. (eds.), *Handbook of Natural Computing*. Berlin Heidelberg: Springer, pp. 673–698.
- Schraudolph, N. and Belew, R. (1992). Dynamic parameter encoding for genetic algorithms. *Machine Learning*, 9(1), pp. 9–21.
- Schwefel, H.-P. (1965). *Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik*. Dipl.-Ing. Thesis, Technical University of Berlin, Hermann Föttinger–Institute for Hydrodynamics.
- Schwefel, H.-P. (1981). *Numerical Optimization of Computer Models*. New York, NY, USA: John Wiley & Sons, Inc.
- Segura, C., Coello, C., Segredo, E., Miranda, G. and Leon, C. (2013a). Improving the diversity preservation of multi-objective approaches used for single-objective optimization. *2013 IEEE Congress on Evolutionary Computation (CEC)*, pp. 3198– 3205.
- Segura, C., Coello Coello, C. A., Miranda, G. and León, C. (2013b). Using multi-objective evolutionary algorithms for single-objective optimization. *4OR* 11(3), pp. 201–228.
- Sivanandam, S. N. and Deepa, S. N. (2007). *Introduction to Genetic Algorithms*. New York: Springer.
- Storn, R. and Price, K. (1995). Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces. Tech. rep., International Computer Science Institute, Berkeley, Tech. Rep. TR95012.
- Syswerda, G. (1989). Uniform crossover in genetic algorithms. In *Proc. 3rd Int. Con. Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 2–9.
- Turing, A. M. (1948). Intelligent Machinery. Report, National Physical Laboratory, Teddington, UK.
- Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59, pp. 433–460.
- Vose, M. D. and Liepins, G. (1991). Punctuated equilibria in genetic search. *Complex Systems*, 5, pp. 31–44.
- Whitley, D. and Kauth, J. (1988). *GENITOR: A different genetic algorithm*. Colorado State University, Department of Computer Science.
- Whitley, D. and Sutton, A. (2012). Genetic algorithms — A survey of models and methods. In Rozenberg, G., Bäck, T. and Kok, J. (eds.), *Handbook of Natural Computing*. Berlin Heidelberg: Springer, pp. 637–671.
- Wright, S. (1931). Evolution in mendelian populations. *Genetics*, 16, pp. 97–159.
- Wright, S. (1932). The roles of mutation, in breeding, crossbreeding and selection in evolution. VI *International Congress of Genetics*, Vol. 1, pp. 356–366.
- Yao, X. and Liu, Y. (1997). Fast evolution strategies. In Angeline, P., Reynolds, R., McDonnell, J. and Eberhart, R. (eds.), *Evolutionary Programming VI, Lecture Notes in Computer Science*, Vol. 1213. Berlin Heidelberg: Springer, pp. 149–161.
- Yao, X., Liu, Y. and Lin, G. (1999). Evolutionary programming made faster. *IEEE Trans. Evol. Comput.*, 3(2), pp. 82–102.
- Young, S. S. (1966). Computer simulation of directional selection in large populations. I. the programme, the additive and the dominance models. *Genetics*, 53, pp. 189–205.

¹ Some preliminary work on coevolution was started in (Barricelli, 1962).

² To the best of the authors' knowledge, Bäck *et al.* (1991) was the first paper where this definition was given.

³ The term building block has also been used with slightly different definitions (Radcliffe, 1997).

⁴ See <http://ls11-www.cs.uni-dortmund.de/rudolph/ppsni>.

⁵ The conference originally known as "International Conference on Genetic Algorithms" was renamed "Genetic and Evolutionary Computation Conference".

Chapter 15

A Survey of Recent Works in Artificial Immune Systems

Guilherme Costa Silva and Dipankar Dasgupta

This report aims to bring the state-of-the-art of artificial immune systems (AISs) and the literature review during the last five years, presenting some Immune-based methods, their applications and provides several considerations on how immune analogies can contribute to the development of more efficient and robust immunity-inspired algorithms.

15.1. Introduction

Artificial immune systems (AISs), alternatively known as immune-inspired systems or immuno-logical computation, is a research line which are developed based on abstractions derived from the biological immune system. Many researches related to computational algorithms have used immunological abstractions as an inspiration, through algorithms based on some immunological theories or models and capable of solving computational or engineering real-world applications.

The biological immune system is an autonomic system able to provide a defense mechanism against possible diseases and threats to the body. Many studies have been performed in order to explain exactly how this mechanism is important in biology. Each theory or model of immune systems can provide analogies to many computational and engineering problems, these analogies serve as metaphors for the development of methods and techniques to reproduce some features found in the biological immune system and then, provide an efficient solution to a specific problem.

These systems can be applied to a wide set of problems in Engineering and in Computer Science, such as optimization, clustering, information retrieval, anomaly detection, machine learning and pattern recognition, among other problems, adopting different abstractions associated to system components or the application environment, depending on the context of the problem studied.

AIS has been widely studied throughout the years, with several approaches and applications in literature, with the development and improvement of these, as well as their applications which have become very popular in literature. In addition, there are some other aspects that can be considered as relevant and recent remarks in the research, such as the increasing of hybrid AIS approaches, the contributions of research in some multidisciplinary studies, and the contributions of AIS approaches in the immunology whose impact is so far unknown, among others.

This survey aims to present the highlights of AIS research during the last five years, including new approaches, improvements in classical techniques and reviews in these algorithms, as well as their applications in different problems in literature. This paper will also review some information about AIS according to biological points of view, but often presenting their relation to the AIS researches. In addition, the survey in (Dasgupta *et al.*, 2011), will be revisited according to these aspects. Further sections will discuss the state-of-the-art, approaches according to analogies and applications of models.

15.2. State-of-the-Art

Early works, starting from (Forrest *et al.*, 1994), are focused on the Self/Non-self Discrimination principle applied to anomaly detection. These methods consist of supervised learning based techniques inspired on the negative selection process, which has a censoring feature occurred in the thymus. The Negative Selection Algorithms (NSA) use Self-based training data for detectors generation through Non-self space, which may be equivalent to an anomaly detection.

The reason for inspiration in the biological immune system may be related to its robustness and the faster response provided in the human immune system, in which a more effective and faster response to an antigen is expected for its next occurrences, after the first time this same antigen was seen in the body. The expected objective of using an immune-inspired approach, once applied to an engineering problem, is to provide a similar behavior for an engineering problem solution, mainly in the concept of anomaly detection.

A first survey about AIS was done in (Dasgupta and Attoh-Okine, 1997). A brief description about immune models, such as Immune Networks and Negative Selection, and applications of first immune-inspired approach, such as Computer Security, Pattern Recognition, Time Series Anomaly Detection and Fault Diagnosis, were reported in the survey, and then, in the first book about these systems in, (Dasgupta, 1998).

The development of novel engineering tools based on immunological theories and models have been provided for many applications as seen on (De Castro and Timmis, 2002), in which a framework for the development of immune-inspired engineering systems, as well as their applications, comparisons to other computational intelligence paradigms and also proposal for some hybrid systems were considered. Since then, applications such as Optimization approaches based on the Clonal Selection Theory or on the B cells, Pattern Recognition approaches and Clustering approaches based on the Immune Idiotypic Network.

Novel approaches to anomaly detection based on alternate models of the biological immune system, like the infectious non-self and danger models, have appeared. These models, implied by their biological counterpart, may depend on prior expert knowledge which defines normal and abnormal events, suggested by biological analogies, as the Self-Non-self principle, which relies on the Negative Selection process, analogous to a supervised machine learning method that defines the normal and anomalous feature space.

Most aspects of these algorithms, including alternate versions and improvements developed, are seen in (Dasgupta and Niño, 2008), a book in which many aspects and most immunological computation objectives and proposals are reviewed and discussed. This book has the most updated information about the research field so far. This survey will cover most aspects of AIS within these last five years, discussing novel approaches and other aspects of the research, as well as providing additional information on how AIS

research has been lead during these years in terms of models, applications and approaches developed.

15.3. Artificial Immune System Approaches

Several immune-inspired systems have been developed in order to provide more efficient results for problems such as anomaly detection, optimization, clustering, time series and pattern recognition, among other applications.

Algorithms that adopt multiple models for immunological abstraction (i.e., clonal selection algorithm (CSA) applied to optimization of negative selection algorithms), the predominant model related to its research and applications focus will be considered in the list of methods in each subsection, since these methods can be combined and are not mutually exclusive.

15.3.1. Immune Response Models

In biology, the main bottleneck was to define how the nature of the immune responses is in fact. The most accepted model is related to the concepts of Self-non-self Discrimination, which consists in recognizing foreign antigens. This model has inspired the computer virus detection approach of Forrest *et al.* (1994), and many approaches and improvements were done since then.

As in biology, it was discovered that “non-self” antigens alone are not sufficient to trigger T cell activation, the model of costimulation signals would explain this issue, and then, other models of immune response were proposed to explain several immunological phenomena. These models have offered other analogies for AIS approaches, mainly related to anomaly detection systems, which would be applied to different problems, depending on the analogy.

Each approach has differences related to data analysis. However, all of them have a transitional link as described in (Costa Silva *et al.*, 2012b). With complementary functions to be further discussed, each immunological model has advantages and disadvantages that should be taken into account for each application environment.

Usually, Self-non-self and negative selection approaches are seen as a distinct group in AIS approaches, as well as immune networks and clonal selection approaches. In this survey, Self-non-self approaches and most models of immune response are grouped into algorithms based on the immune response, since these techniques are applied to anomaly detection and they have points in common, being a part of a transitional link.

The response models which inspire anomaly detection systems will be further discussed regarding the corresponding model, usually based on T cells or their interaction with antigen presenting cells (APCs), but not limited to this analogy. Since the immune system analogy is closely related to the anomaly detection problem, by providing a response to pathogens which may threaten the body, most algorithms are related to anomaly detection, but not all of them, as there are other aspects to be explored in these models. The following section will bring some concepts and advances in these

approaches.

15.3.1.1. *Self-non-self/negative selection-based algorithms*

The concept of Self-non-self Discrimination defines the Biological Immune System with the purpose to defend the body against foreign substances (non-self). Considering these substances as potential causes of diseases, the body must respond to these. For this purpose, the immune cells, especially lymphocytes, should not recognize the substances in the body (Self) through a maturation process, known as the negative selection. This process eliminates all lymphocytes which have high affinity to self antigens.

In order to improve the training mechanism of these algorithms, several approaches were considered, as the *V-detector*, a well-developed Negative Selection-based approach with coverage estimation. This algorithm was exposed to sensitivity tests in (Ji and Dasgupta, 2009), in order to expose its advantages and drawbacks.

Some other improvements for NSA were employed, as in (Almeida *et al.*, 2010) with some operators such as radius and detector overlap checking, in (Li *et al.*, 2010) with an outlier robust inspired by immune suppression and applied for noise in (Wang *et al.*, 2011) with boundary management for detectors and in (Gong *et al.*, 2012) with a further training mechanism. Detector generation aspects were reviewed in (Jinyin and Dongyoung, 2011), and the cooperative co-evolution detector generation, a parallel gene-based model, is included. In (Wang *et al.*, 2012), fractional distances are tested in order to verify their applicability to high dimensional problems, since data relative contrast is related to data dimensionality.

The development of NS-based approaches without detectors was considered in (Liskiewicz and Textor, 2010), for string-based applications. In (Costa Silva *et al.* 2012), a fuzzy view of negative and positive selection processes was discussed with approaches proposed. In this view, antigen recognition has a fuzzy nature and the objective of thymic selection is the maturation of cells with an intermediate affinity to self. This model considers both positive and negative selection mechanisms. Two approaches were proposed based on the model, one of them without detector generation but considering the non-self space as a deviation of self patterns according to a fuzzy inference system.

Recently, shape-space representations were evaluated in (McEwan and Hart, 2009), then, some classifiers and an alternative representational abstraction based on linear squares were presented to demonstrate the flaws of a n -dimensional-based technique and how they can be outperformed by alternative representations. The basis discovery and decomposition in the immune repertoire representation were represented by a Matching Pursuit like process and based on equivalent algebraic and algorithmic mechanisms. The shape-space representation for machine learning AIS was considered inferior, because it is considered to reduce potential value of immune-inspired approaches.

Some other interesting improvements or applications of self-non-self-based methods are listed in [Table 15.1](#).

The main advantages of negative selection-based algorithms are the ease implementation and intuitive principle, based on the input training data, the shape-space concept and distance metrics like euclidean or hamming distance, depending on the data. The training phase of the algorithm also favors the approach by the management of the non-self Space. However, the algorithm has issues regarding the test phase, mainly because of its limitations due to the *curse of dimensionality* issue. In addition, these approaches may have some context issues that may represent a hindrance to a proper anomaly detection.

15.3.1.2. *The infectious non-self/pattern recognition receptors-based algorithms*

Some models were proposed in order to provide a proper explanation about biological immune system and its problems. One of them was the Infectious Non-self model, which extends the two-signal costimulatory model and tries to explain some elements that could lead to an immune response.

Table 15.1: Some other negative selection-based related works.

Approach	Reference	Description	Application
Negative Selection	(Poggolini and Engelbrecht, 2013)	Introduction of a feature detection rule with the correlation between rules of a particular problem domain.	Car evaluation dataset
Negative Selection	(Ghosh and Srinivasan, 2011)	Use of spherical detectors in the non-self space as detectors with a boundary detection feature.	The Distillation Column Startup Case Study
Arisytsis	(Ma <i>et al.</i> , 2008)	Addition of an antigen feedback mechanism that provides maturing process and once activated by more antigens they become legitimate detectors.	Intrusion Detection Dataset (KDD Cup'99)
T-detector Maturation Algorithm (TMA)	(Chen <i>et al.</i> , 2012)	Addition of overlap rate metric, suppression operator to control distance among detectors according to logic operators and the combination of these features and of a self-radius learning mechanism in another version.	—
NSA with Mahalanobis Distance (MDNS)	(Chen, 2013)	Employment of Mahalanobis distance metric to improve the classification power and provide more quality to detectors.	Medical diagnosis and quality inspection problems
Negative Selection	(Mahapatra <i>et al.</i> , 2013)	Use of NSA as an operator that combines the different results of maximum information entropy and global thresholding.	Image thresholding problems
NSA	(Zeng <i>et al.</i> , 2013)	Improvement of NSA through a bidirectional matching rule for a detector.	—
NSA	(Chen <i>et al.</i> , 2013)	Addition of operators based on Clonal Selection.	Wireless sensor network problems
GB-RNSA	(Chen <i>et al.</i> , 2013)	Use of detectors allocation based on the coverage of grids in non-self space. Comparison of its performance to real-valued NSA and V-Detector.	—
Evolutionary NSA	(Xu <i>et al.</i> , 2009)	Evaluation of complexity and influence of self set in average time of the algorithm.	Anomaly Detection problems
Multilevel Immune Learning Algorithm	(Chen, 2013)	Improvements in the algorithm regarding the detection generation using a variable-recessive based threshold.	Intrusion Detection problems

Self-non-self based method	(Wang and Jiang, 2011)	“A dynamically assigned sense of self of the protected program”, which aims to prevent re-classification of data in problems.	Malware Detection problems
Self-non-self based method	(Secker <i>et al.</i> , 2008)	Application of cloning, mutation and memory mechanisms and a confirmation signal to affinity of information based in costimulatory signals.	Web mining
Negative Database (NDB)	(Zhao and Luo, 2013)	Use of real-valued representation in an alternative to binary representation to improve reversal of database.	Privacy-preserving data publication problems
Negative Survey	(Bao <i>et al.</i> , 2013)	Review of methods to estimate positive surveys.	Information retrieval
Negative Survey	(Groat <i>et al.</i> , 2013)	An alternate application of negative surveys.	Participatory sensing applications
Negative Authentication	(Dasgupta and Azeem, 2008)	Generation of invalid credentials (anti-passwords) in order to detect or filter invalid attempts of authentication.	Computer security
Negative Authentication	(Dasgupta and Saha, 2010)	Validation and sensitive analysis (number of passwords, confusion parameters and coverage) of the model.	Computer security

An approach considered in this group was the Conserved Self Pattern Recognition Algorithm, applied to a recurrent database of anomaly detection problems and improved with selective mechanisms and tests performed with comparative datasets among different approaches. The algorithm is also based on the Negative Selection, however, with costimulatory mechanisms based in the Pattern Recognition Receptors. An improved version of the algorithm applied to computer security was developed in (Yu and Dasgupta, 2011) with a near deterministic mechanism of detector allocation.

The Infectious Non-self model has also inspired a Dendritic Cell-based approach: The Toll-like Receptor Algorithm was designed to intrusion detection problems with training mechanisms and a simplified signal scheme. The interaction between APCs and T cells is the basis of the algorithm. A Structured version of the algorithm was proposed in (Nejad *et al.*, 2012), this version considers the Non-self space analysis a criterion for antigen recognition after signal exposure.

A NSA-based approach with PRR inspired mechanisms was proposed in (Zheng *et al.*, 2013). The named PRR-2NSA combines the inspiration on Pattern Recognition Receptors, whose data are generated via hard clustering and dissimilarity measurements, and a “Dual Negative Selection” strategy, in which NSA is applied to classifier maturation to assure it does not match with any other classifier, and once again to training data, to assure it does not match with any normal data. The proposed algorithm is tested to dataset benchmarks and is compared to the 2NSA without PRR and the V-detector, with better performance. A similar strategy for APC data generation was adopted in the semi-supervised Immune Classifier Generation Algorithm (SICGA) in (Ying, 2003), but with APC and PRR data

being generated based on k-means algorithm and APC radius metric, and tests being performed in Iris, Chess, and Breast Cancer Wisconsin datasets in comparison to V-detector results.

In summary, there are few approaches inspired on the Infectious Non-self model of immune response. However, this model is supposed to be halfway between Self-Non-self-based Model and the Danger Model in terms of development of AIS approaches, as the concept of signals, implied by the pattern recognition receptors, and contextualization of anomaly represented by PAMPs or by a conserved self pattern are features that indicate anomalies in an application.

15.3.1.3. *Danger model-based algorithms*

Another proposal was the Danger Model, usually referred as Danger Theory,¹ which defines that the immune response occurs during a distress event from the body, and activation signals are sent by damaged cells. Both models also define a higher influence of innate immunity on adaptive immunity.

Once introduced as a new immune-inspired paradigm and designed for computer security applications, this immunological model would provide a second generation of immune-inspired algorithms. Mainly represented by the Dendritic Cell Algorithm, evaluated in (Greensmith and Aickelin, 2009), these algorithms were based in the correlation between system behavior (Danger and Safe signals) and potential anomalous processes (Antigens).

The algorithm was further simplified in order to work as a deterministic algorithm, with further formalization, formulations, advances, comparison to other algorithms, functionality analysis and complexity analysis, which clarifies most implementing issues for the algorithm. Mathematical aspects related to its geometrical interpretation and linearity are also discussed. A progress analysis and some suggestions about all these DCA mechanisms were discussed in (Ding *et al.*, 2013), and an updated study is seen in (Gu *et al.*, 2013). Other Danger model related works can be seen in [Table 15.2](#), which in a comparison to the Infectious Non-self model, there are more approaches inspired on this immunological model and some advances in their study.

Table 15.2: Some other Danger Model-based related works.

Approach	Ref.	Description	Application
A Danger Model-based Architecture	(Elsadig and Abdullah, 2009)	An autonomous and self-responding system with robustness, capacity of detection and prevention of threats and even to recover from damages.	Intrusion prevention system with self-Healing
A machine learning ensemble framework	(Zhu and Tan, 2011)	Application of signals, danger signals and danger zones concepts, to define interactions between classifiers and some operations such as the self-trigger process to evaluate conflicting results.	Spam filtering
DM-FD	(De Almeida <i>et al.</i> , 2010)	A Danger Model inspired method which evaluates anomalous data using an input for a fuzzy inference system based on APC signal transduction and its output being based on a mathematical model of NK cells reaction against tumors.	Fault detection
Danger Model Immune Algorithm (DMIA)	(Xu <i>et al.</i> , 2013)	An optimization algorithm which employs some danger operators applied to population, and an adaptive danger zone radius.	General optimization problems
DTAIS	(Zhang <i>et al.</i> , 2013)	An optimization algorithm which employs three different types of cell and co-evolution mechanisms.	Dynamic nonlinear constrained and single-objective problems
Danger Model-based classification	(Zhang and Yi, 2010)	A classification method based on shape-space representations.	Classification datasets

The Danger Model inspiration can provide many interpretations about the problem environment, mainly in anomaly detection problems, for which this paradigm is widely employed. The expert knowledge is one of the main forms of representation for the analogy, which does not necessarily need to represent danger contexts. Unlike most models representation, Danger Model based approaches may not need a training phase for its algorithms, but a proper representation of its signals is necessary, as stated in (Costa Silva *et al.*, 2012b), and how to obtain this representation for some problems is still a recurring challenge in literature. Some tests in malware detection in (Shafiq *et al.*, 2008) indicate that DCA detection is good, as it has a low rate of false alarms, but compared to other approaches, is far from perfect, even the real-valued negative selection has a higher detection rate according to some results.

Even in some intrusion detection problems, the Danger Model has some limitations such as antigen sampling, robust signal selection and time correlation problems, and some adaptations may be provided for the analogy, as discussed in (Vella *et al.*, 2010). Depending on the context, these algorithms may need an expert model to be employed and provide proper results.

15.3.2. Clonal Selection-Based Algorithms

The Clonal Selection Theory² is related to the expansion of lymphocytes, antibody formation in response to the antigen presence, and faster responses in further new exposures to the same antigen. This theory has inspired the Clonal Algorithm (CLONALG) applied to optimization and machine learning problems, as well as classification problems, as in (Sharma and Sharma, 2011). The algorithm has somatic hypermutation, with some insights reported in (Jansen and Zarger, 2011), and their importance to an application context in (Ulutas and Kulturel-Konak, 2013), diversity features and is able to search for local and global optima solutions. Since then, it has been widely applied to many optimization problems in the literature. A deep and comparative survey of some approaches, with the emerging research about these algorithms are discussed in (Al-Sheshtawi *et al.*, 2010; Ulutas and Kulturel-Konak, 2011).

In (McEwan and Hart, 2010), the Competitive Exclusion mechanism of clonal selection was discussed, based on generation and filtering aspects, the mathematical models involved and some biological aspects discussed, such as competition factors between clones. As clonal selection has been often seen as an “asexual” selection-based evolutionary approach since its proposal, the authors have proposed two models for the competitive exclusion mechanism: as an approximator with greedy heuristics and global optimization, and as a learning approach. Both approaches were developed in terms of dynamic systems approach rather than the evolutionary approach, and the authors have reinforced that both approaches can coexist and be integrated, and the model developed in the work should be resumed as well.

The research in (Oliveira *et al.*, 2013) adopts an alternative representation for the algorithm inspired by the Pittsburgh-style representation in Genetic-Based Machine Learning. In this representation, the main objective is to select as few instances as possible to represent the training set data without accuracy losses and number of required instances is set dynamically and evaluated during the affinity assessment. The main features of the model are the measurement of the performance of all selected prototypes during execution, representation without redundancy, faster distance calculation between prototypes. For validation purposes, several algorithms were compared to the new approach, and the novel method has shown some promising results.

Some other interesting works are reviewed in [Table 15.3](#).

Table 15.3: Some other clonal selection theory-based related works.

Approach	Ref.	Description	Application
CSA	(Cuevas <i>et al.</i> , 2012)	Development of Clonal Selection to deal with estimation of gaussian mixture models with fast convergence and a low sensitivity to initial conditions	Multithreshold segmentation problem
CSA	(Muddada <i>et al.</i> , 2010)	The application of Structural Similarity Index to solution evaluation in exploiting the solution structures and its use in threshold divergence, normalization, and proper mutation and cloning operators	General optimization problems
Several CSA approaches	(Yunfang, 2012)	Proposal of a framework to analyze four algorithms: CLONALG, MISA, NNIA, and CMOIA as its principles, mainly used operators and processing methods	Multiobjective optimization problems
NNIA	(Gong <i>et al.</i> , 2013)	Improvements on the algorithm to optimize Modified Ratio Association and Ratio Cut objectives for revealing multi-resolution community structures in complex networks	Complex networks optimization
Novel immune clonal algorithm (NICA)	(Shang <i>et al.</i> , 2012)	Proposal of an algorithm with dominance function and a selection operator according to it	Multiobjective optimization problems

CSA	(Ma <i>et al.</i> , 2009)	Proposal of an algorithm with the concept of immunodominance and three measurements, with different strategies to each population generated through each measurement	Multi-objective clustering problems
CSA	(Jabeen and Baig, 2010)	Proposal of a Genetic Programming approach with main features of clonal-based algorithms, aiming to avoid bloat and reduce training time	Classification problems
CSA	(Gan <i>et al.</i> , 2009)	Another Genetic Programming-based CSA	Fault detection
CSA	(Karakasis and Stafylopatis, 2008)	Combination of Gene Expression Programming (GEP) to Clonal Selection	Classification problems
CSA	(Tang <i>et al.</i> , 2010)	Another GEP-based CSA	Intrusion detection system
Immunoglobulin based artificial immune system (IAIS)	(Chung and Liao, 2013)	A novel clonal-based technique with the isotype switching, which combines different mutation methods	Hybrid flow shop scheduling
GT-AIS	(Jarosz and Burczyski, 2010)	A CSA with Game theory-based fitness evaluation mechanism	Multi-objective optimization problems
AIRS	(Jenhani and Elouedi, 2012)	Literature review and a new version of the algorithm with a new parameter	Classification Problems
EXPAIRS	(Golzari <i>et al.</i> , 2011)	Improvement on the algorithm by a nonlinear resource allocation employment to improve affinity metrics	Classification Problems
RL-AIS	(Karakose, 2013)	A CSA with a reinforcement learning mechanism based in Q-learning	Remote sensing data.
CSA	(Riff <i>et al.</i> , 2013)	An application of CSA for reinforcement learning with the (C, n)-strategy applied to parameter control	Combinatorial Optimization
CSA	(Hu <i>et al.</i> , 2013)	A CSA based on multi-attribute decision-making with co-evolution mechanisms	Clothing uniform assignment problem

CSA	(Ding <i>et al.</i> , 2011)	Another CSA with co-evolution applied to decision-making in optimization	Clothing uniform assignment problem
CSA	(Liu <i>et al.</i> , 2012)	An algorithm which applies B cells as bases and combines different T cells analogies to improve solutions and maintain diversity of them	Grid scheduling problems
AC-CS	(Mohammadi <i>et al.</i> , 2012)	An associative classification algorithm inspired on clonal selection and whose affinity measure is based on rules confidence	Classification datasets
AISLFS	(Dudek, 2012)	A classification algorithm local feature selection mechanism	Classification datasets
AIFSA	(Fouad <i>et al.</i> , 2011)	A clonal-based algorithm used as a wrapper and weighting algorithm	Feature selection
Motif Tracking Algorithm (MTA)	(Wilson <i>et al.</i> , 2010)	A clonal-based algorithm applied to motif detection	Time-series analysis
CSA	(Wilson <i>et al.</i> , 2010)	A clonal-based algorithm applied to optimization of SCGM1,1 forecasting	Network security

Clonal selection theory has contributed in several aspects for the development of different systems, many of them applied to optimization related problems. Immunological memory and cloning have been explored, and some operations such as hypermutation, affinity maturation and the selection itself, have been exploited, as several algorithms with different objectives have been proposed, since CLONALG until some multiobjective and well-defined algorithms, in order to solve more complex optimization, or even machine learning problems. Since the proposal of Clonal Selection-based algorithms, these approaches have been widely studied and some improvements were proposed.

The next model of algorithms, based on the Artificial Immune Network theory, shares many features with Clonal Selection-based approaches and employs more sophisticated features in addition to the ones studied in these approaches. These algorithms are also applied to a diverse set of more complex applications, in comparison to Clonal-based algorithms.

15.3.3. Immune Network-Based Algorithms

The Artificial Immune Network paradigm is inspired on the Idiotypic Network Hypothesis, in which is postulated that immune responses may be triggered through idiotypes, possibly unique to antibody and supposedly expressed on Immunoglobins (B-cells), also discovered on TCRs. These idiotypes not only recognize antigens, but also recognize paratopes from other receptors, allowing mutual interactions between immune cells and providing more effective responses. The network theory has inspired some approaches, such as an immune network for diagnosis, learning and data mining application, among other models.

Some interesting applications of immune network theory are in automated systems, such as in (Khan and de Silva, 2012), in which the immune network is applied to a self-

regulated fault tolerant multi-robot cooperation system, as a robot is modeled as an antibody and its interaction environment is modeled as an antigen. The objective of the application is to provide coordination and cooperation among robots. Other robotic-based applications of immune-inspired systems can be seen in (Raza and Fernandez, 2012).

Most Immune Network approaches are reviewed in [Table 15.4](#).

Table 15.4: Some other Immune Network theory-based related works.

Approach	Ref.	Description	Application
Immune Network Architecture	(Fernandez-Leon <i>et al.</i> , 2011)	Development of autonomous mobile robot with immune network metaphor applied to different low-level behavior coordination influenced by environmental conditions	Robotics
A decentralized Immune Network algorithm	(Świącicki, 2008)	An algorithm with levels of decentralization of the network measured according to antigens and their number during network training	Data mining
Adaptive Immune Response Network (AIRN)	(Liu <i>et al.</i> , 2009)	An algorithm with a processing feature based on matching between features, as the network has to reveal patterns in the original dataset	Clustering datasets
The Immune Feature Extracting Network (IFEN)	(Ge <i>et al.</i> , 2012)	A feature extraction algorithm with a different features such as selection, cloning and diversity maintenance	Feature extraction related
RA-AIS	(Li and He, 2013)	An Immune Network approach with proper mutation, cloning and suppression operators applied to RFID reader collision avoidance problem	Resource allocation

opt-aiNET	(Yang <i>et al.</i> , 2011)	Application of the immune-inspired approach to collaborative RFID problem	Resource allocation
aiNET	(De Fran��a <i>et al.</i> , 2010)	Review of aiNet conceptual and practical aspects	Literature review
dt-aiNET	(Zhang <i>et al.</i> , 2013)	Proposal of an aiNet with danger zone-based local search operators	General optimization problems
Predication based aiNET	(Xu <i>et al.</i> , 2010)	Proposal of an aiNet with immune predication operator and defined local and global search phases	General optimization problems
aiNET	(Xu <i>et al.</i> , 2010)	Proposal of two operators: self-organized criticality based suppression and q-gaussian mutation operator	General optimization problems
WBMOIA	(Xu <i>et al.</i> , 2010)	Proposal of an aiNET based algorithm with a weighted sum of objectives as an affinity function and truncation of similar individuals	Multiobjective optimization problems
Immune Network based	(Chen and Zang, 2011)	Proposal of a clustering algorithm in which its components can change and learn patterns. Hierarchical clustering mechanism and antibody memory cells are adopted	Clustering
AIMCA & IMCS	(Liu <i>et al.</i> , 2010)	Proposal of two aiNET based algorithms: AIMCA, with dynamical adjust of parameters, and IMCS, with evolution of parameters and a poly-clonal operator in antibodies	Combinatorial optimization problems
Immune Network based	(Liu <i>et al.</i> , n.d.)	Proposal of a single B-cell based method aiming to avoid high scale of network and suppression rates	SAR image classification
DHPAINC	(Deng <i>et al.</i> , 2013)	Proposal of an approach in which the decision hyperplane is constructed through decision antibodies and those with high quality are stable, defined by a recognition ability function	Classification and speech recognition
Autopoietic Model based	(Nanas <i>et al.</i> , 2010)	Proposal of an information filtering system inspired by the Autopoietic model of immune networks, with self organization and distributed mechanisms	Web data mining
Agent-based immune network architecture	(Hilaire <i>et al.</i> , 2010)	Immune network based model of adaptive cooperation at the local level and of emergent behavior at the global level in Multi-Agent Systems	Robo soccer experiments

Some of earlier immune network approaches have been developed as an extension for the clonal selection-based methods. However, since the analogy has been more explored throughout last years, some novel and interesting approaches have also been appeared in literature in order to provide solution of other problems, such as robotics and data mining.

Clustering and optimization are still a recurrent application of the immune network-based approaches as well. These aspects will be further discussed in the research analysis.

15.3.4. Algorithms Based on Other Models

The further approaches presented here are based on other models or features of the immune system and these approaches are focused on other analogies provided by the immune system, as presented in [Table 15.5](#). Some of them can serve as alternative approaches to different applications.

Interestingly, the cognitive paradigm of immune system (Cohen, 2000), which states that immune system recognizes both self and non-self and immune responses are mediated through cooperation of immune cells has lately inspired an information retrieval architecture in (Hu *et al.*, 2008). The proposed system relies on interactive networks between agents of the system and a co-evolutionary mechanism led by an affinity maturation through the networks. Studies about development of systems inspired on this paradigm have started in (Andrews and Timmis, 2007; Voigt *et al.*, 2007), but since then, few works have explored this paradigm of immune system, which can provide interesting analogies.

According to Hart *et al.* (2009), modeling collaborative network among immune entities can lead to the development of novel approaches fit for their purpose of solving engineering problems. Analogies are taken from innate immunity and their interaction to the adaptive immunity elements, modern models of immune networks, and the cognitive paradigm. These analogies were discussed and then related to some real-world applications.

Table 15.5: AIS related works based on other aspects.

Approach	Ref.	Description	Application
T-Cell receptor density	(Owens <i>et al.</i> , 2013)	Algorithm based in a mathematical model of TCR density and feedback estimation, with a comparative to some kernel density estimation-based techniques	Anomaly detection
AIDEN	(Pathak <i>et al.</i> , 2012)	A method that adopts a density concept and considers TCRs stimulation and interaction to B cells in the form of Antigen Recognition Balls	Clustering datasets
Complement System based	(Aitken <i>et al.</i> , 2008)	Proposal of a method based on one of pathways of the complement system, based on message passing processes. Requires a matching function	Engineering problems
NK cells based	(Laurentys <i>et al.</i> , 2011)	Proposal of an algorithm inspired by Natural Killer Cells functions: Receptor balance and the education processes	Fault detection
NK cells based	(Fu <i>et al.</i> , 2012)	An algorithm inspired by Natural Killer Cells with activating and inhibition cytokines signals applied to spyware detection	Malware detection
T cells based	(Aragon <i>et al.</i> , 2011)	An algorithm inspired by T Cells with three groups and idynamic tolerance factor and population change evaluation mechanisms	Dynamic optimization problems
Dynamic Effector Regulatory Algorithm	(Guzella <i>et al.</i> , 2007)	A method that incorporates cytokines and regulatory cells and adopts different concepts of Self and Non-self spaces, based on effector and regulatory cells distribution and recognition of normal and abnormal processes	Fault detection
Innate immunity based	(Guzella <i>et al.</i> , 2007)	Discussion of modeling aspects of innate immunity based algorithms in terms of multi-level data fusion mechanisms and how real-world problems should reflect on these algorithms	Intrusion detection systems

Humoral Artificial Immune System	(Narayanan and Ahmad, 2012)	Novel technique with diverse concepts exploited for supervised learning using mechanisms as similarity measures, affinity thresholds, maturation, hyper-mutation, and memory cells, among others	Machine Learning
Artificial Immune Privilege Site Model	(Singh <i>et al.</i> , 2011)	Method based on antigen tolerance without inflammatory responses	Robotics and Computer security
Memory Cells based	(Suarez-Tangil <i>et al.</i> , 2011)	Learning and memory analogies are explored to provide the generation of event correlation rules as a novel multi-step attacks detector	Computer security
Immune models	(Suarez-Tangil <i>et al.</i> , 2011)	Proposal of comparison regarding two types of simulation: Simulation of Dynamic System (SDS), a mathematics-based immune model, and Simulation Based in Agents (ABS), based on models	Approach evaluation
Cross-regulation based	(Abi-Haidar and Rocha, 2011)	An agent-based method inspired by the dynamics of a population of T-cells and APCs for antigen recognition	Data classification and feature selection
Tunable activation threshold (TAT-AIS)	(Antunes and Correia, 2010)	A novel algorithm inspired on two concepts: homeostasis and clonal size regulation	Temporal anomaly detection
TAT-AIS	(Antunes and Correia, 2010)	Employment of TAT-AIS in another application	Network Intrusion detection systems

15.4. A Brief Summary about Hybrid AIS Approaches

AIS can also be combined to other approaches as described in (De Castro and Timmis, 2002), with description of diverse combinations of paradigms, and in (Dasgupta and Niño, 2008), with the example of NS-SOM model described. The development of hybrid systems has increased in the literature, not only for AIS, but also for many other nature-inspired algorithms. This survey will further describe the influence of hybrid approaches in AIS research.

Hybrid system can be divided in two groups. In the first group, the use of tools or methods that can extend or enhance AIS to provide proper features in problem solving will be discussed, such as Probability Theory, Fuzzy Logic, Information-based Tools, among others. In the other group, systems that employ different algorithms or techniques with their functions mixed will be discussed, such as Artificial Neural Networks, Evolutionary Algorithms, Swarm Intelligence Algorithms, other machine learning methods, among others.

15.4.1. Useful Tools for AIS Enhancement

Several tools can be employed in order to extend AIS features or to enhance most aspects, such as probability or bayesian tools, fuzzy set theory, information theory, kernel methods, and other tools since they are not considered as specific algorithms. Hybrid approaches involving multiple algorithms will be further discussed in the other group.

Some of these tools have been already adopted in early approaches, as in the example of the first Self-Non-self-based system in (Forrest *et al.*, 1994), whose detection is probabilistic. Several methods rely on these tools for modeling purposes, according to features that should be provided for a particular problem. In addition, algorithms based on clonal selection need these features because of their stochastic nature as these are applied to generation of antibodies and in the somatic hypermutation operators. However, since these features were not the focuses of their respective approaches, but features incorporated to these systems, they cannot be considered as hybrid approaches at all. As each tool can enhance AIS functionalities, a further discussion will be presented of some of these tools.

In this group, some tools can be used to provide enhancement for AIS, in terms of functions and mechanisms. The following list includes, but is not limited to these considered approaches:

- Probabilistic methods, such as gaussian or bayesian tools;
- Fuzzy and Rough sets theory;
- Information theory tools;
- Kernel Functions;
- Other learning and memory mechanisms;

- Chaos theory, quantum computing and other methods.

These tools can be used to improve or replace mechanisms on AIS approaches, in the case of probabilistic methods. Examples of these techniques are the family of Probabilistic Artificial Immune Systems based on Negative Selection in (Mohammadi *et al.*, 2012) and both Bayesian Artificial Immune System (BAIS) (Castro and Von Zuben, 2010) and Gaussian Artificial Immune System (GAIS) (Castro and Von Zuben, 2008) applied to optimization, with these algorithms also having multiobjective versions.

Fuzzy Set Theory, which can deal with uncertain or imprecise information, may offer proper models of aspects and mechanisms of the immune systems, providing powerful interactions, according to De Castro and Timmis (2002). These aspects are considered mainly for adaptive immunity features, as the antigen recognition is approximate. However, the use of Fuzzy Logic is not limited to these aspects, as the immune system has several components. Some examples of algorithms are an Immune Network based in (Szabo *et al.*, 2012), a Dendritic Cell-based in (Chelly *et al.*, 2012), and the fuzzy recognition algorithms proposed in (Costa Silva *et al.*, 2012a).

Information theory tools have been used to enhance AIS functions through entropic divergence metrics, such as Kullback–Leibler, Rényi and von Neumann, or the Dempster–belief Theory, which performs classification by computing evidences. One example of these approaches are the Local Concentration (LC), proposed in (Zhu and Tan, 2011), to perform feature extraction and is based on tendency calculation via probability of occurrences, the vector of LCs is constructed by sliding window measurements of both censored and legitimate genes. The LC model can be constructed based on term selection that can be employed by information-based methods.

The use of kernel functions was less considered, and the discussion in (Guzella *et al.*, 2008) has presented how kernel functions should extend AIS functions and features in the same way as used in machine learning paradigms such as SVM algorithms. The possibility of replacement of affinity functions by kernel functions were considered to map feature space. A test using aiNet approach mapped in a Gaussian space was performed in the research. In (Ozsen *et al.*, 2009), a kernel-based AIS for classification inspired on clonal selection was proposed, and the affinity function between Ag–Ab is replaced by a kernel-based distance measurement. Benchmarks of UCI Database were applied to the proposed method.

These are some examples of useful tools in the development of different techniques or the improvement of existing ones without alter significantly the main idea of either the abstract model of an algorithm or functions for which these algorithms were designed for. Instead, these tools should provide a better suitability of the model or even permit the feasibility of an analogy for the development of new techniques applied to problem solving. Approaches that are developed mixing multiple techniques or employing features of different algorithms or even paradigms will be further discussed in the next subsection.

15.4.2. Hybridization of AIS and Other Paradigms

In this subsection, hybrid approaches that use mechanisms from AIS and other algorithms or even multiple algorithms will be discussed. Differently from the other group, in which paradigms or tools are implemented in terms of AIS modeling, in this group, all approaches employ different algorithms in the same system, in a high level of system hybridization. The list of approaches that can be used for this purpose include, but are not limited to the following:

- Nature Inspired approaches:
 - Artificial Neural Networks.
 - Genetic Algorithms and other evolutionary algorithms.
 - Swarm intelligence approaches.
- Other Machine Learning systems:
 - Naive Bayes.
 - k-Nearest Neighbors (kNN).
 - Support Vector Machines (SVMs).
 - Most Clustering algorithms.
- Fuzzy systems:
 - Fuzzy C-means.
 - Takagi-Sugeno systems.
- Meta-heuristics.
- Meta-learning tools.

Examples of applications for possible hybrid AIS are mining rules from neural, weight optimization, ensemble in classification problems and generation of rule-bases. Some examples of these approaches can be seen in [Table 15.6](#). The ones based in Swarm Intelligence have some interesting aspects and a further discussion is presented.

Swarm Intelligence-based approaches is another nature-inspired paradigm that have been widely applied to optimization problems and then, comparisons to AIS approaches have been discussed in (Timmis *et al.*, 2010), since both paradigms have many aspects of direct parallels, since Swarm Intelligence exploits the result of individual behaviors in coordinated population behavior as a whole and AIS exploits immune functions and models. Both approaches have been also discussed in terms of self-organization, positive and negative feedbacks, amplification factors and multiple signals of phenomena. The scientific and abstract aspects of swarm interactions have been discussed and both approaches can be considered as complementary fields of research that also can be

combined to develop new techniques. Similarities between both approaches have been discussed in (Xiao *et al.*, 2011), such as structure of individuals, their interaction, and system structures, as well as their learning, memory, feedback, and adaptability aspects, among other points. It is possible that Swarm Intelligence-based approaches, as well as AIS, are emerging approaches whose potential is still high for analogy exploiting.

Table 15.6: Some Hybrid AIS related works.

Approach	Ref.	Hybrid with	Description	Application
RABNET	(Masutti and Castro, 208)	Self-organizing Neural Network	Antibody based system with cloning and pruning mechanisms applied to real-valued problems	CVRP problem
Negative Selection	(Davis <i>et al.</i> , 2010)	Genetic Algorithm	NSA enhanced by GA optimization of detector generation	Anomaly detection
T-Cell receptor density algorithm (RDA)	(Hilder <i>et al.</i> , 2011)	Genetic Algorithm	Parameters of RDA are enhanced by genetic operators	Anomaly detection
Imaboost	(Taud <i>et al.</i> , 2010)	Several classifiers	Transformation of Adaboost with clonal selection operators into a novel technique	Machine Learning Ensemble
CSA	(Aydin <i>et al.</i> , 2011)	SVMs	Optimization of SVM parameters	Fault diagnosis
CSA	(Bernardino <i>et al.</i> , 2010)	k-NN algorithm	Development of a Similarity-based Surrogate Model to clonal selection	Optimization
NSA and CSA	(Chao and Tan, 2009)	Machine Learning approaches	Training of classifiers using the immune approaches	Computer Security

Main examples of these approaches are combinations of AIS and PSO, as PSO may be applied to antibody improvements under mutation operators in classifiers or clustering methods. Several approaches were discussed in (Wu, 2012), including a novel immune-inspired method, presented to solve constrained global optimization problems.

There are many possibilities of hybrid approaches, since there are many paradigms to be considered. How these algorithms can be applied to a given problem is also another aspect of these mixed systems, since these algorithms, as well as AIS approaches, can serve as ensemble for one or many algorithms, or improve features or results of a given algorithm, or even be a part of the algorithm processing. The approaches cited here are some examples of how AIS can improve or be improved by other techniques to solve harder problems, as for their complexity, a single algorithm may not be enough.

Some of these combinations, however, may imply on redundant features, since AIS have features in common with other paradigms of machine learning or mainly nature-inspired applications, which are subject to similarities between paradigms, as in the example of Clonal Selection algorithms and Evolutionary Algorithms approaches. Some

other examples will be further discussed in the sense of AIS research.

15.5. Impacts in Biological Research

Immunology serving as a source of inspiration for computational systems may provide some multidisciplinary links and many interactions among researchers from different fields of study and a perspective of benefits for immunology, computer science and engineering respective researches, as further discussed in terms of AIS impact in biology and how computation could inspire biological models in research.

Reviews done in (Timmis *et al.*, 2008) point that the development of AIS algorithms may provide interdisciplinary links between immunology, computer science, mathematics, and engineering, since the immunology provides metaphors for the creation of novel solutions to problems. The authors have provided a framework for modeling novel approaches from immunology through a methodology evaluation, some reference models and the further development of techniques from the abstract model, as well as some features to be provided in newer AIS approaches. It is also suggested that these models may help immunologists in bringing some understanding of the immune system as well as proper representations in AIS for engineering systems through greater interaction between each group of researchers.

As immunology may inspire computer systems, computation can also be an inspiration for insights about immunology, as defined by Cohen (2007), in this work are described the view of the immune system as a computational entity and immunological concepts and functions in terms of computation. Some computation concepts were applied to immunological components, such as the system states, its cells and their actions in the immune system. The paper also reinforces the need of interactions between immunologists and computer scientists.

The research in (Navlakha and Bar-Joseph, 2011) was related to biological modeling and inspiration for strategies to problem solving as well as the similarities between computation and biology and their differences. According to the authors, computational systems are often focused on speed and biological systems are focused in dynamic adaptability to changes and their decision-making mechanisms. This suggests that biological inspirations are very useful to computational systems. Features of biological and analogous computational systems are discussed and their research was considered as a *computational thinking of biological systems*, which provides more understanding of biological processes as well as improvements in algorithms development.

The ecology of immune system is discussed in (Tauber, 2008), in this paper, the author also discuss the ‘Immunocomputing’ research described as immunology formalization aspects and its quality as a fruitful source for applications to various problems. It is also implied that AIS is involved in multidisciplinary studies including fields of immunology modeling, mathematical modeling and computer simulations, among other forms of simulating immune models.

The artificial immune systems research, as seen on these works, may provide some significant contributions to immunology in the sense of understanding the immune system, as well as providing a multidisciplinary research between immunologists and computer scientists or engineers in order to establish interactions that should improve the understanding of the real functions of immune systems and reinforce the development of novel techniques and provide better results in problem solving.

At the present moment, there are few works pointing this aspect of AIS. However, there are some definitions stated in (Hart *et al.*, 2009), such as ‘Immunoengineering’, which is inspired on immunoecology (study of principles applied to immunological functions) and immunoinformatics (related to decision-making paradigms), whose elements are reunited to provide adaption and applications. These studies would provide a framework that supports the development of mathematical and computational models, as well as their validation through benchmark problems. In summary, understanding AIS may reinforce the understanding of immunological principles and models, as well as provide the growth of the research on bio-inspired systems.

15.6. Research and Applications

In this section, a summary about AIS applications and focuses is presented in the sense of how AIS research is being currently applied, as most aspects of current state of the research about AIS have been discussed on the survey in (Dasgupta *et al.*, 2011). The main remarks discussed are the popularity of techniques, mainly negative selection and clonal selection algorithms, and the advances in the research, such as interdisciplinary aspects, emerging research about other immunological models, AIS textbooks and their highlights and conferences about AIS or which provides the spreading of current state of AIS related works.

Immune-inspired methods are still appearing in most computational intelligence or evolutionary computation conferences. However, during this year, the International Conferences on Artificial Immune Systems (ICARIS) has become a part of Advances in Artificial Life, ECAL 2013 Conference, due to a drop in submissions of the conference that was supposed to happen during the current year.

Some AIS related up-to-date textbooks have been published. Regarding the current state-of-the-art, in (Dasgupta and Niño, 2008) is described the overview of most AIS techniques and models that have been developed, as well as their related analogies and the applications for which they were applied. An optimization focused book in (Mo, 2008), presents some AIS and other nature-inspired methods and improvements and discusses them. Since then, most AIS related books recently published are related to applications, such as computer security problems in (Al-Anezi and Al-Dabagh, 2012) applied to intrusion detection and malware, or in (Unterleitner, 2012) also applied to virus detection.

In these years, some reviews were done, as in (Ramakrishnan and Srinivasan, 2009) for AIS applied to computer security systems and using agent-based approaches to solve the problem, Network Intrusion Detection-based approaches are reviewed in (Chauhan *et al.*, 2013, Singh *et al.*, 2013) in terms of developing techniques, in (Wong and Lau, 2009) a review about optimization methods was done with description of design, benchmark and applications to real life problems, in (Malhotra *et al.*, 2012) a survey on the applications of AIS in various fields related to pattern recognition is presented, in (Raza and Fernandez, 2012), robotic-based applications of AIS are presented, many of them inspired on the immune network theory, and in (Dalal and Singh, 2012), a review of AIS-based applications to Mobile Ad-hoc Networks was presented.

A graphical illustration regarding publications per year within last five years is shown in [Figure 15.1](#).

A list of recent applications is presented in a quantitative illustration of published papers in [Figure 15.2](#).

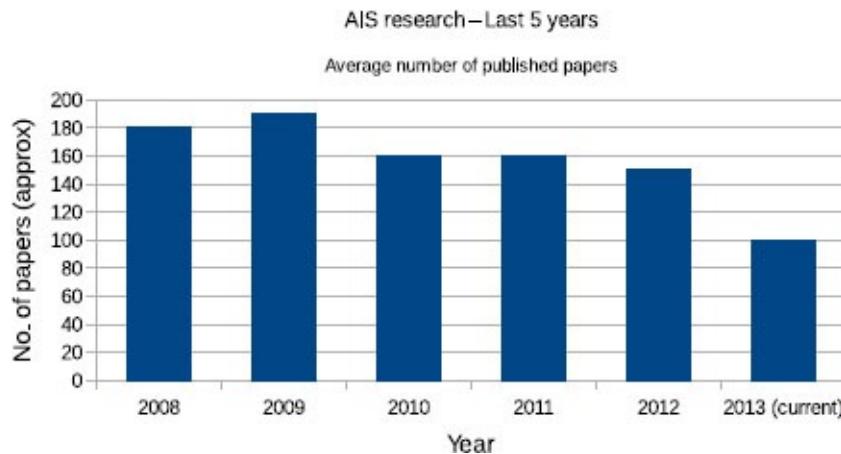


Figure 15.1: Papers published per year.

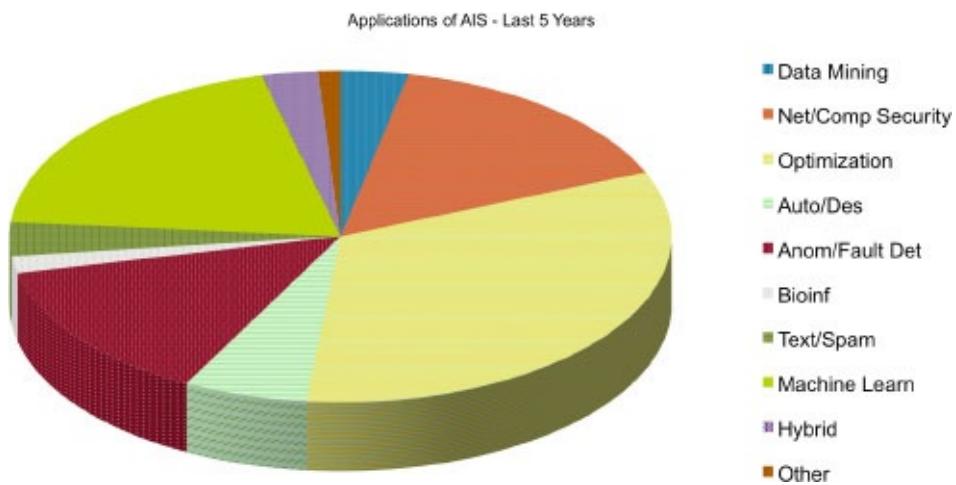


Figure 15.2: Distribution of AIS applications.

These data imply that AIS research has emerged and then is currently updated with more publications, mainly in optimization, in which clonal selection based approaches are constantly applied. Applications to security and anomaly detection are also updated with more approaches and improvements for negative selection-based algorithms. The other immune response models are currently emerging, mainly the dendritic cell algorithm, which is being applied for computer or network security as well. Immune networks have also some remarkable applications, such as clustering, robotics, and some optimization problems.

The Figure 15.3 illustrates these publications distributed per immune model or algorithm developed. These data are considered by their features, as Negative Selection approaches are based on Self-Non-self Discrimination, Clonal Selection are based on antibody production, cloning and mutation features, Immune Networks are based on network models, and PRR and Danger approaches uses features of the other immune response models. If these approaches have different features than the previous models, they are included in Other category and if more than one model or approach are developed, they are included in Multi category, which means that more than one AIS approach was adopted.

From previous data, the Figure 15.4 is regarding the research of these algorithms during these years. According to these data, Clonal Selection-based approaches have been

consolidated, because of its application to optimization problems. Negative Selection and Immune Networks based approaches are also popular, due to their applicability and features provided. The other immune response models are emerging approaches that had many researches, but they are still in development.

[Figure 15.4](#) presents the relationship between AIS models and their applications through number of papers published in these last five years, for Computer and Network Security, Anomaly Detection, Optimization, Machine Learning and the other applications (including Data Mining and Text for example). As expected, clonal selection and optimization are typically related. There are other relations, such as negative selection and anomaly detection, or danger model based approaches and applications for computer and network security, as some examples.

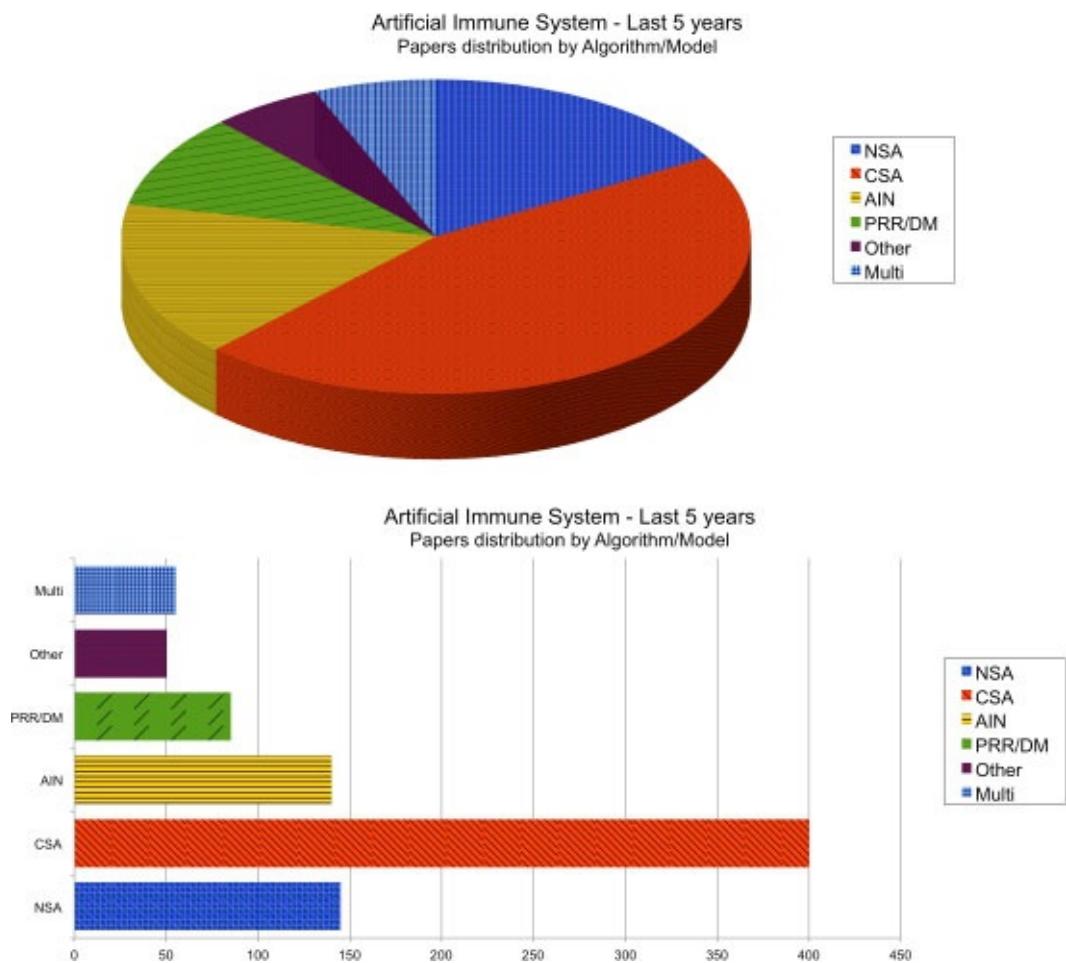


Figure 15.3: Quantitative distribution of papers published of AIS models.

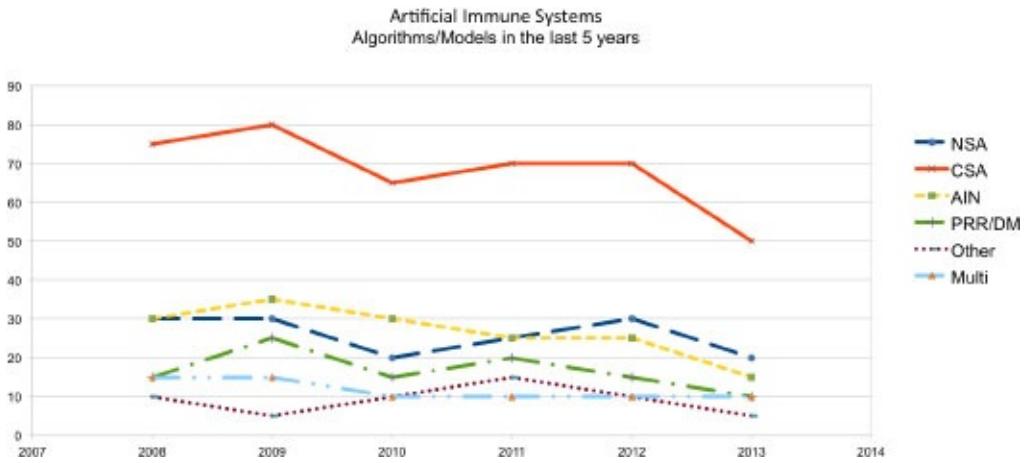


Figure 15.4: Quantitative distribution of papers published of AIS models per year.

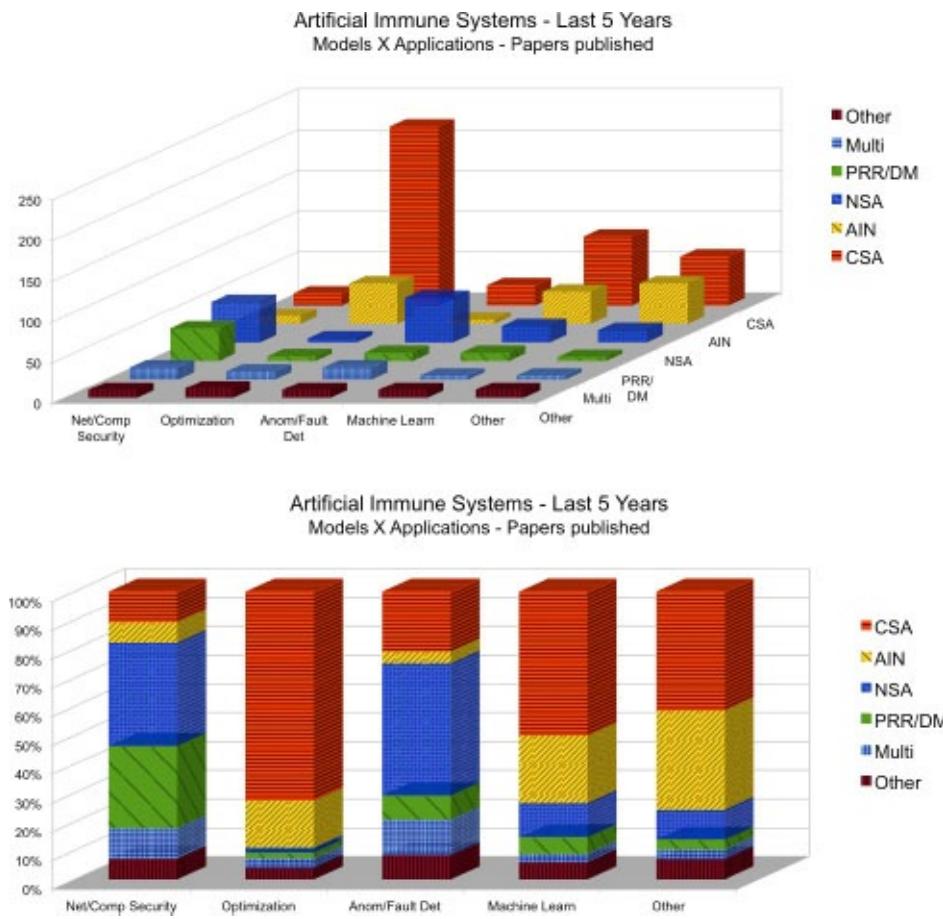


Figure 15.5: Distribution of papers published of AIS models per applications.

In [Figure 15.6](#), paper distribution is categorized by classical or hybrid approaches, illustrating the proportion of these approaches in these last five years and the quantitative analysis throughout years.

The [Figure 15.7](#) considers the development of classical and hybrid approaches by the based model and their proportional distribution. As presented by these graphics, hybridized versions of Clonal and Immune Networks algorithms are the most popular in AIS research.

The [Figure 15.8](#) considers the situation of classical and hybrid AIS approaches regarding to their application. Except for Anomaly Detection, in which hybrid approaches has more papers published, classical approaches are more dominant for most applications.

For hybrid approaches, it was also considered the employment of AIS as an improvement of these systems, which is becoming a recurring approach.

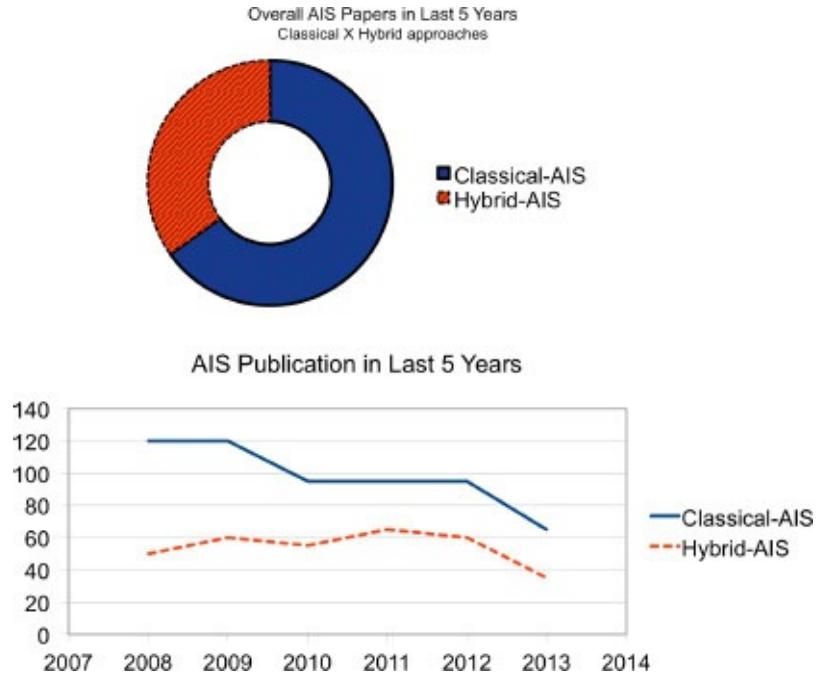


Figure 15.6: Classical X Hybrid approaches in these last five years.

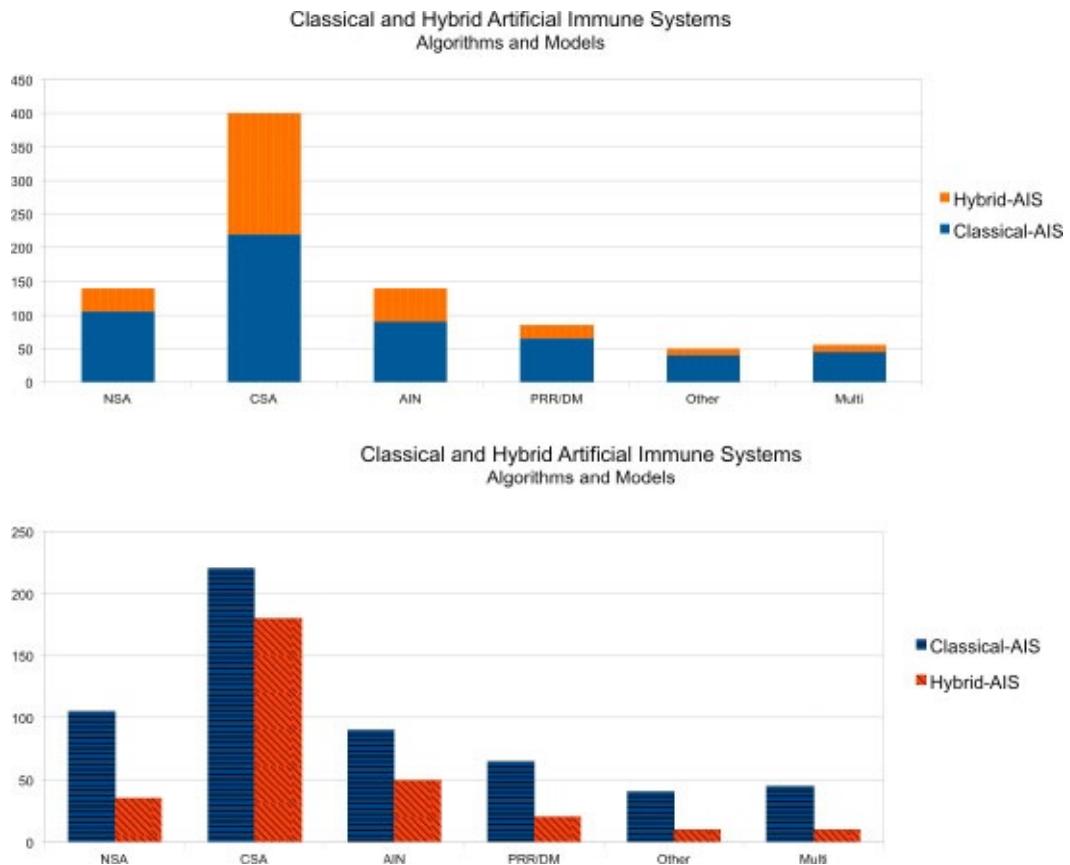


Figure 15.7: Classical X Hybrid approaches regarding the model studied.

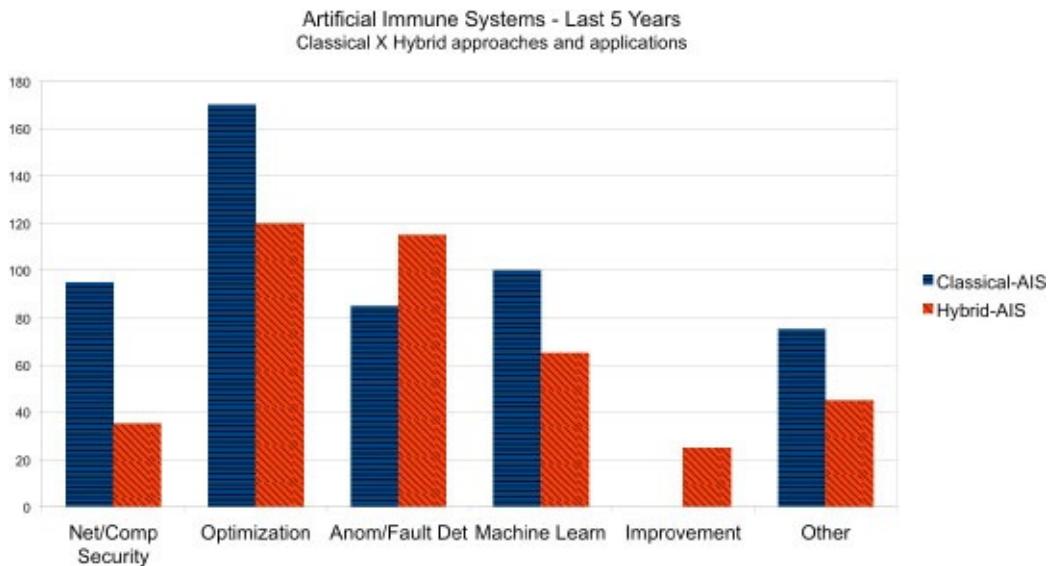


Figure 15.8: Classical X Hybrid approaches regarding the application. For hybrid approaches, AIS as improvements are also listed.

In a global view of this analysis, Artificial Immune Systems can be considered as a well developed and consolidated field of research as many publications are presented, considering the several applications of these techniques to be exploited.

15.7. Remarks

This work has presented the current state-of-the-art of the research field and some aspects such as current state of approaches, their applications, some contributions to multidisciplinary researches and the distribution of research according to many aspects within these last five years.

AIS has been consolidated as a nature-inspired paradigm that employs metaphors by reproducing models and mechanisms from immunology in order to solve problems. Several works have been published and some models have appeared throughout these last years.

State-of-the-art models, such as Negative Selection and Clonal Selection have been successfully employed in their proper applications and are currently consolidated as successful immune-inspired algorithms. Immune network-based applications are also well applied to some problems in literature, and the other immune response models, which have inspired some novel algorithms, such as CSPRA and DCA, are emerging as their potential are still to be properly explored for their applications.

In other part, the development and study of hybrid approaches in which AIS methods are being implemented is increasing, as different techniques or their combination may reproduce different and improved results. In this view, AIS are being improved by some methods, as well as serving as improvement for other systems. Possibly, each mechanism may be isolated for the better exploitation, as some researches are focused on specific algorithms, in order to provide better problem solving through different combinations.

Actually, as the biological immune system has many features to be exploited, there are many possibilities of approaches, considering mechanisms, models and even biological mechanisms involved on immune processes. The current state of the art has exploited few of these characteristics, most of them in a high level of abstraction and focusing features. Newer approaches are exploiting more biological factors and providing mathematical or computational models for these mechanisms, as they are being studied. Biological analogies may not be perfect, but they have to be suitable for development of proper algorithms for a given problem.

However, the success of Artificial Immune Systems may be explained by the results provided by these analogies and the intensive research regarding these algorithms and how to improve them in terms of technique or by providing better results. Seemingly, not only AIS, but most nature-inspired methods have several features to be exploited, serving as analogies in development of new systems.

Acknowledgments

The authors acknowledge the support of Brazilian funding agency CAPES, which supported the first author during this study.

References

- Abi-Haidar, A. and Rocha, L. M. (2011). Collective classification of textual documents by guided self-organization in t-cell cross-regulation dynamics. *Evolutionary Intelligence*, 4(2), pp. 69–80.
- Aitken, J. M., Clarke, T. and Timmis, J. I. (2008). The pathways of complement. In Bentley, P. J., Lee D. and Jung, S. (eds.), *Artificial Immune Systems, Lecture Notes in Computer Science*. Berlin Heidelberg: Springer, pp. 364–375.
- Al-Anezi, M. M. K. and Al-Dabagh, N. B. I. (2012). *Multilayer Artificial Immune Systems for Intrusion & Malware Detection: Concepts, Methods, Design and Implementation*. Germany: LAP Lambert Academic Publishing.
- Al-Sheshtawi, K. A., Abdul-Kader, H. M. and Ismail, N. A. (2010). Artificial immune clonal selection algorithms: A comparative study of clonalg, opt-ia, and bca with numerical optimization problems. *Int. J. Computer Science and Network Security (IJCSNS)*, 10(4), pp. 24–30.
- Andrews, P. and Timmis, J. (2007). Alternative inspiration for artificial immune systems: Exploiting cohen's cognitive immune model. In Flower, D. and Timmis, J. (eds.), *In Silico Immunology*. US: Springer, pp. 119–137.
- Antunes, M. and Correia, M. E. (2011). Tunable immune detectors for behaviour-based network intrusion detection. In Lio, P., Nicosia, G. and Stibor, T. (eds.), *Artificial Immune Systems, Lecture Notes in Computer Science*. Berlin Heidelberg: Springer, pp. 334–347.
- Antunes, M. J. and Correia, M. E. (2010). Temporal anomaly detection: An artificial immune approach based on t cell activation, clonal size regulation and homeostasis. In Arabnia, H. R. (ed.), *Advances in Computational Biology, Advances in Experimental Medicine and Biology*. New York: Springer, pp. 291–298.
- Aragón, V. S., Esquivel, S. C and Coello, C. A. C. (2011). A t-cell algorithm for solving dynamic optimization problems. *Inform. Sci.*, 181(17), pp. 3614–3637.
- Aydin, I., Karakose, M. and Akin, E. (2011). A multi-objective artificial immune algorithm for parameter optimization in support vector machine. *Appl. Soft Comput.*, 11(1), pp. 120–129.
- Bao, Y., Luo, W. and Zhang, X. (2013). Estimating positive surveys from negative surveys. *Stat. Probabil. Lett.*, 83(2), pp. 551–558.
- Bernardino, H., Barbosa, H. and Fonseca, L. (2010). A faster clonal selection algorithm for expensive optimization problems. In Hart, E., McEwan, C., Timmis, J. and Hone, A. (eds.), *Artificial Immune Systems, Lecture Notes in Computer Science*. Berlin Heidelberg: Springer, pp. 130–143.
- Castro, P. A. D. and Von Zuben, F. (2010). A gaussian artificial immune system for multi-objective optimization in continuous domains. *Hybrid Intelligent Systems (HIS), 2010 10th Int. Conf.*, pp. 159–164.
- Castro, P. A. D. and Von Zuben, F. J. (2008). MOBAIS: A Bayesian artificial immune system for multi-objective optimization. In Bentley, P. J., Lee D. and Jung, S. (eds.), *ICARIS, Lecture Notes in Computer Science*. Berlin Heidelberg: Springer, pp. 48–59.
- Chao, R. and Tan, Y. (2009). A virus detection system based on artificial immune system. *Computational Intelligence and Security, 2009. CIS '09. Int. Conf.*, Vol. 1, pp. 6–10.
- Chauhan, P., Singh, N. and Chandra, N. (2013). Article: A review on intrusion detection system based on artificial immune system. *Int. J. Comput. Appli.*, 63(20), pp. 36–40.
- Chelly, Z., Smiti, A. and Elouedi, Z. (2012). Coid-fdcm: The fuzzy maintained dendritic cell classification method. In Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L. and Zurada, J. (eds.), *Artificial Intelligence and Soft Computing, Lecture Notes in Computer Science*. Berlin Heidelberg: Springer, pp. 233–241.
- Chen, B. and Zang, C. (2011). Emergent damage pattern recognition using immune network theory. *Smart Structures and System*, 8(1), pp. 69–92.
- Chen, G., Zhang, L. and Bao, J. (2013). An improved negative selection algorithm and its application in the fault diagnosis of vibrating screen by wireless sensor networks. *J. Comput. Theor. Nanos.*, 10(10), pp. 2418–2426.
- Chen, J., Zhang, Q. and Fang, Z. (2012). Improve the adaptive capability of tma-or. In Omatsu, S., De Paz Santana, J. F., González, S. R., Molina, J. M., Bernardos, A. M. and Rodríguez, J. M. C. (eds.), *Distributed Computing and Artificial Intelligence, Advances in Intelligent and Soft Computing*. Berlin Heidelberg: Springer, pp. 665–671.
- Chen, L.-F. (2013). An improved negative selection approach for anomaly detection: with applications in medical

- diagnosis and quality inspection. *Neural Comput. Appl.*, 22(5), pp. 901–910.
- Chen, S. (2013). Optimized multilevel immune learning algorithm in abnormal detection. *Information Technology Journal*, 12(3), pp. 514–517.
- Chung, T.-P. and Liao, C.-J. (2013). An immunoglobulin-based artificial immune system for solving the hybrid flow shop problem. *Appl. Soft Comput.*, 13(8), pp. 3729–3736.
- Cohen, I. (2000). *Tending Adam's Garden: Evolving the Cognitive Immune Self*. Academic Press.
- Cohen, I. R. (2007). Real and artificial immune systems: computing the state of the body. *Nat. Rev. Immunol.*, 07(07), pp. 569–574.
- Costa Silva, G., Palhares, R. M. and Caminhas, W. M. (2012a). Immune inspired fault detection and diagnosis: A fuzzy-based approach of the negative selection algorithm and participatory clustering. *Expert Syst. Appl.*, 39(16), pp. 12474–12486.
- Costa Silva, G., Palhares, R. M. and Caminhas, W. M. (2012b). A transitional view of immune inspired techniques for anomaly detection. In Yin, H., Costa J. A. F. and Barreto, G. (eds.), *Intelligent Data Engineering and Automated Learning—IDEAL 2012, Lecture Notes in Computer Science*. Berlin Heidelberg: Springer, pp. 568–577.
- Cuevas, E., Osuna-Enciso, V., Zaldivar, D., Pérez-Cisneros, M. and Sossa, H. (2012). Multithreshold segmentation based on artificial immune systems. *Math. Prob. Eng.*, pp. 1–20.
- Dalal, T. and Singh, G. (2012). Review of artificial immune system to enhance security in mobile ad-hoc networks. *Int. J. Computer Science & Management Studies (IJCSMS)*, 12(02), pp. 82–85.
- Dasgupta, D. (1998). *Artificial Immune Systems and Their Applications*, 1st edn. New York: Springer-Verlag.
- Dasgupta, D. and Attoh-Okine, N. (1997). Immunity-based systems: A survey. *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation, 1997 IEEE Int. Conf.*, Vol. 1, pp. 369–374.
- Dasgupta, D. and Azeem, R. (2008). An investigation of negative authentication systems. *Proc. 3rd Int. Conf. Information Warfare and Security, CSIIRW '09*.
- Dasgupta, D. and Niño, L. (2008). *Immunological Computation: Theory and Applications*, 1st edn. Boston, MA, USA: Auerbach Publications.
- Dasgupta, D. and Saha, S. (2010). Password security through negative filtering. *2010 International Conference on Emerging Security Technologies (EST)*, pp. 83–89.
- Dasgupta, D., Yu, S. and Niño, F. (2011). Recent advances in artificial immune systems: Models and applications. *Appl. Soft Comput.*, 11(2), pp. 1574–1587.
- Davis, J., Perhinschi, M. G. and Moncayo, H. (2010). Evolutionary algorithm for artificial-immune-system-based failure-detector generation and optimization. *J. Guid. Control Dynam.*, 33(2), pp. 305–320.
- de Almeida, C. A. L., Palhares, R. M. and Caminhas, W. M. (2010). Design of an artificial immune system based on danger model for fault detection. *Expert Syst. Appl.*, 37, pp. 5145–5152.
- de Almeida, C. A. L., Ronacher, G., Palhares, R. M. and Caminhas, W. M. (2010). Design of an artificial immune system for fault detection: A negative selection approach. *Expert Syst. Appl.*, 37(7), pp. 5507–5513.
- De Castro, L. N. and Timmis, J. (2002). *Artificial Immune Systems: A New Computational Intelligence Approach*. Berlin: Springer-Verlag.
- de França, F. O., Coelho, G. P., Castro, P. A. D. and von Zuben F. J. (2010). Conceptual and practical aspects of the ainet family of algorithms. *Int. J. Natural Computing Research (IJNCR)*, 1, pp. 1–35.
- Deng, Z.-L., Tan, G.-Z., He, P. and Ye, J.-X. (2013). A decision hyper plane heuristic based artificial immune network classification algorithm. *J. Cent. South Univ.*, 20(7), pp. 1852–1860.
- Ding, L., Yu, F. and Yang, Z. (2013). Survey of dca for abnormal detection. *J. Softw.*, 8(8), pp. 2087–2094.
- Ding, Y.-S., Hu, Z.-H. and Zhang, W.-B. (2011). Multi-criteria decision making approach based on immune co-evolutionary algorithm with application to garment matching problem. *Expert Syst. Appl.*, 38(8), pp. 10377–10383.
- Dudek, G. (2012). An artificial immune system for classification with local feature selection. *IEEE Trans. Evolutionary Computation*, 16(6), pp. 847–860.

- Elsadig, M. and Abdullah, A. (2009). Biological inspired intrusion prevention and self healing system for network security based on danger theory. *Int. J. Video & Image Processing and Network Security (IJVIPNS-IJENS)* 09, pp. 16–28.
- Fernandez-Leon, J. A., Acosta, G. G. and Mayosky, M. A. (2011). From network-to-antibody robustness in a bio-inspired immune system. *Biosystems*, 104(3), pp. 109–117.
- Figueredo, G. P., Aickelin, U. and Siebers, P.-O. (2011). Systems dynamics or agent-based modelling for immune simulation? *Proc. 10th Int. Conf. Artificial Immune Systems*, Berlin, Heidelberg, ICARIS'11, Springer-Verlag, pp. 81–94.
- Forrest, S., Perelson, A. S., Allen, L. and Cherukuri, R. (1994). Self-non-self discrimination in a computer. *Proc. 1994 IEEE Sympo. Research in Security and Privacy*, p. 202.
- Fouad, W., Badr, A. and Farag, I. (2011). Aifsa: A new approach for feature selection and weighting. In Abd Manaf, A., Zeki, A., Zamani, M., Chuprat, S. and El-Qawasmeh, E. (eds.), *Informatics Engineering and Information Science, Communications in Computer and Information Science*. Berlin Heidelberg: Springer, pp. 596–609.
- Fu, J., Yang, H., Liang, Y. and Tan, C. (2012). Bait a trap: Introducing natural killer cells to artificial immune system for spyware detection. *Proc. 11th Int. Conf. Artificial Immune Systems*, Berlin, Heidelberg, ICARIS'12, Springer-Verlag, pp. 125–138.
- Gan, Z., Zhao, M.-B. and Chow, T. W. (2009). Induction machine fault detection using clone selection programming. *Expert Syst. Appl.*, 36(4), pp. 8000–8012.
- Gao, J., Fang, L. and Wang, J. (2010). A weight-based multiobjective immune algorithm: Wbmoia. *Eng. Optimiz.*, 42(8), pp. 719–745.
- Ge, H., Yan, X., Okada, K. and Li, Y. (2012). Alpha matting using immune feature extraction network. In Wang, F., Lei, J., Lau, R. and Zhang, J. (eds.), *Multimedia and Signal Processing, Communications in Computer and Information Science*. Berlin Heidelberg: Springer, pp. 207–214.
- Ghosh, K. and Srinivasan, R. (2011). Immune-system-inspired approach to process monitoring and fault diagnosis. *Ind. Eng. Chemi. Res.*, 50(3), pp. 1637–1651.
- Golzari, S., Doraisamy, S., Sulaiman, M. N. and Udzir, N. I. (2011). An efficient and effective immune based classifier. *J. Comput. Sci.*, 7(2), pp. 148–153.
- Gong, M., Chen, X., Ma, L., Zhang, Q. and Jiao, L. (2013). Identification of multi-resolution network structures with multi-objective immune algorithm. *Appl. Soft Comput.*, 13(4), pp. 1705–1717.
- Gong, M., Zhang, J., Ma, J. and Jiao, L. (2012). An efficient negative selection algorithm with further training for anomaly detection. *Knowl. Based Syst.*, 30, pp. 185–191.
- Greensmith, J. and Aickelin, U. (2009). Artificial dendritic cells: Multi-faceted perspectives. In Bargiela, A. and Pedrycz, W. (eds.), *Human-Centric Information Processing Through Granular Modelling, Studies in Computational Intelligence*. Berlin/Heidelberg: Springer, pp. 375–395.
- Groat, M. M., Edwards, B., Horey, J., He, W. and Forrest, S. (2013). Application and analysis of multidimensional negative surveys in participatory sensing applications. *Pervasive Mob. Comput.*, 9(3), pp. 372–391.
- Gu, F., Greensmith, J. and Aickelin, U. (2013). Theoretical formulation and analysis of the deterministic dendritic cell algorithm. *Biosystems*, 111(2), pp. 127–135.
- Guzella, T. S., Mota-Santos, T. A. and Caminhas, W. M. (2007). A novel immune inspired approach to fault detection. *Proc. 6th Int. Conf. Artificial Immune Systems*, Berlin, Heidelberg, ICARIS'07, Springer-Verlag, pp. 107–118.
- Guzella, T., Mota-Santos, T. and Caminhas, W. (2008). Artificial immune systems and kernel methods. In Bentley, P., Lee, D. and Jung, S. (eds.), *Artificial Immune Systems, Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer, pp. 303–315.
- Haktanirlar Ulutas, B. and Kulturel-Konak, S. (2011). A review of clonal selection algorithm and its applications. *Artif. Intell. Rev.*, 36(2), pp. 117–138.
- Hart, E., McEwan, C. and Davoudani, D. (2009). Exploiting collaborations in the immune system: The future of artificial immune systems. In Mumford, C. and Jain, L. (eds.), *Computational Intelligence, Intelligent Systems Reference Library*. Berlin, Heidelberg: Springer, pp. 527–558.

- Hilaire, V., Lauri, F., Gruer, P., Koukam, A. and Rodriguez, S. (2010). Formal specification of an immune based agent architecture. *Eng. Appl. Artif. Intel.*, 23(4), pp. 505–513.
- Hilder, J., Owens, N., Hickey, P., Cairns, S., Kilgour, D., Timmis, J. and Tyrrell, A. (2011). Parameter optimisation in the receptor density algorithm. In Lió, P., Nicosia, G. and Stibor, T.(eds.), *Artificial Immune Systems*, Lecture Notes in Computer Science. Berlin Heidelberg: Springer, pp. 226–239.
- Hu, Z.-H., Ding, Y.-S. and Lua, X.-J. (2008). Cognitive immune system-based co-evolutionary information retrieval model. *2007 Workshop on Intelligent Information Technology Applications*, 3, pp. 212–215.
- Hu, Z.-H., Yu, X.-K. and Sheng, Z.-H. (2013). An immune co-evolutionary algorithm based approach for problems related to clothing uniform assignment. *Int.J.Cloth.Sci.Tech.*, 25(1), pp. 70–82.
- Jabeen, H. and Baig, A. (2010). Clonal-gp framework for artificial immune system inspired genetic programming for classification. In Setchi, R., Jordanov, I., Howlett R. and Jain, L.(eds.), *Knowledge-Based and Intelligent Information and Engineering Systems*, Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 61–68.
- Jansen, T. and Zarges, C. (2011). Variants of immune inspired somatic contiguous hypermutations. *Theor. Comput. Sci.*, 412(6), pp. 517–533.
- Jarosz, P. and Burczynski, T. (2010). Coupling of immune algorithms and game theory in multiobjective optimization. In Rutkowski, L., Scherer, R., Tadeusiewicz, R., Zadeh, L. and Zurada, J. (eds.), *Artifical Intelligence and Soft Computing*, Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 500–507.
- Jenhani, I. and Elouedi, Z. (2012). Re-visiting the artificial immune recognition system: A survey and an improved version. *Artif. Intel. Rev.*, pp. 1–13.
- Ji, Z. and Dasgupta, D. (2009). V-detector: An efficient negative selection algorithm with “probably adequate” detector coverage. *Inform. Sciences*, 179(10), pp. 1390–1406.
- Jinyin, C. and Dongyong, Y. (2011). A study of detector generation algorithms based on artificial immune in intrusion detection system. *2011 3rd Int. Conf. Computer Research and Development (ICCRD)*, Vol. 1, pp. 4–8.
- Karakasis, V. and Stafylopatis, A. (2008). Efficient evolution of accurate classification rules using a combination of gene expression programming and clonal selection. *Evolutionary Computation, IEEE Trans.*, 12(6), pp. 662–678.
- Karakose, M. (2013). Reinforcement learning based artificial immune classifier. *The Scientific World Journal*, p.7.
- Khan, M. T. and de Silva, C. W. (2012). Autonomous and robust multi-robot cooperation using an artificial immune system. *Int. J. Robot. Autom.*, 27(1).
- Knidel, H., de Castro, L. N. and Zuben, F. J. V. (2005). Rabnet: A real-valued antibody network for data clustering. *Genetic and Evolutionary Computation Conference, GECCO 2005, Proc., Washington DC, USA*, June 25–29, Beyer, H.-G. and Reilly, U.-M. O. (eds.), ACM, pp. 371–372.
- Laurentys, C. A., Palhares, R. M. and Caminhas, W. M. (2011). A novel artificial immune system for fault behavior detection. *Expert Syst. Appl.*, 38(6), pp. 6957–6966.
- Li, G., Li, T., Zeng, J. and Li, H. (2010). An outlier robust negative selection algorithm inspired by immune suppression. *J. Comput.*, 5(9).
- Li, Z. and He, C. (2013). Optimal scheduling-based {RFID} reader-to-reader collision avoidance method using artificial immune system. *Appl. Soft Comput.*, 13(5), pp. 2557–2568.
- Liskiewicz, M. and Textor, J. (2010). Negative selection algorithms without generating detectors. *Proc. 12th Annual Conf. Genetic and Evolutionary Computation*, New York, NY, USA, GECCO’10, ACM, pp. 1047–1054.
- Liu, C.-H., Huang, W.-H. and Chang, P.-C. (2012). A two-stage ais approach for grid scheduling problems. *Int. J. Prod. Res.*, 50(10–12), pp. 2665–2680.
- Liu, R., Jiao, L., Li, Y. and Liu, J. (2010). An immune memory clonal algorithm for numerical and combinatorial optimization. *Frontiers of Computer Science in China*, 4(4), pp. 536–559.
- Liu, R., Niu, M. and Jiao, L. (n.d.). A new artificial immune network classifier for SAR image. 7496, pp. 74960W.
- Liu, T., Zhang, L. and Shi, B. (2009). Adaptive immune response network model. In Huang, D.-S., Jo, K.-H., Lee, H.-H., Kang, H.-J. and Bevilacqua, V. (eds.), *Emerging Intelligent Computing Technology and Applications. With Aspects of Artificial Intelligence*, Lecture Notes in Computer Science. Berlin Heidelberg: Springer, pp. 890–898.

- Ma, W., Jiao, L. and Gong, M. (2009). Immunodominance and clonal selection inspired multiobjective clustering. *Prog. Nat. Sci.*, 19(6), pp. 751–758.
- Ma, W., Tran, D. and Sharma, D. (2008). Negative selection with antigen feedback in intrusion detection. In Bentley, P., Lee, D. and Jung, S. (eds.), *Artificial Immune Systems*, Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 200–209.
- Mahapatra, P. K., Kaur, M., Sethi, S., Thareja, R., Kumar, A. and Devi, S. (2013). Improved thresholding based on negative selection algorithm (nsa). *Evolutionary Intelligence*, pp. 1–14.
- Malhotra, A., Baheti, A. and Gupta, S. (2012). Pattern recognition approaches inspired by artificial immune system. *Int. J. Comput. Appl.*, 44(20), pp. 12–16.
- Masutti, T. and Castro, L. (2008). A neuro-immune algorithm to solve the capacitated vehicle routing problem. In Bentley, P., Lee, D. and Jung, S. (eds.), *Artificial Immune Systems*, Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 210–219.
- McEwan, C and Hart, E. (2009). Representation in the (artificial) immune system. *J. Mathematical Modelling and Algorithms*, 8(2), pp. 125–149.
- McEwan, C and Hart, E. (2010). Clonal selection from first principles. In Hart, E., McEwan, C., Timmis J. and Hone, A. (eds.), *Artificial Immune Systems*, Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 18–32.
- Mo, H. (2008). *Handbook of Research on Artificial Immune Systems and Natural Computing: Applying Complex Adaptive Technologies*. Hershey, PA: IGI Publishing.
- Mohamed Elsayed, S., Rajasekaran, S. and Ammar, R. (2012). An artificial immune system approach to associative classification. In Murgante, B., Gervasi, O., Misra, S., Nedjah, N., Rocha, A., Taniar, D. and Apduhan, B. (eds.), *Computational Science and Its Applications – ICCSA 2012*, Lecture Notes in Computer Science. Berlin Heidelberg: Springer, pp. 161–171.
- Mohammadi, M., Akbari, A., Raahemi, B. and Nassersharif, B. (2012). A real time anomaly detection system based on probabilistic artificial immune based algorithm. In Coello Coello, C. A., Greensmith, J., Krasnogor, N., Lio, P., Nicosia, G. and Pavone, M. (eds.), *Artificial Immune Systems*, Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 205–217.
- Muddada, R., Zhang, F., Tiwari, M. and Zhang, W. (2010). A new index to evaluate solutions in the clonalg algorithm: Structural similarity index. In Huang, G., Mak, K. and Maropoulos, P. (eds.), *Proceedings of the 6th CIRP-Sponsored International Conference on Digital Enterprise Technology, Advances in Intelligent and Soft Computing*. Berlin Heidelberg: Springer, pp. 1119–1132.
- Nanas, N., Vavalis, M. and Roeck, A. (2010). Words, antibodies and their interactions. *Swarm Intelligence*, 4(4), pp. 275–300.
- Narayanan, A. and Ahmad, W. (2012). Humoral artificial immune system (hais) for supervised learning. *Int. J. Comput. Int. Appl.*, 11(01), p. 1250004.
- Navlakha, S. and Bar-Joseph, Z. (2011). Algorithms in nature: The convergence of systems biology and computational thinking. *Mol. Syst. Biol.*, 7, 546.
- Nejad, F., Salkhi, R., Azmi, R. and Pishgoo, B. (2012). Structural trl algorithm for anomaly detection based on danger theory. *9th International ISC Conf. Information Security and Cryptology (ISCISC)*, pp. 156–161.
- Oliveira, L. O. V. B., Drummond, I. N. and Pappa, G. L. (2013). A new representation for instance-based clonal selection algorithms. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pp. 2259–2266.
- Owens, N. D., Greensted, A., Timmis, J. and Tyrrell, A. (2013). The receptor density algorithm. *Theor. Comp. Sci.*, 481, pp. 51–73.
- Ozsen, S., Gunes, S., Kara, S. and Latifoglu, F. (2009). Use of kernel functions in artificial immune systems for the nonlinear classification problems. *Information Technology in Biomedicine, IEEE Trans.*, 13(4), pp. 621–628.
- Pathak, V., Dhyani, P. and Mahanti, P. (2012). Aiden: A density conscious artificial immune system for automatic discovery of arbitrary shape clusters in spatial patterns. *Broad Research in Artificial Intelligence and Neuroscience (BRAIN)*, 3(3), pp. 5–11.
- Poggiolini, M. and Engelbrecht, A. (2013). Application of the feature-detection rule to the negative selection algorithm.

Expert Syst. Appl., 40(8), pp. 3001–3014.

- Ramakrishnan, S. and Srinivasan, S. (2009). Intelligent agent based artificial immune system for computer security—a review. *Artif. Intel. Rev.*, 32(1–4), pp. 13–43.
- Raza, A. and Fernandez, B. R. (2012). Immuno-inspired robotic applications: A review. *CoRR abs/1202.4261*.
- Riff, M. C., Montero, E. and Neveu, B. (2013). Reducing calibration effort for clonal selection based algorithms: A reinforcement learning approach. *Knowl.-Based Syst.*, 41, pp. 54–67.
- Secker, A., Freitas, A. A. and Timmis, J. (2008). Aisiid: An artificial immune system for interesting information discovery on the web. *Appl. Soft Comput.*, 8(2), pp. 885–905.
- Shafiq, M. Z., Khayam, S. A. and Farooq, M. (2008). Improving accuracy of immune-inspired malware detectors by using intelligent features. *Proc. 10th Annual Conf. Genetic and Evolutionary Computation*, New York, NY, USA, GECCO '08, ACM, pp. 119–126.
- Shang, R., Jiao, L., Liu, F. and Ma, W. (2012). A novel immune clonal algorithm for mo problems. *Evolutionary Computation, IEEE Trans.*, 16(1), pp. 35–50.
- Sharma, A. and Sharma, D. (2011). Clonal selection algorithm for classification. In Lio, P., Nicosia, G. and Stibor, T. (eds.), *Artificial Immune Systems*, Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 361–370.
- Shi, Y., Xie, G., Zhang, Y., Peng, X., Huang, B., Yao, D. and Yue, G. (2013). An immune-based scgm(1,1)c prediction model for network security situation. *J. Comput. Inform. Syst.*, 9, pp. 4395–4406.
- Shimo, H. K. and Tinos, R. (2013). Use of self-organizing suppression and q-gaussian mutation in artificial immune systems. *International Journal of Intelligent Computing and Cybernetics*, 6(3), pp. 296–322.
- Singh, C. T., Sahoo, G. and Ghose, M. K. (2011). Artificial immune privileged sites as an enhancement to immuno-computing paradigm. *CoRR abs/1102.3971*.
- Singh, S., Singh, J. P. and Shrivastva, G. (2013). A review: A is based intrusion detection system. *Int. J. Adv. Res. Comput. Commun. Eng.*, 2(6), pp. 2387–2391.
- Suarez-Tangil, G., Palomar, E., Pastrana, S. and Ribagorda, A. (2011). Artificial immunity-based correlation system. In *SECRYPT 2011—Proceedings of the International Conference on Security and Cryptography, Seville, Spain, July, 18–21 SECRYPT is part of ICETE—The International Joint Conference on e-Business and Telecommunications*, Lopez, J. and Samarati, P. (eds.), SciTePress, pp. 422–425.
- Swiecicki, M. (2008). An algorithm of decentralized artificial immune network and its implementation. In Darzentas, J., Vouros, G., Vosinakis, S. and Arnellos, A. (eds.), *Artificial Intelligence: Theories, Models and Applications*, Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 407–412.
- Szabo, A., de Castro, L. and Delgado, M. (2012). Fainet: An immune algorithm for fuzzy clustering. In *2012 IEEE Int. Conf. Fuzzy Systems (FUZZ-IEEE)*, pp. 1–9.
- Tang, W., Peng, L.-M., Yang, X.-M., Xie, X. and Cao, Y. (2010). Gep-based framework for immune-inspired intrusion detection. *TIIS*, 4(6), pp. 1273–1293.
- Tauber, A. I. (2008). The immune system and its ecology. *Philos. Sci.* 75(2), pp. 224–245.
- Taud, H., Herrera-Lozada, J. C. and Alvarez-Cedillo, J. (2010). Adaboost classifier by artificial immune system model. In Martínez-Trinidad, J. F., Carrasco-Ochoa, J. A. and Kittler, J. (eds.), *Advances in Pattern Recognition*, Lecture Notes in Computer Science. Berlin Heidelberg: Springer, pp. 171–179.
- Timmis, J., Andrews, P. and Hart, E. (2010). On artificial immune systems and swarm intelligence. *Swarm Intelligence*, 4(4), pp. 247–273.
- Timmis, J., Andrews, P., Owens, N. and Clark, E. (2008). Immune systems and computation: An interdisciplinary adventure. In Calude, C., Costa, J., Freund, R., Oswald, M. and Rozenberg, G. (eds.), *Unconventional Computing*, Lecture Notes in Computer Science. Berlin Heidelberg: Springer, pp. 8–18.
- Twycross, J. and Aickelin, U. (2010). Information fusion in the immune system. *CoRR abs/1003.1598*.
- Ulutas, B. H. and Kulturel-Konak, S. (2013). Assessing hypermutation operators of a clonal selection algorithm for the unequal area facility layout problem. *Eng. Optim.*, 45(3), pp. 375–395.
- Unterleitner, M. (2012). *Computer Immune System for Intrusion and Virus Detection: Adaptive detection mechanisms*

and their implementation. AV Akademikerlag.

- Vella, M., Roper, M. and Terzis, S. (2010). Danger theory and intrusion detection: Possibilities and limitations of the analogy. In Hart, E., McEwan, C., Timmis, J. and Hone, A. (eds.), *Artificial Immune Systems*, Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 276–289.
- Voigt, D., Wirth, H. and Dilger, W. (2007). A computational model for the cognitive immune system theory based on learning classifier systems. In *Proc. 6th Int. Conf. Artificial Immune Systems*, Berlin, Heidelberg, ICARIS'07, Springer-Verlag, pp. 264–275.
- Wang, D., Zhang, F. and Xi, L. (2011). Evolving boundary detector for anomaly detection. *Expert Syst. Appl.*, 38(3), pp. 2412–2420.
- Wang, S., Yang, T. and Wang, K. (2012). Self/non-self discrimination based on fractional distance. In *Computer Science Service System (CSSS), 2012 Int. Conf.*, pp. 1777–1780.
- Wang, X. and Jiang, X. (2011). Artificial malware immunization based on dynamically assigned sense of self. In Burmester, M., Tsudik, G., Magliveras, S. and Ilić, I. (eds.), *Information Security*, Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 166–180.
- Wilson, W. O., Birkin, P. and Aickelin, U. (2010). The motif tracking algorithm. *CoRR abs/1006.1526*.
- Wong, E. and Lau, H. Y. K. (2009). Advancement in the twentieth century in artificial immune systems for optimization: Review and future outlook. *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE Int. Conf.*, pp. 5195–4202.
- Wu, J.-Y. (2012). Solving constrained global optimization problems by using hybrid evolutionary computing and artificial life approaches. *Math. Probl. Eng.*, 36.
- Xiao, R., Chen, T. and Tao, Z. (2011). An analytical approach to the similarities between swarm intelligence and artificial immune system. *Innovations in Bio-inspired Computing and Applications (IBICA), 2011 Second Int. Conf.*, pp. 112–115.
- Xu, B., Luo, W., Pei, X., Zhang, M. and Wang, X. (2009). On average time complexity of evolutionary negative selection algorithms for anomaly detection. *Proc. first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, New York, NY, USA, GEC '09, ACM, pp. 631–638.
- Xu, Q., Wang, L. and Si, J. (2010). Predication-based immune network for multimodal function optimization. *Eng. Appl. Artif. Intel.*, 23(4), pp. 495–504.
- Xu, Q., Wang, S. and Zhang, C. (2012). Structural design of the danger model immune algorithm. *Inf. Sci.*, 205, pp. 20–37.
- Yang, H., Guo, J. and Deng, F. (2011). Collaborative RFID intrusion detection with an artificial immune system. *J. Intell. Inf. Syst.*, 36(1), pp. 1–26.
- Ying, W. (2013). The semi-supervised immune classifier generation algorithm based on data clustering. *J. Computational Information Systems*, 9(9), pp. 3407–3414.
- Yu, S. and Dasgupta, D. (2011). An effective network-based intrusion detection using conserved self pattern recognition algorithm augmented with near-deterministic detector generation. *Computational Intelligence in Cyber Security (CICS), 2011 IEEE Symp.*, pp. 17–24.
- Yunfang, C. (2012). A General Framework for Multi-Objective Optimization Immune Algorithms. *Int. J. Intell. Syst. Appli.*, 4(6), pp. 1–13.
- Zeng, J., Qin, Z. and Tang, W. (2013). Anomaly detection using a novel negative selection algorithm. *J. Comput. Theor. Nanosci.*, 10(12), pp. 2831–2835.
- Zhang, C. and Yi, Z. (2010). A danger theory inspired artificial immune algorithm for on-line supervised two-class classification problem. *Neurocomputing*, 73(7–9), pp. 1244–1255.
- Zhang, R., Li, T., Xiao, X. and Shi, Y. (2013). A danger-theory-based immune network optimization algorithm. *The Scientific World J.*, p.13.
- Zhang, Z., Yue, S., Liao, M. and Long, F. (2013). Danger theory based artificial immune system solving dynamic constrained single-objective optimization. *Soft Comput.*, pp. 1–22.
- Zhao, D. and Luo, W. (2013). Real-valued negative databases. *Proc. 12th European Conf. Artificial Life (ECAL 2013)*, pp. 884–890.

- Zheng, X., Zhou, Y. and Fang, Y. (2013). The dual negative selection algorithm based on pattern recognition receptor theory and its application in two-class data classification. *J. Comput.*, 8(8), pp. 1951–1959.
- Zhu, Y. and Tan, Y. (2011). A danger theory inspired learning model and its application to spam detection. In Tan, Y., Shi, Y., Chai Y. and Wang, G. (eds.), *Advances in Swarm Intelligence*, Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 382–389.
- Zhu, Y. and Tan, Y. (2011). A local-concentration-based feature extraction approach for spam filtering. *Information Forensics and Security, IEEE Trans.*, 6(2), pp. 486–497.

¹ The Danger Model is seen as a theory, but according to immunologists, it is postulated as a model rather than a theory. This is also applied to Infectious Non-self Model.

² The Clonal Selection, according to immunologists, is in fact accepted as a theory.

Chapter 16

Swarm Intelligence: An Introduction, History and Applications

Fevrier Valdez

In this paper, a survey about the algorithms based in swarm intelligence (SI) is presented. In this case, are analyzed the most popular algorithms such as the particle swarm optimization (PSO), ant colony optimization (ACO), bee swarm optimization (BCO), and the Bat algorithm (BA). These algorithms are mentioned in the paper because they have demonstrated be superior with respect to the other optimization methods based in swarms in some applications, and also the algorithms are inspired in swarm intelligence and are similar in some aspects.

16.1. Introduction

Swarm intelligence (SI) originated from the study of colonies, or swarms of social organisms. Studies of the social behavior of organisms in swarms prompted the design of very efficient optimization and clustering algorithms. For example, simulation studies of the graceful, but unpredictable, choreography of bird flocks led to the design of the particle swarm optimization (PSO) algorithm, and studies of the foraging behavior of ants resulted in ant colony optimization (ACO) algorithms. A major thrust in algorithmic development is the design of algorithmic models to solve increasingly complex problems. Enormous successes have been achieved through the modeling of biological and natural intelligence, resulting in so-called “intelligent systems”. These intelligent algorithms include artificial neural networks, evolutionary computation, swarm intelligence, artificial immune systems, and fuzzy systems. Together with logic, deductive reasoning, expert systems, case-based reasoning, and symbolic machine learning systems, these intelligent algorithms form part of the field of *Artificial Intelligence* (AI). Just looking at this wide variety of AI techniques, AI can be seen as a combination of several research disciplines, for example, computer science, physiology, philosophy, sociology, and biology.

Studies of social animals and social insects have resulted in a number of computational models of SI. Biological swarm systems that have inspired computational models include ants, termites, bees, spiders, fish schools, and bird flocks. Within these swarms, individuals are relatively simple in structure, but their collective behavior is usually very complex. The complex behavior of a swarm is a result of the pattern of interactions between the individuals of the swarm over time. This complex behavior is not a property of any single individual, and is usually not easily predicted or deduced from the simple behaviors of the individuals. This paper is organized as follows: in [Section 16.1](#), a brief introduction about the swarm intelligence is presented, [Section 16.2](#) describe with detail the methods analyzed in the paper, [Section 16.3](#) shows the applications that have been used for these methods. In [Section 16.4](#), the conclusion are presented and finally the references are included in the last section.

16.2. History

Nowadays application of SI algorithms has been very popular because they are able to solve a variety of complex optimization problems. The term “Swarm intelligence”, that denotes this “collective intelligence” has come into use (Beni, 1988, 1989, 1992; Bonabeau *et al.*, 1997). Swarm Intelligence (Bonabeau *et al.*, 1997) is the part of AI based on study of actions of individuals in various decentralized systems. Therefore, in this paper, we explain some heuristic techniques based on population and throughout history have been used by researchers as optimization methods to solve problems of everyday life and industry. This chapter relates the history and description of the stochastic algorithms PSO, ACO, BCO, and BA. However, in this section, the most popular optimization algorithms based in populations in chronological order are presented in [Table 16.1](#).

16.2.1. Ant Colony Optimization

One of the first behaviors studied by researchers was the ability of ants to find the shortest path between their nest and a food source. From these studies and observations followed the first algorithmic models of the foraging behavior of ants, as developed by Marco Dorigo (Dorigo, 1992, 1994). Since then, research in the development of Algorithms, based on Ants have been very popular, resulting in a large number of algorithms and applications. These algorithms that were developed as a result of studies of ant foraging behavior are referred to as instances of the ant colony optimization metaheuristic (Dorigo *et al.*, 2000; Dorigo and Caro, 1999). This paper provides an overview of the ACO algorithms, and the first algorithms that have been developed. [Section 16.2.1](#) gives an overview of the foraging behavior of real ants, and introduces the concepts of stigmergy and artificial ants. This section presents a very simple ant algorithm implementation to illustrate the basic principles of these algorithms. Sections 16.2.1.1 to 16.2.1.4 respectively, discuss the ant system (AS), elitist system (EAS), rank-based AS (ASrank) and max-MIN ant system (MMAS). These algorithms were the first set of ant algorithms implemented, mainly with reference to the traveling salesman problem (TSP).

Table 16.1: Popular optimization methods inspired in populations in chronological order.

Year	Optimization methods inspired in populations
2010	Bat Algorithm (BA) [5]
2010	Artificial Bee Algorithm [6]
2009	Gravitational Search Algorithm [7]
2007	Intelligent water drops [8]
2005	Harmony Search Algorithm [9]
2005	Honey Bee Algorithm [10]
2002	Bacterial Foraging Algorithm [11]
2002	Estimation of Distribution Algorithm [12]
1995	PSO [13]

1992	ACO [14]
1989	Tabu Search [15]
1983	Simulated Annealing [16]
1979	Cultural Algorithms [17]
1975	GAs [18]
1966	Evolutionary Programming [19]
1965	Evolution Strategies [20]

The first ACO algorithm has been applied to the TSP (Dorigo, 1992, 1994; Dorigo *et al.*, 2000; Dorigo and Caro, 1999). Given a set of cities and the distances between them, the TSP is the problem of finding the shortest possible path which visits every city exactly once. Possible path which visits every city exactly once, it can be represented by a complete weighted graph $G = (N, E)$ N is the set of nodes representing the cities and E is the set of edges. Each edge is assigned a value d_{ij} which is the distance between cities i and j . When applying the ACO algorithm to the TSP, a pheromone strength $\tau_{ij}(t)$ is associated to each edge (i, j) , where $\tau_{ij}(t)$ is a numerical value which is modified during the execution of the algorithm and t is the iteration counter.

The skeleton of the ACO algorithm applied to the TSP is:

```

procedure ACO algorithm for TSPs
set parameters, initialize pheromone
trails while
(termination
condition not
met) do Tour
construction
Pheromone
update end
end

```

The ACO metaheuristic is inspired by observing the behavior of real ant colonies, which presented an interesting feature, on how to find the shortest paths between the nest and food, on its way the ants deposit a substance called pheromone, this trail allows the ants back to their nest from the food; it uses the evaporation of pheromone to avoid a unlimited increase of pheromone trails and allow to forget the bad decisions (Dorigo and Gambardella, 1996, 1997). Where is the probability of ant k goes from node i to node j , is the heuristic information, say, prior knowledge, is the possible neighborhood of an ant k

when in the city i , is the pheromone trail, α and β are parameters that determine the influence of the pheromone and prior knowledge, respectively.

16.2.2. Ant System

Initially, three different versions of AS were proposed (Dorigo, 1992, 1994; Dorigo *et al.*, 2000; Dorigo and Caro, 1999). These were called ant-density, antquantity, and ant-cycle. Whereas in the ant-density and ant-quantity versions the ants updated the pheromone directly after a move from one city to an adjacent city, in the ant-cycle version the pheromone update was only done after all the ants had constructed the tours and the amount of pheromone deposited by each ant was set to be a function of the tour quality. Nowadays, when referring to AS, one actually refers to ant-cycle since the two other variants were abandoned because of their inferior performance.

The two main phases of the AS algorithm constitute the ants' solution construction and the pheromone update. In AS a good heuristic to initialize the pheromone trails is to set them to a value slightly higher than the expected amount of pheromone deposited by the ants in one iteration; a rough estimate of this value can be obtained by setting, $\forall(i, j)$, $\tau_{ij} = \tau_0 = m/C^{nn}$, where m is the number of ants, and c^{nn} is the length of a tour generated by the nearest-neighbor heuristic. The reason for this choice is that if the initial pheromone values τ_0 's are too low, then the search is quickly biased by the first tours generated by the ants, which in general leads toward the exploration of inferior zones of the search space. On the other side, if the initial pheromone values are too high, then many iterations are lost waiting until pheromone evaporation reduces enough pheromone values, so that pheromone added by ants can start to bias the search.

In AS, m ants build a tour of the TSP. Initially, ants are put on randomly chosen cities. At each construction step, ant k applies a probabilistic action choice rule, called *random proportional rule*, to decide which city to visit next. In particular, the probability with which ant k , currently at city i , chooses to go to city j is

$$p_{ij}^k \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum l \in N_i^k [\tau_{il}]^\alpha [\eta_{il}]^\beta}, \quad \text{if } j \in N_i^k, \quad (1)$$

where $\eta_{ij} = 1/d_{ij}$ is a heuristic value that is available *a priori*, α and β are two parameters which determine the relative influence of the pheromone trail and the heuristic information, and N_i^k is the feasible neighborhood of ant k when being at city i , that is, the set of cities that ant k has not visited yet. The update of pheromone trails is as follow:

After all the ants have constructed their tours, the pheromone trails are updated. This is done by first lowering the pheromone value on all arcs by a constant factor, and the adding pheromone on the arcs the ants have crossed in their tours, pheromone evaporation is made by

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij}, \quad \forall(i, j) \in L, \quad (2)$$

where $0 < p \leq 1$. The parameter ρ is used to avoid unlimited accumulation of the pheromone trails and it enables the algorithm to delete bad decisions previously taken by ants. In fact, if an arc is not chosen by the ants, its associated pheromone value decreases exponentially in the number of iterations. After evaporation, all ants deposit pheromone on the arcs they have crossed in their tour:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^m \Delta\tau_{ij}^k \quad \forall(i, j) \in L, \quad (3)$$

where $\Delta\tau_{ij}^k$ is amount of pheromone ant k deposits on the arcs it has visited.

16.2.3. Elitist Ant System

A first improvement on the initial AS, called the elitist strategy for Ant System (EAS), was introduced in Dorigo (Dorigo and Caro, 1999; Dorigo and Gambardella, 1996, 1997). The idea is to provide strong additional reinforcement to the arcs belonging to the best tour found since the start of the algorithm; this tour is denoted as T^{bs} (best-so-far tour) in the following. Note that this additional feedback to the best-so-far tour (which can be viewed as additional pheromone deposited by an additional ant called best-so-far ant) is another example of a daemon action of the ACO metaheuristic. The update of pheromone trails is very similar as in AS and the only difference in the additional reinforcement of tour T^{bs} is achieved by adding a quantity e/C^{bs} to its arcs, where e is a parameter that defines the weight given to the best-so-far tour T^{bs} , and C^{bs} is its length. Thus, the following equation for the pheromone deposit becomes

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^m \Delta\tau_{ij}^k + e \Delta\tau_{ij}^{bs}, \quad (4)$$

where $\Delta\tau_{ij}^k$ is defined as in AS.

In EAS, the pheromone evaporation is implemented as in AS. Computational results presented in Dorigo (Dorigo, 1992, 1994; Dorigo and Gambardella, 1996, 1997) suggest that the use of the elitist strategy with an appropriate value for parameter e allows AS to both find better tours and find them in a lower number of iterations.

16.2.4. Rank-Based Ant System

Bullnheimer *et al.* (1997) proposed a modification of AS as follows: Equation (1) to allow only the best ant to update pheromone concentrations on the links of the global-best path, Equation (2) to use elitist ants, and Equation (3) to let ants update pheromone on the basis of a ranking of the ants. In ASrank the ants are sorted by increasing tour length and the quantity of pheromone that an ant deposits is weighted according to the rank r of the ant.

In the iteration only the $(w - 1)$ best-ranked ants and the ant that produced the best-so-far tour are allowed to deposit pheromone. The best-so-far tour gives the strongest feedback, with w . For ASrank, the global update rule is defined as follow:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{r=1}^{w-1} (w - r) \Delta \tau_{ij}^r + w \Delta \tau_{ij}^{bs}. \quad (5)$$

The results of an experimental evaluation by Bullnheimer *et al.* (1999c) suggest that ASrank performs slightly better than EAS and significantly better than AS.

16.2.5. MAX-MIN Ant System

MMAS (Stützle, 1997a, 1997b) introduces four main modifications with respect to AS. First, it strongly exploits the best tours found: only either the iteration-best ant, that is, the ant that produced the best tour in the current iteration, or the best-so-far ant is allowed to deposit pheromone. Unfortunately, such a strategy may lead to a stagnation situation in which all the ants follow the same tour, because of the excessive growth of pheromone trails on arcs of a good, although suboptimal, tour. To counteract this effect, a second modification introduced by MMAS is that it limits the possible range of pheromone trail values to the interval $[\tau_{min}, \tau_{max}]$. Third, the pheromone trails are initialized to the upper pheromone trail limit, which, together with a small pheromone evaporation rate, increases the exploration of tours at the start of the search. Finally, in MMAS, pheromone trails are reinitialized each time the system approaches stagnation or when no improved tour has been generated for a certain number of consecutive iterations. The update of pheromone trail is as follows:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta \tau_{ij}^{best}. \quad (6)$$

Experimental results indicate that for small TSP instances it may be best to use only iteration-best pheromone updates, while for large TSPs with several hundreds of cities the best performance is obtained by giving an increasingly stronger emphasis to the best-so-far tour T^{bs} is chosen for the trail update (Stützle, 1999).

16.2.6. Particle Swarm Optimization

PSO is a population-based stochastic optimization technique developed by Eberhart and Kennedy in 1995, inspired by the social behavior of bird flocking or fish schooling (Kennedy and Eberhart, 1995).

PSO shares many similarities with evolutionary computation techniques such as GA (Castillo and Melin, 2002). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike the GA, the PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called

particles, fly through the problem space by following the current optimum particles (Angeline, 1998).

Each particle keeps track of its coordinates in the problem space, which are associated with the best solution (fitness) it has achieved so far (The fitness value is also stored). This value is called *pbest*. Another “best” value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the neighbors of the particle. This location is called *lbest*. When a particle takes all the population as its topological neighbors, the best value is a global best and is called *gbest*.

The particle swarm optimization concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its *pbest* and *lbest* locations (local version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward *pbest* and *lbest* locations.

In the past several years, PSO has been successfully applied in many research and application areas. It is demonstrated that PSO gets better results in a faster, cheaper way compared with other methods (Angeline, 1998).

Another reason that PSO is attractive is that there are few parameters to adjust. One version, with slight variations, works well in a wide variety of applications. Particle swarm optimization has been used for approaches that can be used across a wide range of applications, as well as for specific applications focused on a specific requirement.

The pseudo code of the PSO is as follows

For each particle

Initialize particle

End

Do

For each particle

Calculate fitness value

If the fitness value is better than the best fitness value (pBest) in history

set current value as the new pBest

End

Choose the particle with the best fitness value of all the particles as the gBest

For each particle

Calculate particle velocity

Update particle position

End

While maximum iterations or minimum error criteria is not attained

16.2.7. Bee Colony Optimization

The Bee Colony Optimization (BCO) metaheuristic (Lučić and Teodorović, 2001, 2002, 2003a, 2003b; Teodorović, 2003b) was proposed recently by Lučić and Teodorović as a new direction method in the field of Swarm Intelligence. The BCO has been successfully applied to various engineering and management problems by Teodorović and coauthors (Teodorović and Dell'Orco, 2005; Teodorović *et al.*, 2006; Marković *et al.*, 2007; Teodorović and Šelmić, 2007). The BCO approach is a “bottom-up” approach to modeling where special kinds of artificial agents are created by analogy with bees. Artificial bees represent agents, which collaboratively solve complex combinatorial optimization problem.

Lučić and Teodorović (Lučić and Teodorović, 2001, 2002, 2003a, 2003b) were among first who used basic principles of collective bee intelligence in solving combinatorial optimization problems. The BCO is a population-based algorithm. Population of *artificial bees* searches for the optimal solution. Artificial bees represent agents, which collaboratively solve complex combinatorial optimization problems. Every artificial bee generates one solution to the problem. The algorithm consists of two alternating phases: *forward pass* and *backward pass*. In each forward pass, every artificial bee is exploring the search space. It applies a predefined number of moves, which construct and/or improve the solution, yielding to a new solution. Having obtained new partial solutions, the bees go again to the nest and start the second phase, the so-called backward pass. In the backward pass, all artificial bees share information about their solutions.

In nature, bees would perform a dancing ceremony, which would notify other bees about the quantity of food they have collected, and the closeness of the patch to the nest. In the BCO, the artificial bees publicize the quality of the solution, i.e., the objective function value. During the backward pass, every bee decides with a certain probability whether to abandon the created partial solution and become again uncommitted follower, or dance and thus recruit the nest mates before returning to the created partial solution. During the second forward pass bees expand previously created partial solutions, by a predefined number of nodes, and after that perform again the backward pass and return to the hive. In the hive, bees again participate in a decision-making process, make a decision, perform third forward pass, etc. The two phases of the search algorithm, forward and backward pass, are performed *iteratively*, until a stopping condition is met. The possible stopping conditions could be, for example, the maximum total number of forward/backward passes, the maximum total number of forward/backward passes without the improvement of the objective function, etc.

The algorithm parameters whose values need to be set prior the algorithm execution

are as follows:

B — The number of bees in the hive.

NC — The number of constructive moves during one forward pass.

In the beginning of the search, all the bees are in the hive. The following is the pseudocode of the BCO algorithm (Lučić and Teodorović, 2001, 2002):

1. Initialization: every bee is set to an empty solution;
2. For every bee do the forward pass:
 - (a) Set $k = 1$; //counter for constructive moves in the forward pass;
 - (b) Evaluate all possible constructive moves;
 - (c) According to evaluation, choose one move using the roulette wheel;
 - (d) $k = k + 1$; If $k \leq NC$ Go To Step b.
3. All bees are back to the hive; // backward pass starts;
4. Sort the bees by their objective function value;
5. Every bee decides randomly whether to continue its own exploration and become a recruiter, or to become a follower (bees with higher objective function value have greater chance to continue its own exploration);
6. For every follower, choose a new solution from recruiters by the roulette wheel;
7. If the stopping condition is not met Go To Step 2;
8. Output the best result.

16.2.8. Bat Algorithm

BA is a bio-inspired algorithm developed by Yang in 2010 and BA has been found to be very efficient. If we idealize some of the echolocation characteristics of microbats, we can develop various bat-inspired algorithms or bat algorithms. For simplicity, we now use the following approximate or idealized rules (Yang, 2010):

1. All bats use echolocation to sense distance, and they also ‘know’ the difference between food/prey and background barriers in some magical way.
2. Bats fly randomly with velocity v_i at position x_i with a fixed frequency f_{min} , varying wavelength λ and loudness A_0 to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission r $[0, 1]$, depending on the proximity of their target.
3. Although loudness can vary in many ways, we assume that the loudness varies from a large (positive) A_0 to a minimum constant value A_{min} .

For simplicity, the frequency f [0, f_{\max}], the new solutions \mathbf{x}_i^t and velocity \mathbf{v}_i^t at a specific time step t are represented by a random vector drawn from a uniform distribution (Goel *et al.*, 2013).

The basic pseudocode developed by the authors for the bat algorithm is described as follows (Yang, 2010):

Objective function $f(\mathbf{x})$, $\mathbf{x} = (x_1, \dots, x_d)^T$

Initialize the bat population \mathbf{x}_i ($i = 1, 2, \dots, n$) and \mathbf{v}_i

Define pulse frequency f_i at \mathbf{x}_i

Initialize pulse rates r_i and the loudness A_i

While ($t < \text{Max numbers of iterations}$)

Generate new solutions by adjusting frequency and updating velocities and locations/solutions [equations (1) to (3)]

if ($\text{rand} > r_i$)

Select a solution among the best solutions

Generate a local solution around the selected best solution

end if

Generate a new solutions by flying randomly

if ($\text{rand} < A_i \& f(\mathbf{x}_i) < f(\mathbf{x}_*)$)

Accept the new solutions and increase r_i and reduce A_i

end if

*Rank the bats and find the current best \mathbf{x}_**

end while

Postprocess results and visualization

The movements of each bat is associated with a velocity \mathbf{v}_i^t and location \mathbf{x}_i^t , at iteration t , in a dimensional search or solution space. Among all the bats, there exist a current best solution \mathbf{x}_* . Therefore, the above three rules can be translated into the updating equations for \mathbf{x}_i^t and velocities \mathbf{v}_i^t :

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta, \quad (7)$$

$$\mathbf{v}_i^t = \mathbf{v}_i^{t-1} + (\mathbf{x}_i^{t-1} - \mathbf{x}_*)f_i, \quad (8)$$

$$\mathbf{x}_i^t = \mathbf{x}_i^{t-1} + \mathbf{v}_i^t, \quad (9)$$

where $\beta [0, 1]$ is a random vector drawn from a uniform distribution (Rodrigues, 2013). As mentioned earlier, we can either use wavelengths or frequencies for implementation, we will use $f_{\min} = 0$ and $f_{\max} = 1$, depending on the domain size of the problem of interest. Initially, each bat is randomly assigned a frequency which is drawn uniformly from $[f_{\min} - f_{\max}]$. The loudness and pulse emission rates essentially provide a mechanism for automatic control and auto zooming into the region with promising solutions (Yang, 2013).

The loudness and pulse rates is used to provide an affective mechanism to control the exploration and exploitation and switch to exploitation stage when necessary, we have to vary the loudness A_i and the rate r_i of pulse emission during the iterations. Since the loudness usually decreases once a bat has found its prey, while the rate of pulse emission increases, the loudness can be chosen as any value of convenience, between A_{\min} and A_{\max} , assuming $A_{\min} = 0$ means that a bat has just found the prey and temporarily stop emitting any sound. With these assumptions, we have

$$A_i^{t+1} = \alpha A_i^t, \quad r_i^{t+1} = r_i^0 [1 - \exp(-\gamma^t)], \quad (10)$$

where α and γ are constants. In essence, here α is similar to the cooling factor of a cooling schedule in simulated annealing. For any $0 < \alpha < 1$ and $\gamma > 0$, we have

$$A_i^t \rightarrow 0, \quad r_i^t \rightarrow r_i^0, \quad \text{as } t \rightarrow \infty. \quad (11)$$

16.3. Applications

In this chapter, a review about the applications of the above mentioned algorithms is presented. In this case, we describe only the applications that have used swarm computing. In particular, in this paper, we have analyzed in detail only four algorithms based in swarm such as ACO, PSO, BCO, and BA algorithms. However, in this section some applications with these algorithms are shown.

16.3.1. ACO in Control and Classification

First, some applications with the ACO algorithm is presented in this section. Several applications have been developed using ACO, in this part the most relevant are shown. In the first paper, Li *et al.*, developed in their work, a one-piece flow production system with the purpose of ensuring just-in-time production. Three approaches are applied to achieve the goal: adopting straightforward schedule policies, relaxing the Talk time, and decreasing the risk of machine failures and operator mistakes. Consequently, a multi-objective design model is proposed, whose aim is to minimize cycle time, changeover count, cell load variation and the number of cells to maximize the extent to which items are completed in a cell. The fuzzy ant colony optimization (FACO) is also presented to solve the formulated problem. In FACO, the fuzzy logic controller (FLC) is used to adapt the evaporated and deposited value of pheromone trail based on the ant's fitness and pheromone trail age. Furthermore, domain knowledge of facility layout, generated based on the travel chart method, is also adaptively injected to improve the performance of FACO. The proposed method is evaluated with real-world data and experimental results demonstrate that the FACO method outperforms many other methods in efficiency, solution quality and facilitation measures (Li and Rong, 2009). Also, in (Castillo *et al.*, 2013) ACO was used to find the correct parameters for the fuzzy logic controller in a complex robotics problem; it is also a task that consumes considerable time (Alcalá-Fdez *et al.*, 2009). Because of their ability to solve complex NP problems, in these papers the authors proposed an algorithm ACO for the selection of the optimal fuzzy controller parameters (Castillo *et al.*, 2013).

Also, in (Ahmadizar and Soltanpanah, 2011) fuzzy logic is used by Ahmadizar *et al.* to improve ACO. In this paper, a solution is generated by an ant based on both pheromone trails modified by previous ants and heuristic information considered as a fuzzy set. Constructed solutions are not guaranteed to be feasible; consequently, applying an appropriate procedure, an infeasible solution is replaced by a feasible one. Then, feasible solutions are improved by a local search. The proposed approach is compared with the existing metaheuristic available in the literature. Computational results demonstrate that the approach serves to be a better performance for large problems.

In 2011, a fuzzy classification system based on Ant Colony Optimization for diabetes disease diagnosis is proposed by Ganji *et al.* The aim of this paper is to use an Ant

Colony-based classification system to extract a set of fuzzy rules for diagnosis of diabetes disease, named FCS-ANTMINER. This paper describes some recent methods and describes a new and efficient approach that leads to considerable results for diabetes disease classification problem. FCS-ANTMINER has new characteristics that make it different from the existing methods that have utilized the ACO for classification tasks. The obtained classification accuracy is 84.24% which reveals that FCS-ANTMINER outperforms several famous and recent methods in classification accuracy for diabetes disease diagnosis (Ganji and Abadeh, 2011). On the other hand, Chang *et al.* developed a fuzzy sliding-mode control for ball and beam system with FACO. This paper mainly addresses the balance control of a ball and beam system, where a pair of decoupled fuzzy sliding-mode controllers (DFSMCs) is proposed. The DFSMC has the advantage of reducing the design complexity, in which the coupling dynamics of the state-error dynamics are considered as disturbance terms. Stability analysis of the ball and beam system with DFSMCs is also discussed in detail. To further improve the control performance, an improved ACO is proposed to optimize the controller parameters. The proposed ACO algorithm has the enhanced capability of fuzzy pheromone updating and adaptive parameter tuning. The proposed ACO-optimized scheme is utilized to tune the parameters of the fuzzy sliding-mode controllers for a real ball-and-beam system. Compared to some conventional ACO algorithms, simulation and experimental results all indicate that the proposed scheme can provide better performance in the aspect of convergence rate and accuracy (Chang *et al.*, 2012).

16.3.2. PSO in Neural Networks, Mathematical Functions and Control

In the first approach of PSO, particles could be a swarm of neural networks attempting to find the lowest error on some classification or regression task. See Particle Swarm Optimization of Neural Network Architectures and Weights (de Lima and Ludermir, 2011).

Also in (Valdez *et al.*, 2010), an Evolutionary method combining Particle Swarm Optimization and GAs using fuzzy logic for parameter adaptation and aggregation is used to optimize the architecture of the modular neural network for face recognition. In this approach, the authors proposed a combination of Genetic Algorithms and PSO to improve the performance of the Neural Network.

Also in (Argha *et al.*, 2013) the adaptation of network weights using PSO was proposed as a mechanism to improve the performance of artificial neural network (ANN) in classification of IRIS dataset. Classification is a machine learning technique used to predict group membership for data instances. To simplify, the problem of classification neural networks are being introduced. This paper focuses on IRIS plant classification using Neural Network. The problem concerns the identification of IRIS plant species on the basis of plant attribute measurements. Classification of IRIS dataset would be

discovering patterns from examining petal and sepal size of the IRIS plant and how the prediction was made from analyzing the pattern to form the class of IRIS plant.

In 2013, Melin *et al.* propose an improvement to the convergence and diversity of the swarm in PSO using fuzzy logic. Simulation results show that the proposed approach improves the performance of traditional PSO. First, benchmark mathematical functions are used to illustrate the feasibility of the proposed approach. Then a set of classification problems are also used to show the potential applicability of fuzzy logic in dynamic parameter adaptation of PSO.

Chiou *et al.* developed a novel method for determining the optimal fuzzy PID-controller parameters of active automobile suspension system using the PSO algorithm. This paper demonstrated in detail how to help the PSO with a Q-learning cooperation method to search efficiently for the optimal fuzzy-PID controller parameters of a suspension system. The design of a fuzzy system can be formulated as a search problem in high-dimensional space where each point represents a rule set, membership functions, and the corresponding system's behavior. In order to avoid obtaining the local optimum solution, a pure PSO global exploration method to search for fuzzy-PID parameter (Li and Wu, 2011) was adopted.

16.3.3. BCO in Traveling Salesman Person and Other Applications

The main goal of Lučić and Teodorović(Lučić and Teodorović, 2001, 2002, 2003a, 2003b) research was not to develop a new heuristic algorithm for the traveling salesman problem but to explore possible applications of Swarm Intelligence (particularly collective bee intelligence) in solving complex engineering and control problems. The TSP is only an illustrative example, which shows the characteristics of the proposed concept. Lučić and Teodorović tested the BCO approach on a large number of numerical examples.

Also, BCO has been successfully tested (Teodorović *et al.*, 2006) in the case of the Routing and Wavelength Assignment (RWA) in All-Optical Networks. This problem is, by its nature similar to the traffic assignment problem. The results achieved, as well as experience gained when solving the RWA problem could be used in the future research of the traffic assignment problem. The RWA problem is described as follows: every pair of nodes in optical networks is characterized by a number of requested connections. The total number of established connections in the network depends on the RWA procedure. RWA problem in all-optical networks could be defined in the following way: Assign a path through the network and a wavelength on that path for each considered connection between a pair of nodes in such a way to maximize the total number of established connections in the network.

16.3.4. BA in Several Applications

Musikapun and Pongcharoen (2012) solved multistage, multi-machine, multi-product

scheduling problems using bat algorithm, and they solved a class of nondeterministic polynomial time (NP) hard problems with a detailed parametric study. They also implied that the performance can be further improved by about 8.4% using optimal set of parameters.

Yang (2013) used the bat algorithm to study topological shape optimization in microelectronic applications so that materials of different thermal properties can be placed in such a way that the heat transfer is most efficient under stringent constraints. It can also be applied to carry out parameter estimation as an inverse problem. If an inverse problem can be properly formulated, then bat algorithm can provide better results than least-squares methods and regularization methods.

Lin *et al.* (2012) presented a chaotic Levy flight bat algorithm to estimate parameters in nonlinear dynamic biological systems, which proved the effectiveness of the proposed algorithm.

Komarasamy and Wahi (2012) studied k-means clustering using bat algorithm and they concluded that the combination of both k-means and BA can achieve higher efficiency and thus performs better than other algorithms.

Khan and Sahai (2012a) presented a study of a clustering problem for office workplaces using a fuzzy bat algorithm. Khan and Sahai (2012a) also presented a comparison study of bat algorithm with PSO, GA, and other algorithms in the context for e-learning, and thus suggested that bat algorithm has clearly some advantages over other algorithms.

Then, they (Khan and Sahai, 2012a) also presented a study of clustering problems using bat algorithm and its extension as a bi-sonar optimization variant with good results.

16.4. Conclusions

After making an exhaustive study of the different optimization algorithms inspired in swarms, we can conclude the importance of these algorithms and we can see that they are capable to improve the performance of many applications, which is usually required in complex optimization problems. For example, BCO is a metaheuristic motivated by foraging behavior of honeybees. It represents general algorithmic framework that can be applied to various optimization problems in management, engineering, and control. ACO takes inspiration from the shortest path searching behavior of various ant species. PSO developed by Eberhart and Kennedy in 1995, inspired by the social behavior of bird flocking or fish schooling also is able for solving many complex optimization problems. Finally BA is the more recent method, inspired in the behavior of the micro bats, however is able to achieve very good results as the methods based in populations. In this paper, we only analyzed the most relevant proposed methods in the last years; however, there are other works with other algorithms where the researchers can find other similar metaheuristics. Therefore, in this study we focused on the methods based in swarm intelligence. Recently, many researchers have been applying swarm intelligence in for solving complex applications and obtain better results than with conventional methods. Also, we can observe in this review that the fuzzy systems can be used to improve methods and not only work with control systems for example or other applications. The swarm intelligence is nowadays used by researchers to achieve the best results in the algorithms and obtain better results than the other techniques.

Acknowledgment

We would like to express our gratitude to CONACYT, Tijuana Institute of Technology for the facilities and resources granted for the development of this research.

References

- Ahmadizar, F. and Soltanpanah, H. (2011). Reliability optimization of a series system with multiple-choice and budget constraints using an efficient ant colony approach. *Expert Syst. Appl.*, 38(4), pp. 3640–3646.
- Alcalá-Fdez, R., Alcalá, MJ. and Gacto, F. (2009). Herrera, learning the membership function contexts forming fuzzy association rules by using genetic algorithms. *Fuzzy Sets Syst.*, 160(7), pp. 905–921.
- Angeline P. J. (1998). Using selection to improve particle swarm optimization. *Proc. 1998 IEEE World Congress on Computational Intelligence*, Anchorage, Alaska, IEEE, pp. 84–89.
- Angeline P. J. (1998). Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences, *Evolutionary Programming VII*, Lecture Notes in Computer Science 1447, Springer, pp. 601–610.
- Argha R., Diptam D. and Kaustav C. Training artificial neural network using particle swarm optimization 3(3), *Int. J. of Adv. Res. Comp. Sci. and Soft. Eng. Res. Paper*. Retrieved from www.ijarcsse.com.
- Beni, G. (1988). The concept of cellular robotic system. *Proc. of the 1988 IEEE Int. Symp. Intell. Control*. Los Alamitos, CA: IEEE Computer Society Press, pp. 57–62.
- Beni, G. and Wang, J. (1989). Swarm intelligence. *Proc. of the Seventh Ann. Meeting of the Robotics Society of Japan*. RSJ Press, Tokyo, pp. 425–428.
- Beni, G. and Hackwood, S. (1992). Stationary waves in cyclic swarms. *Proc. of the 1992 Int. Symp. on Intelligent Control*. IEEE Computer Society Press, Los Alamitos, CA, pp. 234–242.
- Bonabeau, E., Dorigo, M. and Theraulaz, G. (1997). *Swarm Intelligence*. Oxford University Press, Oxford.
- Bullnheimer, B., Kotsis, G. and Strauss, C. (1997). Parallelization strategies for the ant system. In Toraldo, G., Murli, A. and Pardalos, P. (eds.), *Kluwer Series on Applied Optimization*, pp. 87–100.
- Castillo, O. and Melin, P. (2002). Hybrid intelligent systems for time series prediction using neural networks, fuzzy logic, and fractal theory. *Neural Networks, IEEE Transactions* 13(6), pp. 1395–1408.
- Castillo, O., Neyoy, H., Soria, J., García, M. and Valdez, F. (2013). Dynamic fuzzy logic parameter tuning for ACO and its application in the fuzzy logic control of an autonomous mobile robot. *Int. J. Adv. Robot Syst.*, 10(51), pp. 1–10.
- Chang, Y. H., Chang, C. W., Tao, C. W., Lin, H. W. and Taur, J. S. (2012). Fuzzy sliding-mode control for ball and beam system with fuzzy ant colony optimization. *Expert Systems with Applications*, 39(3), pp. 3624–3633.
- Colorni, A., Dorigo, M. and Maniezzo, V. (1992a). Distributed optimization by ant colonies. In Varela, F. J. and Bourgine, P. (eds.), *Proceedings of the First European Conference on Artificial Life*. Cambridge, MA: MIT Press, pp. 134–142.
- de Lima, N. F. and Ludermir, T. B. Frankenstein PSO applied to neural network weights and architectures, *Evolutionary Computation (CEC), 2011 IEEE Congress*. pp. 2452–2456.
- Dorigo. M. (1992). *Optimization, Learning and Natural Algorithms*. Ph.D. thesis, Politecnico di Milan, Milano, Italy.
- Dorigo, M. (1994). Learning by probabilistic boolean networks. *Proc. IEEE Int. Conf. Neural Netw.*, pp. 887–891.
- Dorigo, M., Bonabeau, E. and Theraulaz, G. (2000). Ant algorithms and stigmergy. *Future Generation Computer Systems*, 16(9), pp. 851–871.
- Dorigo, M. and Di Caro, G. (1999). Ant colony optimization: A new meta-heuristic. *Proc. of the IEEE Congress on Evolutionary Computation*, 2, p. 1477.
- Dorigo, M. and Di Caro, G. (1999). The ant colony optimization meta-heuristic. In Corne, D., Dorigo, M. and F. Glover, (eds.), *New Ideas in Optimization*, pp. 11–32. London: McGraw-Hill.
- Dorigo M. and Gambardella, L. (1996). A study of some properties of Ant-Q. *Proc. of the Fourth Int. Conf. on Parallel Problem Solving from Nature*, pp. 656–665.
- Dorigo, M. and Gambardella, L. M. Ant colonies for the travelling salesman problem. *Biosystems*, 43(2), pp. 73–81.
- Eberhart, R. C. and Kennedy, (1995). A new optimizer using particle swarm theory, *Proc. Sixth Int. Symposium on Micro Machine and Human Science*, Nagoya, Japan, pp. 33–43.
- Ganji, M. F. and Abadeh, M. S. (2011). A fuzzy classification system based on Ant Colony Optimization for diabetes disease diagnosis. *Expert Systems with Applications*, 38(12), pp. 14650–14659.

- Geem, Z. W. (2008). Novel derivative of harmony search algorithm for discrete design variables. *Appl. Math. Comput.*, 199(1), pp. 223–230.
- Glover, F. (1989). Tabu Search—Part I, ORSA. *Journal on Computing*, 1, No. 3, pp. 190–206. First comprehensive description of tabu search.
- Goel N., Gupta D. and Goel S. (2013). *Performance of Firefly and Bat Algorithm for Unconstrained Optimization Problems*, Department of Computer Science Maharaja Surajmal Institute of Technology GGSIP University C-4, Janakpuri, New Delhi, India.
- Holland, J. H. (1975). Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. Ann Arbor, MI: University of Michigan Press.
- Karaboga, D. and Akay, B. (2009). A survey: Algorithms simulating bee swarm intelligence. *Artificial Intelligence Rev.* 31(1), pp. 68–85.
- Kennedy, J. and Eberhart, R. C. (1995). Particle swarm optimization. *Proc. of IEEE Int. Conf. on Neural Networks*, Piscataway, NJ, pp. 1942–1948.
- Kennedy, J. and Eberhart, R. C. Particle swarm optimization. *Proc. IEEE Int. Conf. on Neural Networks*, Piscataway, NJ, pp. 1942–1948.
- Khan, K. and Sahai, A. (2012a). A comparison of BA, GA, PSO, BP and LM for training feed forward neural networks in e-learning context, *Int. J. Intelligent Systems and Applications (IJISA)*, 4, No. 7, pp. 23–29.
- Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598), pp. 671–680.
- Komarasamy, G. and Wahi, A. (2012). An optimized K-means clustering technique using bat algorithm, *European J. Scientific Research*, 84, No. 2, pp. 263–273.
- Lin, J. H., Chou, C. W., Yang, C. H. and Tsai, H. L. (2012). A chaotic levy flight bat algorithm for parameter estimation in nonlinear dynamic biological systems, *J. Computer and Info. Tech.* 2, No. 2, pp. 56–63.
- Li, S. G. and Rong, Y. L. (2009). The reliable design of one-piece flow production system using fuzzy ant colony optimization, *Computers & Operations Research*, 36(5), pp. 1656–1663.
- Li, C. and Wu, T. (2011). Adaptive fuzzy approach to function approximation with PSO and RLSE. *Expert Systems with Applications*, 38, pp. 13266–13273.
- Lučić, P. and Teodorović, D. (2001). Bee system: Modeling combinatorial optimization transportation engineering problems by swarm intelligence. In *Preprints of the RISTAN IV Triennial Symposium on Transportation Analysis*, São Miguel, Azores Islands, Portugal, pp. 441–445.
- Lučić, P. and Teodorović, D.: Transportation modeling: An artificial life approach. *Proc. of the 14th IEEE 'Int. Conf. on Tools with Artificial Intelligence*, Washington, DC, pp. 216–223.
- Lučić, P. and Teodorović, D. (2003a). Computing with bees: Attacking complex transportation engineering problems. *Int. J. Artif. Intell. T.* 12, pp. 375–394.
- Lučić, P. and Teodorović, D. (2003b). Vehicle routing problem with uncertain demand at nodes: The bee system and fuzzy logic approach. Verdegay, J. L. (ed.), *Fuzzy Sets in Optimization*. Heidelberg, Berlin: Springer-Verlag, pp. 67–82.
- Marković, G., Teodorović, D. and Aćimović, Raspopović, V. (2007). Routing and wavelength assignment in all-optical networks based on the bee colony optimization. *AI Commun.* 20, pp. 273–285.
- Melin, P., Olivas, F., Castillo, O., Valdez, F., Soria, J. and Valdez, M. (2013). Optimal design of fuzzy classification systems using PSO with dynamic parameter adaptation through fuzzy logic. *Expert Syst. Appl.* 40(8): pp. 3196–3206.
- Musikapun, P. and Pongcharoen, P. (2012). Solving multi-stage multi-machine multi-product scheduling problem using bat algorithm. *2nd Int. Conf. on Management and Artificial Intelligence (IPEDR)*, 35, IACSIT Press, Singapore, pp. 98–102.
- Ocenasek, J. and Schwarz, J. (2002). Estimation of distribution algorithm for mixed continuous-discrete optimization problems. In *Proc. of the 2nd Euro-International Symposium on Computational Intelligence*, IOS Press, Amsterdam, the Netherlands, (pp. 227–232).

- Rashedi, E., Nezamabadi-pour, H. and Saryazdi, S. (2009). GSA: A Gravitational Search Algorithm. *Inf. Sci.*, 179(13), pp. 2232–2248.
- Reynolds, R. G. (1994). An introduction to cultural algorithms, *Proc. of the 3rd Ann. Conf. on Evolutionary Programming*, World Scientific Publishing, RiverEdge NJ, pp. 131–139.
- Rodrigues, D., Pereira, L., Nakamura, R., Costa, K., Yang, X., Souza, A. and Papa, J. P. (2013). A wrapper approach for feature selection based on Bat Algorithm and Optimum-Path Forest, Department of Computing, Universidade Estadual Paulista, Bauru, Brazil.
- Shah-Hosseini, H. (2009). The intelligent water drops algorithm: A nature-inspired swarm-based optimisation algorithm, *Int. J. Bio-Inspired Computation*, 1(1/2), pp. 71–79.
- Stützle, T. (1997a). An ant approach to the flow shop problem. Technical report AIDA-97-07, FG Intellektik, FB Informatik, TU Darmstadt, Germany.
- Stützle, T. (1997b). MAX-MIN Ant System for the quadratic assignment problem. Technical report AIDA-97-4, FG Intellektik, FB Informatik, TU Darmstadt, Germany.
- Tang, W. J., Wu, Q. H. and Saunders, J. R. (2006). Bacterial foraging algorithm for dynamic environments. *IEEE Congress on Evolutionary Computation Sheraton Vancouver Wall Centre Hotel*, Vancouver, BC, Canada, July 16–21.
- Teodorović, D. (2003b). Transport modeling by multi-agent systems: A swarm intelligence approach. *Transport. Plan. Techn.* 26, pp. 289–312.
- Teodorovic, D. and Dell'orco, M. (2005). Bee colony optimization—A cooperative learning approach to complex transportation problems, *Advanced OR and AI Methods in Transportation*, pp. 51–60.
- Teodorović, D. and Dell'Orco, M. (2005). Bee colony optimization—A cooperative learning approach to complex transportation problems. In advanced OR and AI methods in transportation. *Proc. of the 10th Meeting of the EURO Working Group on Transportation*, Poznan, Poland, pp. 51–60.
- Teodorović, D., Lučić, P., Marković, G. and Dell' Orco, M. (2006). Bee colony optimization: Principles and applications. In Reljin, B. and Stanković, S. (eds.), *Proc. of the Eight Seminar on Neural Network Applications in Electrical Engineering—NEUREL 2006*, University of Belgrade, Belgrade, pp. 151–156.
- Teodorović, D. and Šelmić, M. (2007). The BCO algorithm for the p median problem. In *Proc. of the XXXIV Serbian Operations Research Conf.* Zlatibor, Serbia.
- Valdez, F., Melin, P. and Castillo, O. (2010). Evolutionary method combining Particle Swarm Optimisation and Genetic Algorithms using fuzzy logic for parameter adaptation and aggregation: The case neural network optimisation for face recognition. *IJAISC*, 2(1/2), pp. 77–102.
- Yang, X. (2010). *A New Metaheuristic Bat-Inspired Algorithm*. Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK.
- Yang, X. S. (2010). A new metaheuristic bat-inspired algorithm. In Gonzalez, J. R. (ed.), *Nature Inspired Cooperative Strategies for Optimization (NISCO 2010). Studies in Computational Intelligence*. Berlin: Springer, pp. 65–74.
- Yang, X. (2013). *Bat Algorithm: Literature Review and Applications*, School of Science and Technology, Middlesex University, The Burroughs, London NW4 4BT, United Kingdom.

Chapter 17

Memetic Algorithms

Qiangfu Zhao and Yong Liu

Roughly speaking, genes carry information for constructing a body, in which they usually work in groups to construct a meaningful body. Similarly, memes carry information for constructing a culture, in which many memes usually work together to construct a meaningful culture. Although in real world genes and memes are realized in different forms (e.g., DNA for genes and language for memes), in evolutionary computation (EC), both of them can be represented using strings defined on some alphabetic set. Thus, in EC we can study genetic evolution and memetic evolution in the same way. We can also combine them together to improve the efficiency and efficacy of the evolutionary search process. *Memetic algorithms (MAs) are search algorithms obtained by combining genetic evolution and memetic evolution.* In MAs, genetic evolution is often used for global search and memetic evolution is used to find the best local search strategy. So far, a great number of MAs have been proposed in the literature. In this chapter, we first define some fundamental concepts related to memetic evolution, and then introduce several templates that may cover most existing MAs. For each template, we provide a pseudo code, so that readers can understand the algorithms easily, and can develop programs to solve their own problems.

17.1. Introduction

Heuristic algorithms often use some *a priori* domain knowledge called *heuristics* to determine the next search point, given the current one. The main purpose of using heuristics is to improve search efficiency, although global optimal solutions may not be guaranteed. To increase the opportunity of obtaining the global optimal solutions, some higher level heuristics are often required. Search algorithms using multiple levels of heuristics are called *metaheuristic algorithms*. To improve the efficiency and the efficacy of search, different heuristics can be hybridized without considering levels. These kinds of hybridized algorithms in general are called *hyper-heuristic algorithms*. Hyper-heuristics are more general than metaheuristics.

In hyper-heuristic algorithms, we often use some heuristics for local search, and some other heuristics for controlling the whole search process, so that the global optimal solution(s) can be obtained with a higher probability (Miller *et al.*, 1993). A simple example is to hybridize a *population based* algorithm with an *individual based* one. The former is used for global search, and the latter for local adaptation. This hybridization, although simple, often leads to very good results for solving hard problems (Krasnogor and Smith, 2005; Ong *et al.*, 2006; Chen *et al.*, 2011).

Noting that some good local search heuristics or strategies can be shared by many individuals, these heuristics together can be considered as the “culture” of a population. It is for this reason that Moscato has named hyper-heuristic algorithms as *memetic algorithms* (MAs) (Moscato, 1989). Here, the term “meme” was produced by Dawkins to mean “*a unit of imitation in cultural transmission which in some aspects is analogous to the gene*” (Dawkins, 1976). In MAs, memes are often considered as local search strategies. The common goal of MAs is to incorporate some good memes into a global search procedure, so that we can find the global optimal solution(s) more efficiently and effectively.

In recent years, a great number of MAs have been proposed in the literature, and many survey papers have been published in several good journals (Krasnogor and Smith, 2005; Ong *et al.*, 2006; Chen *et al.*, 2011). Although authors of these papers have tried different ways to summarize and classify MAs, it is still difficult for the beginners to understand the relationships between different MAs. The goal of this chapter is to summarize different MAs in a more systematic way, so that readers can study, use, and propose their own MAs easily.

The rest of the chapter is organized as follows. Section 17.2 provides a brief introduction to memetic evolution theory, so that readers can understand what a meme really is, and what an MA should be. In Sections 17.3 and 17.4, we investigate, respectively, some *adaptive* MAs and *evolutionary* MAs, and provide pseudo codes for them. Section 17.5 is the conclusion of this chapter.

17.2. Meme and Memetic Evolution

It is known that *Darwin's theory of evolution* is universal, and is not limited to evolution of *biological lives* on the earth. In fact, the universal Darwinism is also applicable to evolution of cultures (Hodgson, 2005). For the latter, Dawkins introduced the term *meme* to represent “a unit of imitation in cultural transmission which in some aspects is analogous to the gene” (Dawkins, 1976). Memes, according to Dawkins, are a kind of *replicators* that can evolve. That is, they have the three fundamental properties for evolution, namely *variation, selection, and retention*. In this section, we try to define meme and some related concepts more clearly, and build a foundation for the following sections.

17.2.1. Meme, Memotype, and Memeplex

Some people argue that unlike genes, we still cannot define the *units or building blocks* for cultures, and thus the meme metaphor may not be correct (Blackmore, 1999). In fact, when talking about memes, Dawkins did not distinguish meme, memotype, and memeplex clearly. This is the main reason why people could not understand the meme theory well. It is a pity that many MA researchers actually do not understand what a meme really is.

Similar to gene, *genotype*, and *phenotype*, we have meme, *memotype*, and *memeplex* (short for *meme complex*). Roughly speaking, a group of memes form a memotype; and a memotype defines a memeplex. For example, to make an origami airplane, a kindergarten teacher may tell the children how to do, step by step, while she makes a demonstration. The ways she selects the paper, folds the corners, folds the edges, etc., can be seen and imitated by the children. Children in group A may try to remember what their teacher says, memorize the instructions, and then make the origami airplane in *the same way*. On the other hand, children in group B may imitate their teacher step by step, try to describe the process using their own languages and memorize, and then make the paper airplanes in *similar ways*. Here, the ways for selecting the paper, folding the corners, folding the edges, etc., form the memeplex; the instructions form the memotype; and the words used in the instructions are memes.

From the origami example we can understand that memes can be defined as *words* or *phrases* in different languages, and be used as the building blocks of different cultures. It is not proper to consider the whole process for doing something as a single meme. Rather, this whole process should be considered as a memeplex. In general, we may define memes as *concepts* represented in words (or Chinese characters) or phrases. Some concept can be very complicated and must be defined using many lower level concepts. In such cases we may define the corresponding meme using its *name or label* (e.g., “*meme*” is the name of the main concept described in this paragraph). Similar to genes, memes contain information for constructing different memplexes (e.g., ideas, rules, instructions, and methods).

Similar to genotype, memotype represents *a group of memes*. Speaking more precisely, it is memotype, not memeplex, which is transmitted from brain to brain. A memotype is the seed or egg of a memeplex. Once it is accepted by a brain, it can grow up using resources of the brain, and become a memeplex. We may classify the memotypes into two categories, namely *strictly described* (SD) memotypes, and *weakly described* (WD) ones. An SD memotype usually describes the memeplex precisely. It can be passed from one brain to another with few variations, and the memeplex can be almost the same as the memotype. On the other hand, a WD memotype is usually short, and may vary greatly when it is passed to other brains. A WD memotype, once accepted by a brain, may grow up in the brain, and generate a big memeplex later.

In the case the memotype is how to ride a bicycle for instance, it can be a very simple instruction like “*just practice and you will do it*”. In such cases, some *tacit knowledge* (Polanyi, 1966) must be learned through practice to compensate the instruction. The same thing happens in genotype based phenotype construction. It is reasonable to consider that information contained in the genotype alone is not enough to construct a phenotype (e.g., a human body) if the egg is not put into a proper environment. That is, some outside information or materials are supposed *to be there* for constructing the phenotype. Similarly, when a person imitates a memeplex, not all information is memorized in the brain. A *seed* or an *egg* will be sufficient, and this seed is nothing but the memotype. The memotype will utilize resources of the brain to construct a memeplex, with variations.

Note that there are memeplexes that cannot be described by *speaking languages*. Examples include music, fashions, hair styles, gestures, videos, etc. However, the memotypes of these memeplexes can also be described using a set of *symbols or attributes*. In general, the memotypes can be represented in strings defined on some alphabetic set, which are nothing but *languages in the sense of computer science*.

17.2.2. Fitness of Memes

Memes can transmit from one brain to another via *imitation* or learning, and their different versions can survive in different brains for a relatively long period. For the memes to evolve, it is necessary to define their goodness or fitness first. Intuitively, good memes can form good memotypes, good memotypes can form good memeplexes, and many good memeplexes together can form a good *mental brain* (or *mind*). The memeplexes can be defined as *agents*, and the mental brain is actually a *society of agents* (Minsky, 1986). In turn, a person with a good mental brain can be very clever, and can be more successful than others. A successful person has more chances to be imitated or learned by others. Through imitation or learning people wish to be equally or more successful. Thus, good memes can be passed from brain to brain easily, and can survive for a long time.

After all, what is the fitness of a meme? Like a gene, a single meme cannot be evaluated directly. A meme can become meaningful only when it is put together with other

memes and form a memotype; and only if this memotype can obtain some brain resources and become a memeplex. Only when a memeplex is meaningful, the corresponding memotype can be accepted more easily by many brains. Here, the term “meaningful” should be defined based on the principle of *survival of the fittest*. Normally, we accept some memotypes if we *believe* (through observations) that the corresponding memeplexes are useful. However, what we believe may not be true or correct. In the process of memetic evolution, some memotypes may become so *clever* that they can be easily passed and accepted by many brains and can grow up inside the brains gradually later. These memotypes may not be useful or even harmful to our humans. But we may accept them before knowing their real values.

Thus, we cannot define the fitness of the memes based on human belief. The memes or memotypes are selected or selected against based only on their own fitness for survival. Similar to evaluation of a gene, the fitness of a meme is evaluated indirectly based on the fitness of the memotypes containing this meme. The memotypes in turn are evaluated by the fitness of the corresponding memeplexes. In fact, any method used for evaluating genes based on genotypes and phenotypes can be used to evaluate memes based on memotypes and memeplexes.

It is also worthwhile to note that the fitness of a meme does not depend, at least does not only depend, on the fitness of the human body. People with weak bodies may have strong *mental brains*, which are constructed by strong memeplexes. However, clever memes do seek for more chances to immigrate to brains with strong bodies so that they can have more opportunity to influence other people and thus have more chances to spread. Clever memes must also govern the brains in which they live to keep the bodies healthy and strong, so that they can have longer time to seek for chances to spread.

17.2.3. Process of Memetic Evolution

In the above discussions, we have already mentioned here and there something related to memetic evolution. For example, through transmission a memotype can find more resources to survive by joining other memotypes (*recombination or crossover*). When a memotype is transmitted to a brain, agents (or memeplexes) already living there often do not accept this new comer, at least they will not accept the new comer as it is. Existing agents will make a collective decision, force the new comer to make some changes (*mutation*), and then accept it. As a result, the same memotype (and thus the corresponding memeplex) can have different variations in different brains. Memotypes that cannot find any place to go will die and disappear (*selected against*). Even this chapter, as a memotype, will be selectively accepted by the readers—some parts may transmit from my brain to this book and from this book to the brains of some readers, and others will be forgotten.

There are many memeplexes, and they can exist in different forms. Some *are living* in

the human brains, some *are sleeping* in the libraries, and some *are wandering* around in the internet. Each person accepts (via imitation or learning) some memeplexes (exist in the brains or other places) that he/she believes useful, recombines them with the memeplexes already in the brain, produces variations, and posts them to some mass media (e.g., books, papers, reports, etc.) if he/she believes that the new memeplexes are useful to other people (and this paragraph is an example of a new memeplex). Thus, we human beings are one of the media for receiving, carrying, reproducing, amplifying, and distributing the memeplexes. In certain sense, we are just the computing machines for the memeplexes to evolve. There is no evidence that memetic evolution is for human. In future, the memeplexes may find better computing machines (e.g., the cyber-space) to evolve and forget completely the old machines that they are using now.

17.3. Adaptive MAs

In this section, we introduce several basic MAs. Although these MAs do not *evolve* the memes, they are very useful for solving many practical problems. Before starting, let us first review briefly the standard genetic algorithm (SGA), and define some terminologies that will be used throughout this chapter.

17.3.1. Standard Genetic Algorithm

SGA (Goldberg, 1989) is used as a basis algorithm of this chapter; although discussions given here can be extended easily to genetic programming (Koza, 1992; 2002), evolution strategies (Beyer and Schwefel, 2002), evolutionary programming (Fogel *et al.*, 1996), and other population based metaheuristic algorithms.

Program-0 is the pseudo code of SGA. In this *program*, we consider each individual in a population P as an *agent*, which is defined as a *structure* containing several elements. In this chapter, whenever we provide a pseudo-code, we use a style *similar to* C language, and try to use full English words as much as possible to make the code more understandable. We use a semicolon to terminate a line, and a double slash “//” to make some short comments. In the program, “**DefineType**” is used to define a new type, the type “**Genotype**” is usually a binary bit string, the type “**Fitness**” is a real number, and the population P is an array of type **Agent**. P contains **NumberAgent** individuals. As in c language, we use $x.g$ and $x.f$ to denote the genotype and the fitness, respectively, of an individual x ; and use $P[i]$ to denote the i th element of P .

Program-0: SGA

```
DefineType Agent {  
    Genotype: g; // Genotype of the agent.  
    Fitness : f; // Fitness of the agent.  
};  
Agent P[NumberAgent];  
Begin  
    P=Initialize(); // Initialize population.  
    For t=1 to NumberGeneration  
        For i=1 to NumberAgent  
            p=Decoding(P[i].g); // Find phenotype.
```

```

P[i].f=Evaluation(p); // Find fitness.

End

If (Terminating condition) break;

P=Reproduction(P); // Update population.

End

End

```

Note that in SGA, it is not necessary for each individual to have a phenotype. The phenotype can be generated and stored in a *working memory* which is shared by all individuals. The genotype of each individual is usually initialized at random. Although some algorithm can also be used to find a better starting point, we do not consider this kind of hybridization in SGA. During evolution, the phenotype of each individual is first decoded (reconstructed) from the genotype, and is then evaluated. Different methods for evaluation can be used for different applications. The *terminating condition* also depends on applications, but usually we may just check if the fitness value of the current best individual is already good enough (e.g., higher than a pre-specified value) or not; or the number of *generations* reaches to a given value (i.e., NumberGeneration) or not.

To reproduce a new population, we first select some parent individuals, then produce new individuals (offspring) by recombining (crossover) these parents, and finally mutate the new individuals. The new individuals together form the population for the next generation. That is, reproduction can be defined as a product of three operators as follows:

$$P(t + 1) = \text{Reproduction}(P(t)) = \text{Mutation}(\text{Crossover}(\text{Selection}(P(t))))$$

In this chapter, we suppose that the population size does not change during evolution. For practical applications, however, we may consider the population as a hybrid of the medium-term and long-term memories used in Tabu search (TS) for intensification (exploitation) and diversification (exploration) (Glover, 1986; 1989; 1990), and use some other meta-heuristics to adjust the population size. We will not discuss this issue in detail here.

Program-1: SHA

```

Begin

P=Initialize(); // Initialize population.

For t=1 to NumberGeneration

```

```

For i=1 to NumberAgent

    p=Decoding(P[i].g); // Find phenotype.

    P[i].f=Evaluation(p); // Find fitness.

End

If (Terminating condition) break;

P=Reproduction(P);

End

x=FindBest(P); // find the best individual.

p=Decoding(x.g); // Find phenotype.

p=LocalSearch(p); // Fine tune phenotype.

End

```

17.3.2. Sequential Hybrid Algorithm (SHA)

It is known that population-based algorithms like SGA are good for global search, but weak for local fine tuning. That is, even if we have already found the rough position of the global optimal solution x^* , it may take time to find the exact position of x^* . A simple idea to solve this problem is to use SGA first and then use some local search algorithm to fine-tune the solution. Many algorithms have been proposed based on this idea (Thangaraj *et al.*, 2011). In this chapter, we call these algorithms SHAs. Program-1 is the pseudo code of an SHA.

The local search algorithm used in SHA can be TS (Glover, 1989), simulated annealing (SA) (Goffe *et al.*, 1994), or any algorithms proposed in the literature. If the problem is to find a multilayer neural network, we may also use the back propagation (BP) algorithm or any of its improved versions for local search (Schaffer *et al.*, 1992). In addition, we may use particle swarm optimization (PSO) (Eberhart and Shi, 2001) or other population-based *Lamarckian* algorithms for fine tuning. In this case, we may just use the whole population P obtained by SGA as the initial population for further search (Thangaraj *et al.*, 2011).

17.3.3. Two-Layered Hybrid Algorithms (TLHAs)

Instead of fine tuning after global search, we may embed the local search into the global search algorithm. This way, we can obtain hybrid algorithms with two layers, namely the global search layer and the local search layer. In this chapter, we call these TLHAs. Program-2 is the pseudo code of a TLHA.

In Program-2, we have assumed that TLHA is a *Baldwinian evolution*. That is, the new phenotype acquired through local search is used only for evaluation. This can be considered as a *default* because Baldwinian evolution is usually better than *Lamarckian* in the sense that better results can be obtained in changing environments. If the environment does not change or does not change quickly, we may also use Lamarckian evolution.

To change the algorithm to Lamarckian, we can introduce an *encoder* that converts the phenotype obtained via local search to a genotype, and then assign the new genotype to the individual under concern. In practice, we may accept the local search result *partially*, to make a balance between full Baldwinian and full Lamarckian evolutions. For example, we may use some of the intermediate results found during local search, encode and then assign the results to the genotype of the current individual.

There is an important difference between local searches in SHA and TLHA. In the former, local search is usually *full*, that is, it should be conducted until convergence; while in the latter local search can be *partial*, that is, we may just run local search for several iterations without waiting for convergence. In fact, full local search for TLHA can be harmful because the evolution process may fall into local minimum easily. In addition, using TLHA we may not have to employ complex algorithms for local search. For example, to solve the traveling salesman problem (TSP) or other combinatorial problems, simple local searches like swapping and visiting neighboring nodes might be sufficient.

Program-2: TLHA

```
Begin
    P=Initialize(); // Initialize population.

    For t=1 to NumberGeneration
        For i=1 to NumberAgent
            P=Decoding(P[i].g); // Find phenotype.

            P=LocalSearch(p); // Update phenotype.

            P[i].f=Evaluation(p); // Find fitness.

        // Use the following line for Lamarckian Evolution
        /* P[i].g=Encoding(p); */

        End

        If (Terminating condition) break;

        P=Reproduction(P);
```

End

End

In the literature, TLHA is often considered as the standard or canonical MA (Ong *et al.*, 2006), although in this chapter we do not adopt this name. We may define the local search strategy used for fine tuning as a memeplex. This memeplex is the culture of the whole population, and is shared by all individuals. However, this memeplex is completely pre-defined, rather than found through evolution. Thus, we can define TLHA as a zeroth-order MA. In fact, SHA is also a zeroth-order MA because the memeplex is pre-defined. In this chapter, we define the *order* of an MA according to its *degree of evolution*. That is, if an MA does not evolve the memeplexes, it is a zeroth-order MA; if the MA uses one or two factors (among variation, selection, and retention) of evolution, it is a first-order MA; and if the MA possesses all features of memetic evolution, it is a second-order or true MA.

17.3.4. Adaptive Two-Layered Hybrid Algorithm (*a*-TLHA)

In using TLHAs, we just employ a local search algorithm without fine tuning the search parameters. To make the algorithms more effective, we can introduce a mechanism to fine-tune the local search parameters. The pseudo code of an improved algorithm is Program-3. Here, the type Parameter can be a parameter array or a structure containing different types of parameters.

Program-3: a-TLHA

```
Agent P[NumberAgent];  
Parameter par;  
Begin  
    P=Initialize(); // Initialize population.  
    par=InitializeParameter(); // Initialize parameters  
    For t=1 to NumberGeneration  
        For i=1 to NumberAgent  
            P=Decoding(P[i].g); // Find phenotype.  
            pp=LocalSearch(p); // Update phenotype.  
            P[i].f=Evaluation(pp); // Find fitness.  
            Gain=P[i].f-Evaluation(p); // Find fitness gain.
```

```

par=Adaptation(par,p,pp,Gain); // Update parameters.

End

If (Terminating condition) break;

P=Reproduction(P);

End

End

```

In Program-3, a local search parameter set (i.e., par) is defined for the whole population and is shared by all individuals. This parameter set is adapted based on local search result of each individual. If we use TS for local search, the parameter set may include the number of iterations, the sizes of different memories (e.g., Tabu list or short-term memory, medium-term memory, and long-term memory), the Tabu-tenure, rates for intensification and diversification, and so on. If we use SA, local search parameters may include the initial temperature, the speed for temperature cooling down, and the number of iterations for obtaining equilibrium. In general, the parameter set should be properly defined for different local search algorithms.

As mentioned earlier, for many problems we may just use *neighborhood-based* search strategies like visiting several points in a neighborhood and swapping adjacent nodes. In such cases, search parameters may include the number of iterations, the radius of the neighborhood, the step size for search, etc. We may fine tune the parameters as follows:

- If the *fitness gain* obtained through local search is *high* (in a fuzzy logic sense), that is, if the local search is successful, we may increase the step size, or increase the neighborhood radius, to accelerate the search process.
- If the fitness gain is negative, that is, if the local search fails, we may decrease the step size, reduce the neighborhood radius, and increase the number iterations, to search more carefully.

The above rules are *local* in the sense that only the current position of the search point is used to make a judgment. To adapt the parameters more effectively, we should also consider memorizing the search history. For example, even if we just use a few values of the fitness gains, say Gain[t], Gain[t-1], and Gain[t-2], we can estimate roughly the trend of the fitness function in the current neighborhood, and can modify the parameters more properly. To estimate the trend of the fitness function, we can use a neural network or some other regression model. The regression model can make the search space more *transparent*, and the local search parameters can be adapted more efficiently.

In Program-3, the parameters are updated based on information acquired by each individual. Thus, the parameters actually *evolve* in the same way as the pheromone trails

do in ant colony optimization (ACO) (Dorigo and Stützle, 2004). In this sense, a-TLHA is already an evolutionary MA. In TLHA, there is no imitation between individuals, but the memeplex is saved in a library, and is shared by all individuals as a culture of the whole population. Thus, we may define a-TLHA as the first-order MA. That is, at least *variation and retention* for memetic evolution is employed in the algorithm.

In Program-3, we have implicitly assumed that the local search algorithm is fixed, and the purpose of adaptation is to fine tune the parameters. We may also assume that the parameters are fixed (to some default values), and through adaption we can select the most suitable local search algorithm. This is actually the meta-Lamarckian learning algorithm given in (Ong and Keane, 2004). In this case, we can select the most appropriate local search algorithm based on the current search position and the fitness gain. For more detailed information, please refer to (Ong and Keane, 2004).

Program-4: Ga-TLHA

```

DefineType Agent {
    Genotype : g;
    Fitness : f;
    Parameter: par;
};

Agent P[NumberAgent];

Begin
    P=Initialize(); // Initialize population.
    For t=1 to NumberGeneration
        For i=1 to NumberAgent
            P=Decoding(P[i].g); // Find phenotype.
            pp=LocalSearch(p,P[i].par); // Update phenotype.
            P[i].f=Evaluation(pp); // Find fitness.
            Gain=P[i].f-Evaluation(p); // Find fitness gain.
            P[i].par=Adaptation(P[i].par,p,pp,Gain);
                // Update parameters.
}

```

```

If (Terminating condition) break;
P=Reproduction(P);
End
End

```

17.3.5. General Adaptive Two-Layered Hybrid Algorithm (Ga-TLHA)

Now, let us consider some of the extensions of a-TLHA. First, as described above, we may prepare *a set of local search algorithms*, so that individuals can select and use different algorithms for local search. Second, we may use *a unique parameter set* for each individual, so that individuals can have different search strategies even if they may share the same local search algorithm. Third, we may allow each individual to memorize its search history, so that different search strategies can be used in different situations.

In fact, the phenotypes of different individuals represent different points in the search space. At different points the landscape of the search space can be completely different. Even for the same individual, it may move from one region to another during search, and the landscapes of these regions can be different. Thus, it is reasonable to use different local search strategies for different individuals or the same individual at different search points.

Program-4 is the pseudo code of an algorithm summarizing the above considerations. This algorithm is called Ga-TLHA. Here, the type `Parameter` is a *queue* (i.e., a sliding window) containing several historical parameter sets. In Ga-TLHA, each individual has its own parameter set(s) for local search, and the parameter set(s) can be updated based on the individual's search behavior.

Before local search, we need to select one algorithm first from the set Ω of local search algorithms based on the parameter set `par`, and then conduct local search using parameters specified, again, by `par`. A memeplex (i.e., one element of `par`) can be defined as follows:

$$par[t] = (j, \pi_1, \pi_2, \dots, \pi_K), \quad (1)$$

where $1 < j \leq |\Omega|$ is the index of a local search algorithm in Ω and $\pi_k (1 < k \leq K)$ is the k th parameter used by this algorithm, and where K is the maximum number of parameters (or slots) that can be used by a local search algorithm.

In Ga-TLHA, the adaptation mechanism is basically the same as that used in the algorithm a-TLHA, except that we should include a method for selecting local search algorithms. The basic idea for selecting the local search algorithm is as follows:

- If the current local search is successful, the local search algorithm used right now is considered good, and we should increase the probability of choosing this algorithm again next time.
- If the local search does not provide a good result, we should reduce the probability of choosing this algorithm.

That is, each individual can assign a probability to each local search algorithm, and choose a proper algorithm based on the probability, which can be considered as the *belief of an* individual to a certain local search algorithm.

Note that although the algorithm Ga-TLHA is a generalized a-TLHA, it is again a first-order MA. That is, although the memeplex is generalized, there is no big difference for evolving the memeplex. Even if we use different sets of parameters for different individuals, the parameters acquired by one individual are not imitated by other individuals during evolution. That is, good memeplex are neither selected nor transmitted.

17.3.6. Further Generalization

To make some preparations for changing a-TLHA or Ga-TLHA into a full evolutionary MA, let us try to generalize the memeplex again. In general, we may allow each individual to use several local search algorithms *sequentially*, and the same local search algorithm can be used again in this sequence. Thus, a memeplex can have the following form:

$$par = (par_1, par_2, \dots), \quad (2)$$

where each element of par is given by (1). For example, an individual may conduct local search using the following strategy:

$$\text{LocalSearch} = L_1(\theta_1)L_5(\theta_5)L_9(\theta_9)L_5(\theta_5).$$

This local search is conducted in four steps, by using the fifth, the ninth, the fifth (again), and the first local search algorithms in turn. That is, local search in this case is a product of four *local search operators*. The parameter sets may contain parameters such as the number of iterations, the search step size, the number of points to search, etc. Among the parameters, there should be a parameter to specify or limit the computing resources (e.g., time and memory) used by each local search operator. The total resource allocated to local search should not be too long. Otherwise, the computational cost will be too high (Bambha *et al.*, 2004). Also, each time a local operator (say L_5 in the above example) is called, a different set of parameters should be used because the parameters depend on the current location of the search point, too.

Using Ga-TLHA and the above generalizations, we may expect that a good local search strategy can appear during evolution, and the individual using this strategy can have a high fitness. However, it is very difficult to pre-define the rules to adapt the

memeplexes of the individuals during evolution. Therefore, we need an algorithm that can evolve the memeplexes fully automatically.

Program-5: D-EMA

```
DefineType Agent {  
    Genotype: [g1,g2]; // g2 = memotype  
    Fitness : f;  
}  
  
Agent P[NumberAgent];  
  
Begin  
    P=Initialize(); // Initialize population.  
    For t=1 to NumberGeneration  
        For i=1 to NumberAgent  
            P=PhenotypeDecoding(P[i].g1); // Find phenotype.  
            par=MemeplexDecoding(P[i].g2); // Find memeplex.  
            P=LocalSearch(p,par); // Update phenotype.  
            P[i].f=Evaluation(p); // Find fitness.  
    End  
    If (Terminating condition) break;  
    P=Reproduction(P);  
End  
End
```

17.4. Evolutionary MAs

To use the adaptive MAs introduced in the previous section, we must answer several questions. Examples include:

- How often should the local search be applied?
- On which agent should local search be used?
- How long should a local search run?

To answer the above questions through trial and error is certainly not efficient. A better way to answer the above questions and others is evolution. Based on the discussions given so far, in this section we introduce several evolutionary MAs. In some sense, only evolutionary MAs are true MAs (Krasnogor and Gustafson, 2002). Of course, this does not mean that evolutionary MAs are better than adaptive MAs for solving all problems (Wolpert and Macready, 1997). In fact, to obtain good memeplexes through evolution might be computationally expensive. In case we have some *a priori* domain knowledge, we may solve the problem more efficiently by using adaptive MAs. In case we do not have enough domain knowledge, it might be better to use evolutionary MAs.

17.4.1. Direct Evolutionary Memetic Algorithm (D-EMA)

The first evolutionary MA to consider is D-EMA, which is a simple extension of SGA or TLHA. The pseudo code of D-EMA is given as Program-5. In this program, the memotype is encoded as the second part of the genotype.

In D-EMA, the memeplex is first constructed (decoded) from the second part of the genotype, and is then used in local search. By default, it is not necessary to include the phenotype and the memeplex in the definition of the individuals. Further, the memeplex represented in (2) can be encoded in a binary bit string. We may also use a character string defined on any alphabetic set. The memeplex is not imitated from other individuals, but is inherited from the parents only. That is, memeplex transmission is done only *vertically*. It is known that this vertical transmission alone can improve the efficiency of search significantly (Krasnogor and Smith, 2001).

Speaking strictly, D-EMA may not be considered as a true MA. According to the definition given in Section 2, memeplexes are cultures that can be shared by part or all individuals in the population. The memeplexes evolved in D-EMA, however, are just *instincts*. The instinct of a good individual may control the individual to behave smartly, and may be helpful for improving search efficiency; but it is not shared by other individuals. However, the memotypes do evolve during the evolution process, and thus in this chapter we define D-EMA as a first-order evolutionary MA.

17.4.2. Two-Level D-EMA (TLD-EMA)

The algorithm D-EMA can be modified or improved in many directions. First, we may evolve the genotype and the memotype one by one. That is, we may consider a double-loop evolution. The outer loop can be used to evolve the local search or learning strategy, and the inner loop can be used to evolve the solution of the given problem. Or the outer loop can be used to evolve the search point, and the inner loop can be used to evolve the local search strategy (Figure 17.1). In fact, the latter has a clearer physical meaning. That is, at any search point, it is necessary to find a suitable search strategy based on the position of the search point in the search space. Only if the search strategy fits the current search point, we can search more effectively and efficiently.

17.4.3. Canonical Evolutionary Memetic Algorithm (C-EMA)

Another direction to change the D-EMA is to allow the memeplexes to imitate. This results in C-EMA. The pseudo code is given by Program-6. In C-EMA, each individual has its own memeplex, which can be modified before local search. The memeplex par of an individual can be recombined (crossover) with that of some other individual in P, and be mutated before being used for local search. In the pseudo code, IMITATION and VARIATION are conditions for conducting imitation and variation, respectively. The main difference between C-EMA and D-EMA is that the latter only allows vertical transmission of memeplexes, but the former *also* allows *horizontal transmission* of memeplexes, i.e., each individual can imitate the strategies of other (possibly better) individuals through recombination.

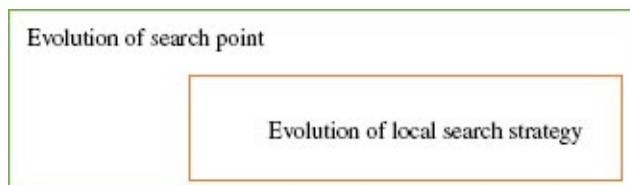


Figure 17.1: A TLD-EMA.

We can consider individuals with *personal* memeplexes as *search agents*. Each individual in the population tries to find the best strategy for local search (i.e., the memeplex), and then tries to find the best solution of the given problem (i.e., the phenotype). When we modify the memeplex of an individual using imitation, the individual to be imitated can be anyone in the population. However, if we observe the human societies, we may have the following considerations:

Program-6: C-EMA

```

DefineType Agent {
    Genotype: [g1,g2]; // g2 is the memotype.
  
```

```

Fitness : f;
};

Agent P[NumberAgent];

Begin

P=Initialize(); // Initialize population.

For t=1 to NumberGeneration

For i=1 to NumberAgent

P=PhenotypeDecoding(P[i].g1); // Find phenotype.

m=P[i].g2;

If(IMITATION) m=Imitation(SelectMemotype(P),m);

// Imitating another individual selected from P.

If(VARIATION) m=Variation(m); // Memeplex mutation.

par=MemeplexDecoding(m);

P=LocalSearch(p,par); // Update phenotype.

P[i].f=Evaluation(p); // Find fitness.

End

If (Terminating condition) break;

P=Reproduction(P);

End

End

```

- The influence of the parents (or direct ancestors) to a child is usually very high, especially in the early stage. Thus, we may assign a higher probability to select the parents to imitate.
- It is better to imitate the elite individual(s) with a higher probability, to improve the fitness of the current individual quickly.
- We may also preserve some search history of each agent, and improve the search strategy properly based on the history.

Thus, a mechanism similar to PSO can be used to modify the memeplex of a given individual, and the updating equations can be given as follows:

$$\begin{aligned} v(t+1) &= av(t) + bw_1(x_p(t) - x_p(t)) + cw_2(x_b(t) - x_b) + dw_3(x_g(t) - x_g), \\ x(t+1) &= x(t) + v(t+1), \end{aligned} \quad (3)$$

where $v(t)$ and $x(t)$ are the search velocity and the position, respectively, of the agent under concern; $x_p(t)$ is the position of one of the parents, x_b is the best position found by the agent itself, x_g is the position of the best agent in the population; a is the inertial, b is the *inheritance influence*, c is the *personal influence*, and d is the *social influence*; and w_1, w_2 , and w_3 are random numbers generated in [0,1].

Note that although Program-6 allows each individual to have its own memeplex, the memeplex so obtained is valid only for one generation. That is, by default CEMA is Baldwinian evolution. The memeplex acquired through recombination and imitation is not passed to the offspring, and is used only for evaluation. We can change the algorithm to a Lamarckian evolution by introducing an encoding mechanism that can encode the acquired memeplex into a memotype. When the landscape of the search space is complex, however, Lamarckian learning can easily fall into a local minimum. Thus, instead of accepting the acquired memeplex directly, we may replace some parts of the memotype with the new materials.

17.4.4. Co-Evolutionary MAs (Co-EMA)

As discussed in Section 2, memeplexes can evolve independently. That is, the population of memeplexes can be separated from that of the phenotypes. In fact, human bodies and human cultures are evolving in a co-existence manner. Some authors referred to this kind of evolution as co-evolution (Smith *et al.*, 2002; Smith, 2003; 2007), although it is neither cooperative co-evolution nor competitive evolution (Zhao, 1997; 1998).

Program-7 is the pseudo code of a Co-EMA. In this program, there are two populations, namely the agent population P and the *spirit* population M . Each individual in M has a memotype and a fitness value. The maximum number of spirits is `NumberSpirit`. Initially, M can be empty or generated at random. During evolution, M is updated based on the fitness values of the existing spirits and the newly generated ones.

In Program-7, a local search strategy for each agent is defined by a new memotype `newMeme[i]`, which in turn is obtained through variation (e.g., recombination and mutation) of two parent memotypes. There are several ways for selecting the parent memotypes (Smith, 2007). If the agents and their spirits are closely *linked*, we may just select the two parents based on the fitness of the agents.

Otherwise, we can select the parent memotypes based on their own fitness values, or just at random.

Program-7: Co-EMA

```
DefineType Agent {
    Genotype: g;
    Fitness : f;
};

DefineType Spirit {
    Memotype: m;
    Fitness : f;
}

Agent P[NumberAgent];
Spirit M[NumberSpirit], newMeme[NumberAgent];

Begin
    (P, M)=Initialize(); // Initialize populations.

    For t=1 to NumberGeneration
        For i=1 to NumberAgent
            P=PhenotypeDecoding(P[i].g); // Find phenotype.

            (m1,m2)=SelectMemotype(M); // Select parent memotypes.

            newMeme[i]=Variation(m1,m2); // Find a new memotype.

            par=MemeplexDecoding(newMeme[i]); // Find memeplex.

            pp=LocalSearch(p,par); // Update phenotype.

            P[i].f=Evaluation(pp); // Find agent fitness.

            Gain=P[i].f-Evaluation(p); // Find fitness gain.

            newMeme[i].f=MemeEvaluation(Gain, p, pp);

            // Find fitness of new memotype;

        End
        If (Terminating condition) break;

        (P,M)=Reproduction(P,M,newMeme);

    End
}
```

Each time when we fine-tune a phenotype, we obtain a fitness gain. We can then define the fitness of the local search strategy based on the fitness gain, the current position p , and the new position pp . In each generation, we produce NumberAgent new spirits. These spirits along with the existing ones can be used together to produce M for the next generation. In this sense, the way for updating M is similar to that used in evolutionary strategy (ES), rather than that of SGA.

As pointed out earlier, the fitness of a local search strategy depends highly on the current position of the search point. Thus, the memeplex fitness defined in Program-7 is local, and a spirit good for one agent may not be good for other agents. To solve this problem, we may define the *effective region(s)* of a spirit using a set ω of phenotypes. That is, the spirit is useful only for phenotypes (and their close neighbors) in ω . For any agent, we can select spirits based on its own phenotype and the effective regions of the spirits.

17.4.5. General Evolutionary Memetic Algorithm (G-EMA)

Both Co-EMA and C-EMA are second-order evolutionary MAs, and they are very similar. The only difference is that in C-EMA each agent has an inherited memeplex. This difference may make C-EMA more stable. This is because the fitness of a local search strategy depends on the current search point, and the parent memotypes selected independent of the phenotype fitness may not fit the current search position. Thus, Co-EMA in its current form may take a large number of generations to find a good combination of agent and spirit.

To solve this problem, we can combine C-EMA and Co-EMA, and obtain the G-EMA, whose pseudo code is given by Program-8. In this program, each agent has its personal memotype (the second part of the genotype). During evolution, this personal memotype is first combined with some other memotype and mutated, and is then used for local search. Hopefully, G-EMA is better than both Co-EMA and C-EMA.

In the literature, the population M of memotypes is often referred to as the meme pool. This meme pool is actually the library or knowledge base for the whole population. In G-EMA, each agent has its inherited instincts. It can imitate its parents or learn knowledge from the library. Hopefully, through imitation and learning, each agent can find a suitable strategy to find a better solution (for the problem under concern).

By default, the knowledge learned by each agent is used only for one generation. That is, the memetic evolution is Baldwinian. We may allow Lamarckian or partial Lamarckian evolution, so that memotypes can be modified in each generation, and re-used in future generations. This way, each agent can conduct *life-time learning*, that is, it can learn in many generations before being selected against. This kind of evolution may encourage agents that can acquire good phenotypes and memeplexes slowly but steadily.

Program-8: G-EMA

```
DefineType Agent {
    Genotype: [g1,g2];
    Fitness : f;
};

DefineType Spirit {
    Memotype: m;
    Fitness : f;
}

Agent P[NumberAgent] ;
Spirit M[NumberSpirit], newMeme[NumberAgent];

Begin
    (P, M)=Initialize(); // Initialize populations.

    For t=1 to NumberGeneration
        For i=1 to NumberAgent
            P=PhenotypeDecoding(P[i].g1); // Find phenotype.

            m=SelectMemotype(M); // Select parent memotype.

            newMeme[i]=Variation(P[i].g2,m); // Find a new memotype.

            par=MemeplexDecoding(newMeme[i]); // Find memeplex.

            pp=LocalSearch(p,par); // Update phenotype.

            P[i].f=Evaluation(pp); // Find agent fitness.

            Gain=P[i].f-Evaluation(p); // Find fitness gain.

            newMeme[i].f=MemeEvaluation(Gain, p, pp);

            // Find fitness of new memotype;

    End

    If (Terminating condition) break;

    (P,M)=Reproduction(P,M,newMeme);

End
```

End

17.4.6. A Simple Example

To show how to use G-EMA, let us consider a simple example. In this example, we suppose that the search space is the parameter space of a neural network controller for navigating a mobile robot. The genotype of each agent is the system parameter set represented in binary string, the phenotype is the system parameter set itself, and the fitness is defined as the performance of the robot (e.g., ability to track a target, efficiency to reach a goal, etc.).

For this example, we can use reinforcement learning (e.g., Q-learning) only for local search (Sutton and Barto, 1998), and define the memeplex as the learning parameters (e.g., the learning rate, the discount factor, and the number of learning cycles). The memotype is the binary representation of the memeplex, and is included in the second part of the genotype.

During evolution, each agent $P[i]$ is evaluated as follows:

- Find the phenotype p from the first part of the genotype;
- Select a memotype m from the meme pool M (which is initialized at random);
- Find a new memotype $newMeme[i]$ by crossover and mutation, based on m and the second part of the genotype;
- Find a parameter set par based on $newMeme[i]$;
- Conduct reinforcement learning on the phenotype p , and obtain a new phenotype pp ;
- Evaluate p and pp , and find the fitness gain $Gain$; and
- Evaluate $newMeme[i]$ based on p , pp , and $Gain$.

Figure 17.2 illustrates the above evaluation process. Based on the evaluation results, the agent population P can be updated in the same way as in SGA, and the meme pool M can be updated using methods proposed for ES.

17.4.7. Hierarchical Evolutionary MA (H-EMA)

So far, we have considered each individual in a population as a search agent. That is, during evolution each agent searches for the best local search strategy while searching for the best position in the problem space. We may consider the whole population as a search agent, and define another higher level population. Thus, in principle we can have a H-EMA.

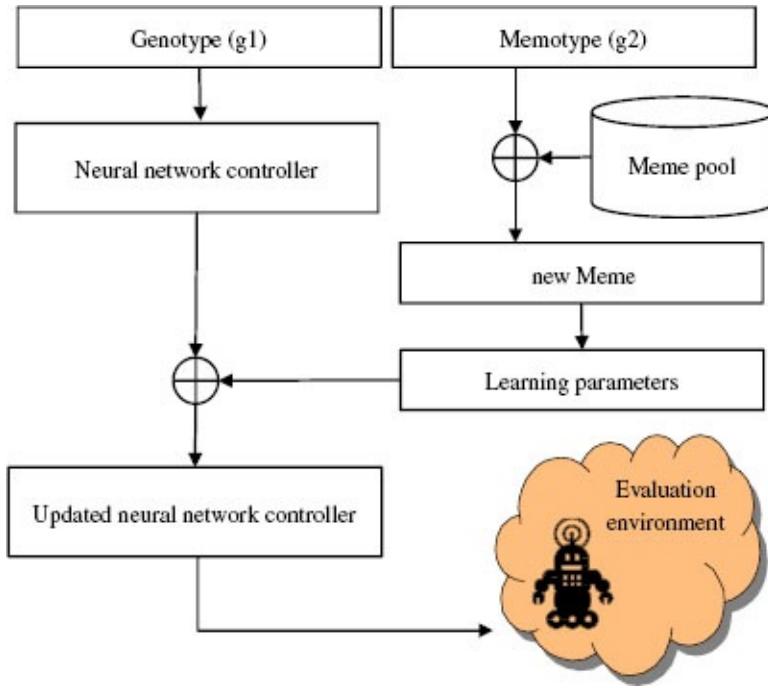


Figure 17.2: Illustration of evaluation process of an agent.

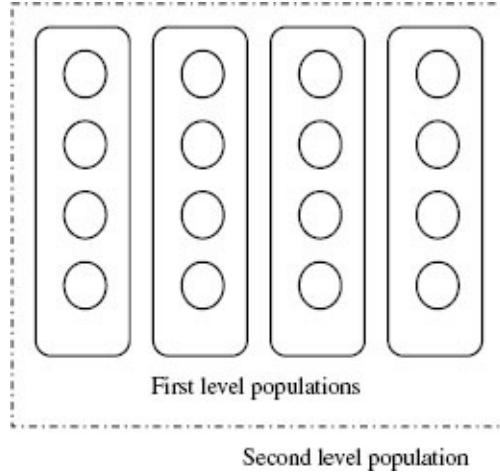


Figure 17.3: Illustration of (H-EMA).

Figure 17.3 illustrates the basic idea of H-EMA. In Figure 17.3, circles represent the *first level agents*, and many of them form a *first level population*, or a *second level agent*. Many second level agents form a *second level population*, which can be defined as a *third level agent*. This way, we can define many levels or hierarchies of evolution. In general, for the j th level, we can use SGA or any MAs to evolve the j th level agents using the j th level population. During evolution, genetic or memetic materials can be transmitted from one agent to another, and this is equivalent to immigration between the $(j - 1)$ th level populations.

17.5. Conclusions and Remarks

In this chapter, we have defined the memes, memotypes, and memeplexes more clearly, and introduced several MAs. Although these algorithms have been summarized in several survey papers, we believe that this chapter is more readable and comprehensible for the beginners of this area. Readers can easily master the whole picture of MAs, and ready to write programs for solving their own problems based on the pseudo codes given here.

In this chapter, we have tried to be *neutral* about the performance of the MAs. We have neither mentioned whether MAs are better than existing meta-heuristic algorithms (e.g., TS, SA-PSO, and ACO), nor pointed out which MA is the best. Based on no free lunch theory (Wolpert and Macready, 1997), any single algorithm cannot be the best for solving all problems. For instance, evolutionary MAs may not be more efficient than adaptive ones. If we have *a priori* domain knowledge, we may try adaptive MAs first. Otherwise, we may try evolutionary MAs, gain some knowledge, and then try to fix some parameters to narrow the search space for memetic evolution.

Note also that this chapter focused only on the algorithm aspect. We have tried to use the pseudo codes to explain the algorithms, so that readers can understand and use them straightforwardly. Encoding and decoding of individuals (i.e., phenotypes and memeplexes), evaluation of fitness, terminating condition, evolutionary operators, and so on, depend on the problem under concern, and are common to all population based algorithms. Readers who are interested in applications may find more information in the references (Choi and Moon, 2005; Garcia *et al.*, 2008; Cowling *et al.*, 2000; Krasnogor *et al.*, 2002).

Further, in this chapter we have put local search just before evaluation of a phenotype. In practice, local search can be put in different positions. For example, local search can be conducted in initialization of the individuals, after evaluation, after crossover and/or mutation, and so on. In fact, *the timing of local search* itself can be considered as one of the memes, and can be determined through evolution. Agents who conduct local search in good timing may find good solutions (for the given problem) more efficiently and effectively.

In this chapter, we did not consider MAs for multi-objective search problems. In principle, if we can formulate multiple objectives into one function (e.g., using different penalties), all results given here can be applied straightforwardly. Otherwise, we need fitness sharing or some other mechanism to keep all solutions close to the Pareto front. Readers who are interested in this topic may refer to (Ishibuchi *et al.*, 2003) and (Knowles and Corne, 2001), and references therein.

References

- Bambha, N. K., Bhattacharyya, S. S., Teich, J. and Zitzler, E. (2004). Systematic integration of parameterized local search into evolutionary algorithms. *IEEE Trans. Evolutionary Computation*, 8, pp. 137–154.
- Beyer, H. G. and Schwefel, H. P. (2002). Evolution strategies: A comprehensive introduction. *Natural Computing*, 1(1), pp. 3–52.
- Blackmore, S. (1999). *The Meme Machine*. New York: Oxford University Press.
- Chen, X., Ong, Y. S., Lim, M. H. and Tan, K. C. (2011). A multi-facet survey on memetic computation. *IEEE Trans. Evolutionary Computation*, 15(5), pp. 591–607.
- Choi, S. S. and Moon, B. R. (2005). A graph-based Lamarckian-Baldwinian hybrid for the sorting network problem. *IEEE Trans. Evolutionary Computation*, 9(1), pp. 105–114.
- Cowling, P., Kendall, G. and Soubeiga, E. (2000). A hyperheuristic approach to scheduling a sales summit. *PATAT 2000*, Springer Lecture Notes in Computer Science, Konstanz, Germany, pp. 176–190.
- Dawkins, R. (1976). *The Selfish Gene*. Oxford, UK: Oxford University Press.
- Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*. Cambridge, MA: MIT Press.
- Eberhart, R. C. and Shi, Y. (2001). Particle swarm optimization: Developments, applications and resources. *Proc. IEEE Congress of Evolutionary Computation*, 1, pp. 27–30.
- Fogel, L. J., Owens, A. J. and Walsh, M. J. (1966). *Artificial Intelligence through a Simulation of Evolution*. New York: John Wiley & Sons.
- Garcia, S., Cano, J. R. and Herrera, F. (2008). A memetic algorithm for evolutionary prototype selection: A scaling up approach. *Pattern Recognition*, 41(8), pp. 2693–2709.
- Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5), pp. 533–549.
- Glover, F. (1989). Tabu Search—(Part 1). *ORSA Journal on Computing*, 1(2), pp. 190–206.
- Glover, F. (1990). Tabu Search—(Part 2). *ORSA Journal on Computing*, 2(1), pp. 4–32.
- Goffe, W. L., Ferrier, G. D. and Rogers, J. (1994). Global optimization of statistical functions with simulated annealing. *Journal of Econometrics*, 60, pp. 65–99.
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley.
- Hodgson, G. M. (2005). Generalizing Darwinism to social evolution: Some early attempts. *Journal of Economic Issues*, pp. 899–914.
- Ishibuchi, H., Yoshida, T. and Murata, T. (2003). Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Trans. Evol. Comput.*, 7, pp. 204–223.
- Knowles, J. and Corne, D. (2001). A comparative assessment of memetic, evolutionary and constructive algorithms for the multi-objective d-MST problem. *Proc. Genetic and Evolutionary Computation Workshop*.
- Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.
- Koza, J. R. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press.
- Krasnogor, N., Blackbourne, B., Burke, E. and Hirst, J. (2002). Multimeme algorithms for protein structure prediction. *Proc. Parallel Problem Solving From Nature*, Lecture Notes in Computer Science, New York: Springer, pp. 769–778.
- Krasnogor, N. and Gustafson, S. (2002). Toward truly ‘memetic’ memetic algorithms: Discussion and proof of concepts. In *Proc. Advances in Nature-Inspired Computation: The PPSN VII Workshops*, pp. 21–22.
- Krasnogor, N., and Smith, J., (2001). Emergence of profitable search strategies based on a simple inheritance mechanism. In Spector, L., Goodman, E., Wu, A., Langdon, W., Voigt, H. M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. and Burke, E. (eds.). *Proc. Genetic and Evolutionary Computation Conference*, San Francisco: Morgan Kaufmann, pp. 432–439.

- Krasnogor, N. and Smith, J. (2005). A tutorial for competent memetic algorithms: Model, taxonomy, and design issues. *IEEE Trans. Evol. Comput.*, 9, pp. 474–488.
- Miller, J. A., Potter, W. D., Gandham, R. V. and Lapena, C. N. (1993). An evaluation of local improvement operators for genetic algorithms. *IEEE Trans. Syst. Man. Cybern.*, 23, pp. 1340–1351.
- Minsky, M. L. (1986). *The Society of Mind*. New York: Simon and Schuster.
- Moscato, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech Concurrent Computation Program*, Publication Rep. 790.
- Ong, Y. S. and Keane, A. J. (2004). Meta-Lamarckian in memetic algorithm. *IEEE Trans. Evol. Comput.*, 8, pp. 99–110.
- Ong, Y. S., Lim, M. H. and Chen, X. (2010). Memetic computation: Past, present and future. *IEEE Comput. Intell. Mag.*, 5(2), pp. 24–31.
- Ong, Y. S., Lim, M. H., Zhu, N. and Wong, K. W. (2006). Classification of adaptive memetic algorithms: A comparative study. *IEEE Trans. Syst. Man. Cybern. B*, 36(1), pp. 141–152.
- Polanyi, M. (1966). *The Tacit Dimension*. Chicago: University of Chicago Press.
- Schaffer, J. D., Whitley, D. and Eshelman, L. I. (1992). Combinations of genetic algorithms and neural networks: A survey of the state-of-the-art. *COGANN-92 Combinations of Genetic Algorithms and Neural Networks*, IEEE Computer Society Press, Los Alamitos, CA, pp. 1–37.
- Smith, J. E. (2003). Co-evolving memetic algorithms: A learning approach to robust scalable optimization. In *IEEE Congress on Evolutionary Computation*. Piscataway, NJ: IEEE Press, 1, pp. 498–505.
- Smith, J. E. (2007). Coevolving memetic algorithms: A review and progress report. *IEEE Trans. Systems, Man, and Cybernetics — Part B*, 37(1), pp. 6–17.
- Smith, J. E. et al., (2002). Co-evolution of memetic algorithms: Initial investigations. In Guervos, G. et al. (eds.) *Parallel Problem Solving From Nature — PPSN VII*. Berlin, Germany: Springer, pp. 537–548.
- Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. Cambridge: The MIT Press.
- Thangaraj, R., Pant, M., Abraham, A. and Bouvry, P. (2011). Particle swarm optimization: Hybridization perspectives and experimental illustrations. *Applied Mathematics and Computation*.
- Wolpert, D. and Macready, W., (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1, 67–82.
- Zhao, Q. F. (1997). A society model for cooperative co-evolutionary algorithms: Some case studies. *Proc. Australia-Japan Joint Workshop on Intelligent and Evolutionary Systems*. Canberra, Australia, pp. 30–36.
- Zhao, Q. F. (1998). A general framework for cooperative co-evolutionary algorithms: A society model. *Proc. IEEE Int. Conf. Evolutionary Computation*, Anchorage, Alaska, pp. 57–62.

Part IV

Hybrid Systems

Chapter 18

Multi-Objective Evolutionary Design of Fuzzy Rule-Based Systems

Michela Antonelli, Pietro Ducange and Francesco Marcelloni

In the last years, the generation of fuzzy rule-based systems (FRBSs) from data has been tackled by using a multi-objective optimization approach, with accuracy and interpretability as the objectives to be optimized. Multi-objective evolutionary algorithms have been so often used in this context that the FRBSs generated by exploiting these algorithms have been denoted as multi-objective evolutionary fuzzy systems (MOEFSs). Recent MOEFSs allow concurrently learning the rule-bases (RBs) of the FRBSs and some elements of the database (DB), such as the granularity and the membership function parameters of the fuzzy partitions, during the evolutionary approach. At the end of the execution, a number of FRBSs with different trade-offs between accuracy and interpretability are produced: the designer can choose the FRBS with the most suitable trade-off for the specific application domain. The aim of this chapter is to present the main features of MOEFSs and to discuss how they are applied to both regression and classification problems. Some example of application is also shown and discussed.

18.1. Introduction

Fuzzy rule-based systems (FRBSs) are widely used in different engineering fields such as control, pattern recognition, system identification and signal analysis. They allow achieving accuracies comparable to other approaches and have the capability of explaining how their outputs are generated from the input values. An FRBS consists of a linguistic rule-base (RB), a database (DB) containing the fuzzy sets associated with the linguistic terms used in the RB and a fuzzy logic inference engine. RB and DB compose the knowledge base (KB) of the FRBS. Formally, an FRBS is a mathematical model that, given an input vector, computes an output value, exploiting the knowledge coded in the RB and in the DB, and an inference process based on fuzzy logic.

The most natural approach to build an FRBS is to elicit the knowledge from a human expert and to codify this knowledge in the KB, but sometimes the complexity of the application domain can make this approach hardly viable. Thus, several methods have been proposed in the literature to generate the KB from data (typically expressed as input–output patterns). When FRBSs are generated with the only objective of maximizing the accuracy, they are generally characterized by a high number of rules and by linguistic fuzzy partitions with a low level of comprehensibility, thus loosing that feature which may make FRBSs preferable to other approaches, namely their interpretability (Alonso *et al.*, 2009; Cordón, 2011; Gacto *et al.*, 2011; Zhou and Gan, 2008). Thus, in the last years, the generation of FRBSs from data has been modeled as a multi-objective optimization problem, taking accuracy and interpretability as the objectives to be optimized. Multi-objective evolutionary algorithms have been so widely used as optimization technique in this framework that the term multi-objective evolutionary fuzzy systems (MOEFSs) has been coined (Cordón, 2011; Ducange and Marcelloni, 2011; Fazzolari *et al.*, 2013a).

In the first MOEFSs proposed in the literature, multi-objective evolutionary algorithms have been used to select (Ishibuchi *et al.*, 1997; Ishibuchi and Yamamoto, 2004) or learn (Cococcioni *et al.*, 2007; Ducange *et al.*, 2010; Ishibuchi *et al.*, 2001) rules, and to perform the tuning (Botta *et al.*, 2009) of the DB with prefixed DB and RB, respectively. On the other hand, the most recent MOEFSs perform the learning (Alcalá *et al.*, 2009; Antonelli *et al.*, 2009, 2011a, 2011b; Pulkkinen and Koivisto, 2010) or the selection (Alcalá *et al.*, 2007; Gacto *et al.*, 2009, 2010; Nguyen *et al.*, 2013) of the rules concurrently with the learning of some elements of the DB, namely the granularity and the membership function parameters of the fuzzy partitions. While in rule selection (RS), rules are selected from a initial set of candidate rules generated from data by some heuristic, in rule learning (RL), rules are created during the evolutionary process.

The aim of this chapter is to present the main features of MOEFSs and to discuss how they are applied to both regression and classification problems. For the different solutions we also show some experimental results. The chapter is organized as follows. [Section 18.2](#) introduces the FRBSs and discusses their interpretability. [Section 18.3](#) describes the most

popular multi-objective evolutionary algorithms used in the framework of MOEFSs. [Section 18.4](#) introduces the main features of MOEFSs, such as chromosome coding, mating operators, and objective functions. [Section 18.5](#) shows how the results of MOEFSs can be analyzed and compared. In [Section 18.6](#), we discuss some experimental results obtained by applying MOEFSs to both regression and classification problems. Finally, [Section 18.7](#) draws some conclusions.

18.2. Fuzzy Rule-Based Systems

A fuzzy set A defined on a universe of discourse U is characterized by a membership function $A(x) : U \rightarrow [0, 1]$ which associates with each element \hat{x} of U a number $A(\hat{x})$ in the interval $[0, 1]$: $A(\hat{x})$ represents the grade of membership of \hat{x} in A (Zadeh, 1965). The support and the core of A are the crisp subsets of A with, respectively, non-zero membership grades and membership grades equal to 1.

Different types of membership functions such as Gaussian, triangular, and trapezoidal, have been proposed and used for designing FRBSs. For the sake of simplicity, in this chapter, we consider triangular fuzzy sets, which are identified by the tuples (a, b, c) , where a and c correspond to the left and right extremes of the support of the fuzzy set, and b to the core. Formally, a triangular membership function can be defined as follows:

$$A(x) = \begin{cases} \frac{a-x}{b-a} & a \leq x \leq b \\ \frac{c-x}{c-b} & b < x \leq c \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Based on the notion of fuzzy set, Zadeh introduced the concept of linguistic variable (Zadeh, 1965). A linguistic variable X is a variable whose values are linguistic terms. X is characterized by the term set $T(X)$ of X , that is, the set of linguistic values of X with each value being a fuzzy set defined on universe U . The set $P = \{A_1, \dots, A_{|T(X)|}\}$, where $|T(X)|$ is the cardinality of $T(X)$, of the fuzzy sets used in defining the terms in $T(X)$ composes a fuzzy partition of the universe U .

Let $X = \{X_1, \dots, X_F\}$ be the set of input variables and X_{F+1} be the output variable. Let U_f , with $f = 1, \dots, F + 1$, be the universe of the f th variable X_f . Let $P_f = \{A_{f,1}, \dots, A_{f,T_f}\}$ be a fuzzy partition of T_f fuzzy sets on variable X_f . Let $\{(\mathbf{x}_1, x_{F+1,1}), \dots, (\mathbf{x}_N, x_{F+1,N})\}$ be a training set composed of N input–output pairs, with $\mathbf{x}_t = [x_{t,1} \dots, x_{t,F}] \in \Re^F$, $t = 1, \dots, N$.

In regression problems, X_{F+1} is a continuous variable and, therefore, $\forall t \in [1 \dots N]$, $x_{F+1,t} \in \Re$. With the aim of estimating the output value corresponding to a given input vector, we adopt an RB composed of M linguistic fuzzy rules expressed as:

$$\begin{aligned} R_m : \text{IF } X_1 \text{ is } A_{1,j_{m,1}} \text{ AND } \dots \text{ AND } X_f \text{ is } A_{f,j_{m,f}} \text{ AND } \dots \\ \dots \text{ AND } X_F \text{ is } A_{F,j_{m,F}} \text{ THEN } X_{F+1} \text{ is } A_{F+1,j_{m,F+1}}, \end{aligned} \quad (2)$$

where $j_{m,f} \in [1, T_f]$, $f = 1, \dots, F + 1$, identifies the index of the fuzzy set (among the T_f linguistic terms of partition P_f), which has been selected for X_f in rule R_m .

In classification problems, X_{F+1} is categorical and $x_{F+1,t} \in C$, where $C = \{C_1, \dots, C_K\}$ is the set of K possible classes. With the aim of determining the class of a given input vector, we adopt an RB composed of M rules expressed as:

$$R_m : \text{IF } X_1 \text{ is } A_{1,jm,1} \text{ AND } \dots \text{ AND } X_f \text{ is } A_{f,jm,f} \text{ AND } \dots \\ \dots \text{ AND } X_F \text{ is } A_{F,jm,F} \text{ THEN } X_{F+1} \text{ is } C_{jm} \text{ with } RW_m \quad (3)$$

where C_{jm} is the class label associated with the m th rule and RW_m is the rule weight, i.e., a certainty degree of the classification in the class C_{jm} for a pattern belonging to the fuzzy subspace delimited by the antecedent of the rule R_m . The FRBS designed for classification problems is often denoted as fuzzy rule-based classifier (FRBC).

Usually, a new fuzzy set $A_{f,0}$ ($f = 1, \dots, F$) is defined for all the F input variables. This fuzzy set, which represents the “don’t care” condition, is characterized by a membership function equal to 1 on the overall universe. The term $A_{f,0}$ allows generating rules that contain only a subset of the input variables (Ishibuchi *et al.*, 2004).

Given an input pattern $\hat{\mathbf{x}} \in \Re^F$, the strength of activation (*matching degree* of the rule with the input) of the rule R_m is computed as:

$$w_m(\hat{\mathbf{x}}) = \prod_{f=1}^F A_{f,jm,f}(\hat{x}_f), \quad (4)$$

where $A_{f,jm,f}(X_f)$ is the membership function associated with the fuzzy set $A_{f,jm,f}$. For the sake of simplicity, in the formula, we have only considered the product as t-norm for implementing the logical conjunction.

In regression problems, the estimated output \hat{x}_{F+1} is obtained by computing the output of any rule R_m , by aggregating these outputs and finally applying a defuzzification strategy. The output of a rule R_m is a fuzzy set $\hat{A}(X_{F+1})$ calculated by applying the implication operator I between $w_m(\hat{\mathbf{x}})$ and the output fuzzy set $A_{F+1,jm,F+1}$:

$$\hat{A}_m(X_{F+1}) = I(w_m(\hat{\mathbf{x}}), A_{F+1,jm,F+1}(X_{F+1})). \quad (5)$$

In the context of MOEFSs, the implication operator is typically implemented as minimum or product. The fuzzy sets inferred from each rule are therefore aggregated to produce the fuzzy set $\hat{A}(X_{F+1})$ as follows:

$$\hat{A}(X_{F+1}) = \max_{m=1, \dots, M} I(w_m(\hat{\mathbf{x}}), A_{F+1,jm,F+1}(X_{F+1})), \quad (6)$$

where the aggregation operator is implemented as maximum. Finally, $\hat{A}(X_{F+1})$ is defuzzified by applying some defuzzification strategy to produce the output \hat{x}_{F+1} . For example, by using the center of gravity strategy (Klir and Yuan, 1995) we obtain:

$$\hat{x}_{F+1} = \frac{\int \hat{A}(X_{F+1}) \cdot X_{F+1} \cdot dX_{F+1}}{\int \hat{A}(X_{F+1}) \cdot dX_{F+1}}. \quad (7)$$

In classification problems, the estimated output class \hat{C} is obtained by computing the

association degree $h_m(\hat{x})$ with class C_{jm} for each rule R_m , and then by applying a reasoning method.

The *association degree* $h_m(\hat{x})$ is calculated as:

$$h_m(\hat{x}) = w_m(\hat{x}) \cdot RW_m. \quad (8)$$

Two different definitions of the rule weight RW_m are commonly found in the literature (Cordon *et al.*, 1999; Ishibuchi *et al.*, 2004):

(1) *The certainty factor*:

$$CF_m = \frac{\sum_{x_t \in C_{jm}} w_m(x_t)}{\sum_{t=1}^N w_m(x_t)}. \quad (9)$$

(2) *The penalized certainty factor*:

$$PCF_m = CF_m - \frac{\sum_{x_t \notin C_{jm}} w_m(x_t)}{\sum_{t=1}^N w_m(x_t)}. \quad (10)$$

The *reasoning method* uses the information from the RB to determine the class label for a specific input pattern. Two different approaches are often adopted in the literature:

- (1) *The maximum matching*: An input pattern is classified into the class corresponding to the rule with the maximum association degree calculated for the pattern.
- (2) *The additive combination*: An input pattern is classified into the class corresponding to the maximum total strength of the vote. For each class C_k , the total strength V_{Ck} is computed as follows:

$$V_{Ck}(\hat{x}) = \sum_{R_m \in RB; C_{jm} = C_k} h_m(\hat{x}). \quad (11)$$

Thus, for each pattern (\hat{x}), each fuzzy rule gives a vote for its consequent class.

For the sake of brevity, in this chapter we consider only the two types of FRBSs described in this section. Actually, other typologies of FRBSs, such as TSKFRBSs (Cococcioni *et al.*, 2011), FRBSs with DNF rules (Casillas *et al.*, 2009) and FRBSs based on multiple granularities (Ishibuchi and Nojima, 2007), have been also considered in the MOEFS specialized literature.

One of the most appealing aspects of FRBSs is their interpretability. Interpretability has been widely discussed in the last years, also in the framework of MOEFSs. Since interpretability is a subjective concept, it is hard to find a worldwide agreed definition and consequently a universal measure of interpretability. Thus, researchers have focused their attention on discussing some factors, which characterize interpretability, and on proposing some constraints which have to be satisfied for these factors. Considerations on the main

factors that influence interpretability can be found in Guillaume (2001) and de Oliveira (1999). A homogeneous description of semantic and syntactic interpretability issues regarding both the RB and the DB has been recently published in a survey on interpretability constraints (Mencar and Fanelli, 2008).

In Zhou and Gan (2008), a taxonomy of fuzzy model interpretability has been proposed in terms of both low-level and high-level interpretability. Low-level interpretability is related to the semantic constraints that ensure fuzzy partition interpretability, while high-level interpretability is associated with a set of criteria defined on the RB. Furthermore, a conceptual framework for characterizing interpretability of fuzzy systems has been introduced in (Alonso *et al.*, 2009): this framework includes a global description of the FRBS structure, on the basis of the taxonomy and constraints discussed in (Mencar and Fanelli, 2008) and (Zhou and Gan, 2008), respectively, and a local explanation for understanding the system behavior. This local explanation considers a number of factors such as inference mechanisms, aggregation, conjunction and disjunction operators, defuzzification, and rule type, which affect the system behavior.

Recently, the most relevant measures and strategies exploited to design interpretable FRBSs have been reviewed in Gacto *et al.* (2011). Here, a taxonomy of the interpretability measures has been proposed by considering two different dimensions, namely semantic and complexity, at RB and DB levels. As shown in [Figure 18.1](#), extracted from Gacto *et al.* (2011), this taxonomy is therefore organized into four quadrants. In the figure, the main measures used for each combination dimension/level are presented in the corresponding quadrant. In the context of MOEFSs, measures in the complexity-RB and semantic-DB quadrants have been mainly employed as objectives of the evolutionary optimization process.

18.3. Multi-Objective Evolutionary Optimization

Multi-objective evolutionary algorithms (MOEAs) have proved to be very effective to search for optimal solutions to problems that incorporate multiple performance criteria (Coello *et al.*, 2007; Zitzler *et al.*, 2004). Typically, these criteria are in competition with each other. Thus, an MOEA has to determine a trade-off between the objectives, since improvement in one objective cannot be generally achieved without detriment to another. Thus, a family of equally valid solutions will exist, where each solution will tend to satisfy a criterion to a higher extent than another.

	<i>Rule Base Level</i>	<i>Data Base Level</i>
<i>Complexity</i>	Number of Rules Number of Conditions Average Rule Length	Number of Features Number of Membership Functions
<i>Semantic</i>	Consistency of Rules Rules Fired at the Same Time Transparency of the Structure Cointension	Coverage Normalization Distinguishability Order Relative and Combined Measures

Figure 18.1: Taxonomy of interpretability measures proposed in the literature.

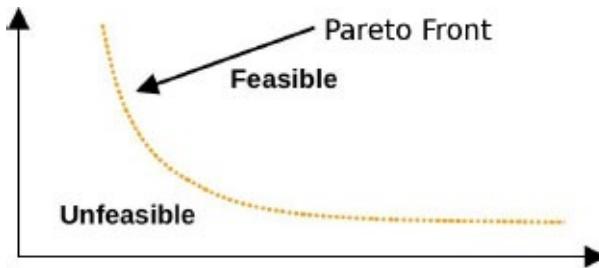


Figure 18.2: An example of Pareto front.

Different solutions are compared with each other by using the notion of Pareto dominance. Let I be the number of objectives. A solution x associated with a performance vector \mathbf{u} dominates a solution y associated with a performance vector \mathbf{v} if and only if, $\forall i \in \{1, \dots, I\}$, u_i performs better than, or equal to, v_i and $\exists i \in \{1, \dots, I\}$ such that u_i performs better than v_i , where u_i and v_i are the i th element of vectors \mathbf{u} and \mathbf{v} , respectively.

A solution is said to be Pareto optimal if it is not dominated by any other possible solution. The set of points that correspond to the Pareto-optimal solutions is denoted as *Pareto front*. Thus, the aim of a multi-objective search algorithm is to discover a family of solutions that are a good approximation of the Pareto front. In [Figure 18.2](#), we show an example of Pareto front for a two-objective minimization problem.

A number of MOEAs have been proposed in the literature (Coello *et al.*, 2007; Zitzler *et al.*, 2004). Some of the most popular among these algorithms are the strength pareto evolutionary algorithm (SPEA) and its evolution (SPEA2) (Zitzler *et al.*, 2001), the Niced Pareto Genetic Algorithm (NPGA) (Horn *et al.*, 1994), the different versions of the Pareto Archived Evolution Strategy (PAES) (Knowles and Corne, 2000), and the Non-

dominated Sorting Genetic Algorithm (NSGA) and its evolution (NSGA-II) (Deb *et al.*, 2002). Some of these algorithms have been used in the context of MOEFSs. SPEA2 and NSGA-II have been employed, for instance, in (Alcalá *et al.*, 2007; Gacto *et al.*, 2009) and (Ducange *et al.*, 2010; Ishibuchi and Nojima, 2007), respectively. PAES has been employed in most of our works on MOEFSs and has also been used for the MOEFSs discussed in the experimental results shown in this chapter. For the sake of brevity, in the following, we will shortly discuss only this MOEA.

18.3.1. PAES and (2+2)M-PAES

PAES (Knowles and Corne, 2000) probably represents the simplest possible nontrivial algorithm capable of generating diverse solutions in the Pareto optimal set. PAES consists of three parts: the candidate solution generator, the candidate solution acceptance, and the non-dominated-solutions archive. The candidate solution generator maintains a single current solution c , and, at each iteration, produces a new solution m from c , by using a mutation operator. The candidate solution acceptance compares m with c . Three different cases can arise:

- (1) c dominates m : m is discarded;
- (2) m dominates c : m is inserted into the archive and possible solutions in the archive dominated by m are removed; m replaces c in the role of current solution;
- (3) Cases 1 and 2 are not satisfied: m is added to the archive only if it is dominated by no solution contained in the archive; m replaces c in the role of current solution only if m belongs to a region with a crowding degree smaller than, or equal to, the region of c .

The *crowding degree* is computed by firstly dividing the space where the solutions of the archive lie into a number of equally sized regions, generating in this way a grid, and then by counting the solutions that belong to the regions. The number of these solutions determines the crowding degree of a region. The recursive subdivision of the space and assignment of grid locations is carried out using an adaptive method that eliminates the need for a niche size parameter. This adaptive method works by calculating the range in the objective space of the current solutions in the archive and adjusting the grid so that it covers this range. Grid locations are then recalculated. This is done only when the range of the objective space of archived solutions changes by a threshold amount to avoid recalculating the ranges too frequently. The only parameter that must then be set is the number of divisions of the space (and hence grid locations) required.

This approach tends to prefer solutions which belong to poorly crowded regions, so as to guarantee a uniform distribution of solutions along the Pareto front. If the archive is full, the solution m is added to the archive if it lies in a less crowded region of the archive than some members of the archive. In this case, a member of the archive from the most crowded region is removed from the archive. PAES terminates after a given number Z of

iterations or when no solution is inserted into the archive for a number G of iterations, with $G < Z$. The candidate solution acceptance strategy generates an archive which contains only non-dominated solutions. On PAES termination, the archive includes the set of solutions which are an approximation of the Pareto front. At the beginning, the archive is empty and the first current solution is randomly generated. The aforementioned PAES is called (1+1)PAES, since it uses a single current solution and generates a single mutant. A more general version of PAES, denoted $(\mu + \gamma)$ PAES, uses μ current solutions and generates γ mutants: each mutant is created by mutating one of the μ current solutions, which is selected via binary tournament selection using the fitness values assigned in the previous iteration. The fitness values are computed by considering the dominance of the current solutions with respect to the archive and their crowding degree.

In the context of MOEFSs, in the work discussed in Cococcioni *et al.* (2007), we have proposed a modified version of $(\mu + \gamma)$ PAES, with $\mu = 2$ and $\gamma = 2$. We denoted this version as (2 + 2)M-PAES, where M-PAES stands for modified PAES. Unlike the original (2 + 2)PAES, which uses only mutation to generate new candidate solutions, our approach exploits the crossover and a set of appropriately mutation operators. We experimentally verified that crossover helps create an approximation of the Pareto front where solutions are uniformly distributed along the front. We start with two randomly generated current solutions c_1 and c_2 . At each iteration, the application of crossover and mutation operators produces two new solutions s_1 and s_2 from the current solutions c_1 and c_2 . Unlike classical PAES, which maintains c_1 and c_2 as current solutions until they are not replaced by solutions with particular characteristics, we randomly extract, at each iteration, the current solutions. If the archive contains a unique solution, c_1 and c_2 correspond to this unique solution. We experimentally verified that the random extraction of the current solutions from the archive allows us to extend the set of non-dominated solutions contained in the archive so as to obtain a better approximation of the Pareto front. As regards the candidate solution acceptance, s_1 and s_2 are added to the archive only if they are dominated by no solution contained in the archive; possible solutions in the archive dominated by s_1 or s_2 are removed.

18.4. MOEFSs

When designing FRBSs, accuracy and interpretability are objectives in competition with each other: an increase in accuracy corresponds typically to a decrease in interpretability. The best trade-off between accuracy and interpretability generally depends on the application context and cannot be fixed *a priori*. Thus, it would be desirable to produce a set of possible optimal FRBSs with different values of accuracy and interpretability, and let the user decide for the best solution on the basis of the specific application context. MOEAs appear to be a very suitable tool for achieving this objective: we can determine an approximation of the Pareto front, where each point in the front represents an FRBS characterized by a different tradeoff between accuracy and interpretability.

Once fixed the type of FRBS, i.e., the model for regression or classification, and the specific defuzzification or reasoning method, the design process of an FRBS consists in determining the rules in the RB and the parameters of the DB that define the membership functions associated with each linguistic term used in the rules. In MOEFSs, the design process is performed as a multi-objective evolutionary optimization process. In the following, we will discuss the different chromosome coding, mating operators and fitness functions used in the literature. Then, we will present a pseudo-code which describes an example of implementation of an MOEFS.

18.4.1. Chromosome Coding

The RB can be derived by selecting rules from a pre-defined set of candidate rules or by evolving the antecedent and consequent parts of each rule from scratch. We denote the two strategies as RS and RL, respectively. Moreover, in order to handle high dimensional datasets, we have recently proposed a new approach based on rule and condition selection (RCS) (Antonelli *et al.*, 2013b). In RCS, starting from a pre-defined set of candidate rules, a reduced number of rules together with a reduced number of conditions for each rule are selected during the evolutionary process.

In MOEFSs based on RS (see for example the work in Alcalá *et al.* (2007)), a binary chromosome is used to codify the overall RB. The i th binary gene identifies if the i th rule of a predefined set of candidate rules has been selected or not. The initial set of rules is generated by exploiting some heuristic: in a number of contributions, the Wang and Mendel algorithm (Wang and Mendel, 1992) for regression problems, and its modified version for classification problems, have been used for generating the set of candidate rules.

Also in MOEFSs based on RL a chromosome codifies the overall RB: in this case each gene in the chromosome identifies the index $j_{m,f}$ of the fuzzy set selected for the corresponding linguistic variable X_f in each rule of the RB. When dealing with evolutionary rule learning, in order to generate compact rule-bases, the maximum number

of rules M_{max} , which can be contained in the RB, is fixed. In [Figure 18.3](#) we show an example of RL chromosome for regression problems.

$j_{1,1}$	\dots	$j_{1,F}$	$j_{1,F+1}$	\dots	$j_{M,1}$	\dots	$j_{M,F}$	$j_{M,F+1}$
-----------	---------	-----------	-------------	---------	-----------	---------	-----------	-------------

Figure 18.3: RL chromosome for regression problems.

$j_{1,1}$	\dots	$j_{1,F}$	C_{j_1}	\dots	$j_{M,1}$	\dots	$j_{M,F}$	C_{j_M}
-----------	---------	-----------	-----------	---------	-----------	---------	-----------	-----------

Figure 18.4: RL chromosome for classification problems.

In the case of FRBCs, the chromosome structure is slightly different. As shown in [Figure 18.4](#), for each rule R_m the chromosome contains the indexes j_m, f of the antecedent, for each input variable X_f , and the consequent class C_{jm} .

The dimension of the chromosome, and consequently of the search space, increases with the increase of the number of input variables. Thus, when this number is high, the multi-objective evolutionary algorithm generally needs a large amount of evaluations to adequately explore the search space and therefore achieve good solutions. To reduce the number of evaluations, the RCS strategy can be used. Let J_{IN} be the set of the M_{IN} candidate rules generated, for example, by applying the Wang and Mendel algorithm to the training set. We recall that, in this case, the antecedents of each candidate rule have a condition for each linguistic variable, since the Wang and Mendel algorithm performs no selection of the conditions. The C_{RB} part of the chromosome is a vector of M_{max} pairs $\mathbf{p}_m = (k_m, \mathbf{v}_m)$, where $k_m \in [0, \dots, M_{IN}]$ identifies the index of the rule in J_{IN} selected for the current RB and $\mathbf{v}_m = [v_{m,1}, \dots, v_{m,F}]$ is a binary vector which indicates, for each condition in the rule, if the condition has to be preserved in the rule ($v_{m,f} = 1$) or replaced by a “don’t care” condition ($v_{m,f} = 0$). If $k_m = 0$, the m th rule is not included in the RB. In this way we manage to generate RBs with a lower number of rules than M_{max} . As an example, let us consider $M_{max} = 3$ and let us suppose to have a two-inputs fuzzy model with four rules generated by the Wang and Mendel algorithm and described by the following J_{IN} matrix:

$$J_{IN} = \begin{bmatrix} 3 & 2 & 2 \\ 5 & 4 & 1 \\ 1 & 5 & 5 \\ 2 & 2 & 4 \end{bmatrix}. \quad (12)$$

In the matrix, rows and columns identify rules and linguistic variables, respectively, and each cell (i, j) contains the fuzzy set used in rule R_m for variable X_f . Let us assume that, during the evolutionary process, the C_{RB} chromosome part shown in [Figure 18.5](#) is generated.

k_1	$v_{1,1}$	$v_{1,2}$	k_2	$v_{2,1}$	$v_{2,2}$	k_3	$v_{3,1}$	$v_{3,2}$
1	1	0	3	0	1	0	0	0

Figure 18.5: An example of RCS chromosome.

Then, the corresponding RB will be represented by the following matrix J :

$$J = \begin{bmatrix} 3 & 0 & 2 \\ 0 & 5 & 5 \end{bmatrix}. \quad (13)$$

As regards the representation of the DB parameters, as stated before, in this chapter we consider triangular fuzzy sets. Further, we suppose that the granularities of each linguistic variable X_f , i.e., the value of T_f , is fixed.

Usually, the parameters which codify triangular fuzzy sets can be coded in a real-valued chromosome. For each fuzzy set $A_{f,j}$, three genes are used for codifying the tuples $(a_{f,j}, b_{f,j}, c_{f,j})$, where $f = 1, \dots, F + 1$ for regression problems and $f = 1, \dots, F$ for classification problems, and $j \in [1, T_f]$. In order to reduce the number of parameters to be identified and, consequently, the search space, in Alcalá *et al.* (2007) and Antonelli *et al.* (2009), two different strategies have been adopted. In (Alcalá *et al.*, 2007), a new coding scheme for the DB has been proposed considering the linguistic 2-tuple representation introduced in Herrera and Martínez (2000), which allows the lateral displacement of the support of a fuzzy set and maintains the interpretability at a good level. This scheme is based on the concept of symbolic translation of a fuzzy set, which is a number in the interval $[-0.5, 0.5]$. The interval defines the domain of a fuzzy set when it is moving between its two adjacent fuzzy sets. Let us consider a generic linguistic fuzzy partition $P_f = \{A_{f,1}, \dots, A_{f,T_f}\}$. Formally, we represent the symbolic translation of a fuzzy set $A_{f,j}$ in P_f by means of the 2-tuple notation, $(A_{f,j}, \alpha_{f,j})$, $A_{f,j} \in P_f$, $\alpha_{f,j} \in [-0.5, 0.5]$. The symbolic translation of a fuzzy set involves the lateral displacement of its associated parameters. This coding scheme decreases the complexity of the parameter optimization, since the 3 parameters considered per fuzzy set are reduced to only 1 symbolic translation parameter.

[Figure 18.6](#) shows the symbolic translation of a label represented by the 2-tuple $(A_{f,3}, -0.3)$ together with the associated lateral displacement. When using this representation a real-coded chromosome, as shown in [Figure 18.7](#), is used. The chromosome contains a sequence of real genes $\alpha_{f,j}$, where $\alpha_{f,j}$, with $j \in [1, T_f]$, $f \in [1, F + 1]$ in regression contexts and $f \in [1, F]$ in classification contexts, is the translation parameter associated with the label $A_{f,j}$ of partition P_f .

In the contribution in Antonelli *et al.* (2009), we have exploited a real-coded chromosome for regression problems, which codifies the positions of the centroids of each fuzzy set in each linguistic variable. Indeed, since we adopt strong fuzzy partitions with, for $j = 2, \dots, T_f - 1$, $b_{f,j} = c_{f,j-1}$, and $b_{f,j} = a_{f,j+1}$, in order to define each fuzzy set of the partition it is sufficient to fix the positions of the cores $b_{f,j}$ along the universe U_f of the f th variable (we normalize each variable in $[0,1]$). Since $b_{f,1}$ and b_{f,T_f} coincide with the extremes of the universe, the partition of each linguistic variable X_f is completely defined by $T_f - 2$ parameters. [Figure 18.8](#) shows the chromosome part which consists of $F + 1$ vectors of real numbers: the f th vector contains the $[b_{f,2}, \dots, b_{f,T_f-1}]$ cores which define

the positions of the membership functions for the linguistic variable X_f . To ensure a good integrity level of the membership functions, in terms of order, coverage and distinguishability (Gacto *et al.*, 2011), $\forall j \in [2..T_f - 1]$, we force $b_{f,j}$ to vary in the definition interval $[b_{f,j} - \frac{b_{f,j} - b_{f,j-1}}{2}, b_{f,j} + \frac{b_{f,j+1} - b_{f,j}}{2}]$.

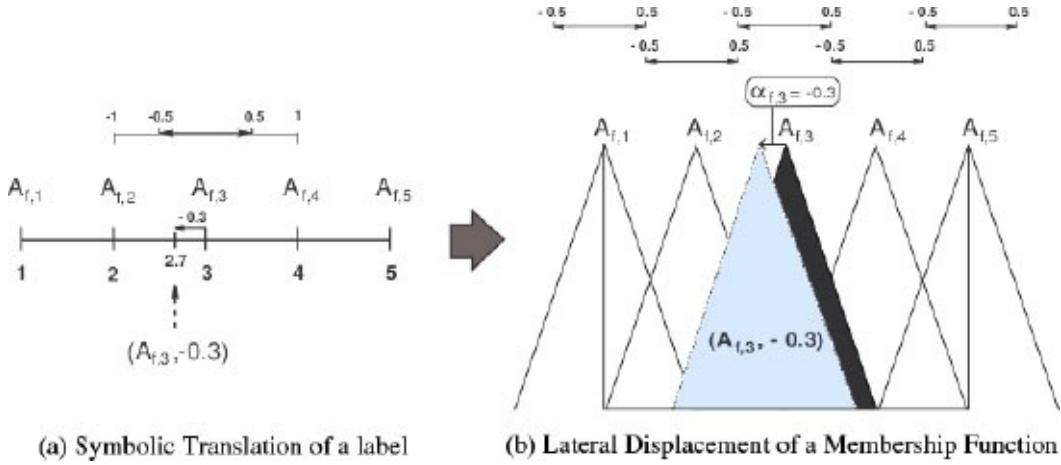


Figure 18.6: Symbolic translation (a) and lateral displacement (b) of the associated fuzzy set.

$\alpha_{1,1}$	\dots	α_{1,T_1}	\dots	$\alpha_{F+1,1}$	\dots	$\alpha_{F+1,T_{F+1}}$
----------------	---------	------------------	---------	------------------	---------	------------------------

Figure 18.7: Chromosome associated with the linguistic 2-tuple representation.

$b_{1,2}$	\dots	b_{1,T_1-1}	\dots	$b_{F+1,2}$	\dots	$b_{F+1,T_{F+1}-1}$
-----------	---------	---------------	---------	-------------	---------	---------------------

Figure 18.8: Real-coded chromosome for strong fuzzy partitions.

18.4.2. Mating Operators

During the evolutionary process of an MOEA, offspring populations, i.e., new solutions from the current solutions, are generated by applying mating operators such as crossover and mutation. Further, in order to select the candidate parents for generating new solutions, appropriate selection strategies are adopted in each specific MOEA.

The half uniform crossover (HUX) and the flip-flop mutation have been used as crossover and mutation operators, respectively, for the binary RS chromosome employed in Alcalá *et al.* (2007). The HUX crossover randomly interchanges the half of the genes that are different in the parents. The flip-flop mutation operator complements each gene with a predefined probability.

For the RL chromosome, operators which allow, respectively, adding, removing or modifying one or more rules have been adopted. For example, in Cococcioni *et al.* (2007), where the chromosome shown in Figure 18.3 has been employed, the following two mutation operators have been defined. The first mutation operator adds γ rules to the RB, where γ is randomly chosen in $[1 \dots \gamma_{max}]$. The upper bound γ_{max} is fixed by the user. If $\gamma + M > M_{max}$, then $\gamma = M_{max} - M$. For each rule R_m added to the chromosome, a random number $\xi \in [1 \dots F]$, which indicates the number of input variables used in the antecedent of the rule, is generated. Then, ξ natural random numbers between 1 and F are generated

to determine the input variables which compose the antecedent part of the rule. Finally, for each selected input variable X_f , a random natural number $j_{m,f}$ between 1 and T_f , which determines the linguistic label $A_{f,jm,f}$ to be used in the corresponding input variable of rule R_m , is generated. To select the consequent linguistic term, a random number in $[1..T_{F+1}]$ is generated. The second mutation operator randomly changes δ antecedent conditions of the RB. The number is randomly generated in $[1..\delta_{max}]$. The upper bound δ_{max} is fixed by the user. For each condition to be modified, a number is randomly generated in $[0..T_f]$, where f is the input variable corresponding to the selected condition.

In (Ducange *et al.*, 2010) we have used the chromosome in [Figure 18.4](#) and, in addition to the previously described mutation operators, have defined an operator which removes λ rules from the RB, where λ is randomly extracted from $[1..\lambda_{max}]$. In the experiments, $\lambda_{max} = \min(\phi_{max} = \min(\phi_{max}, M - M_{min}))$, where ϕ_{max} is fixed by the user, M is the number of rules of RB coded by the chromosome and M_{min} is the minimum number of rules that must be contained in the RB for performing the classification task (at least one per each class).

For both the chromosomes in [Figures 18.3](#) and [18.4](#), the one point crossover has been used in Cococcioni *et al.* (2007) and Ducange *et al.* (2010). This operator has been applied as follows: it cuts the selected parent chromosomes c_1 and c_2 at some chosen common gene and swaps the resulting sub-chromosomes. The common gene is chosen by extracting randomly a number in $[M_{min}, \rho_{min}]$, where ρ_{min} is the minimum number of rules in c_1 and c_2 .

As regards RCS, in Antonelli *et al.* (2013b) we have applied the one-point crossover and have defined two mutation operators. Let c_1 and c_2 be two selected parent chromosomes. For the crossover operator, we choose the common gene by extracting randomly a number in $[1,\rho_{max}-1]$, where ρ_{max} is the maximum number of rules in c_1 and c_2 . The crossover point is always chosen between two rules and not within a rule. The one-point crossover can generate an RB with one or more pairs of equal rules. In this case, we simply eliminate one of the rules from each pair setting the corresponding k_m to zero. This allows us to reduce the total number of rules. The first step for both the mutation operators is to randomly select a rule (i.e., a pair $\mathbf{p}_m = (k_m, \mathbf{v}_m)$) in the chromosome. The first operator replaces the value of k_m in the selected pair with an integer value randomly generated in $[1, \dots, M_{IN}]$, where M_{IN} is the number of candidate rules. If the old value of k_m was equal to zero, the new chromosome will contain an additional rule. The second operator modifies the antecedent \mathbf{v}_m of the selected rule by complementing each gene v_m,f with a probability equal to P_{cond} ($P_{cond} = 2/F$ in the experiments). The two operators are applied with two different probabilities. After applying the two mutation operators, we check the chromosome for duplicate rules in the RB.

Finally, as regards the real-coded chromosome, which codifies the parameters of the membership functions, the BLX- α operator has been applied in a number of contributions (Alcalá *et al.*, 2007; Alcalá *et al.*, 2007; Antonelli *et al.*, 2013b). Let us assume that $c_1 = (x_1 \dots x_g)$ and $c_2 = (y_1 \dots y_g)$, ($x_i, y_i \in \Re, i = 1, \dots, g$), are the two real-coded chromosomes that are going to be crossed. Using the BLX- α crossover, one descendant $z = (z_1, \dots, z_g)$ is obtained, where z_i is randomly (uniformly) generated within the interval $[l_i, u_i]$, with $l_i = \max\{a_i, c_{min} - \chi\}$, $u_i = \min\{b_i, c_{max} + \chi\}$, $c_{min} = \min\{x_i, y_i\}$, $c_{max} = \max\{x_i, y_i\}$, and $\chi = (c_{max} - c_{min}) \cdot \alpha$. As regards the mutation operator, it is often defined an operator that simply changes a gene value at random.

18.4.3. Fitness Functions

In the context of MOEFSs, accuracy and interpretability are two objectives that usually are taken into consideration in designing FRBSs. The two objectives are in conflict: indeed, an increase in system accuracy generally corresponds to a decrease in interpretability and vice versa. In the literature, several definitions of accuracy and interpretability can be found.

As regards the accuracy, the definition of a specific measure depends on the type of problem, namely regression or classification, to be tackled. In the case of regression problems, the approximation error between the actual and the estimated outputs is usually used to evaluate the accuracy of the FRBSs. For example, in (Alcalá *et al.*, 2007; Antonelli *et al.*, 2013b) the half of the mean squared error ($MSE/2$) has been adopted. $MSE/2$ is defined as:

$$MSE/2 = \frac{1}{2 \cdot |S|} \sum_{l=1}^{|S|} (x_{F+1}^l - \hat{x}_{F+1}^l)^2, \quad (14)$$

where $|S|$ is the cardinality of the dataset, \hat{x}_{F+1}^l is the output obtained by the FRBS when the l th input pattern is considered, and x_{F+1}^l is the desired output.

When dealing with classification problems, the classification rate (CR) is typically used as accuracy measure. CR is calculated as follows:

$$CR = 100 \cdot \frac{NCC}{|S|}, \quad (15)$$

where $|S|$ is the cardinality of the dataset and NCC is the number of patterns correctly classified. Sometimes, in order to optimize the accuracy of the generated solutions, during the evolutionary process, the error rate (ER) is directly minimized (Ishibuchi and Nojima, 2007). ER is computed as follows:

$$ER = 100 - CR. \quad (16)$$

In the framework of binary classifiers, i.e., when the number of classes is equal to 2, the Receiver Operating Characteristic (ROC) curve analysis (Fawcett, 2006) is typically adopted to evaluate the capability of correctly classifying patterns. Indeed, especially in the case of imbalanced datasets (Ducange *et al.*, 2010), the True positive rate (TPR) and the False positive rate (FPR) are calculated instead of *CR*. These two metrics are defined as follows:

$$TPR = \frac{TP}{TP + FN}, \quad FPR = \frac{FP}{TN + FP}, \quad (17)$$

where TP = true positive (number of positives correctly classified), TN = true negative (number of negatives correctly classified), FP = false positive (number of negatives classified as positives), FN = false negative (number of positives classified as negatives). TPR and FPR coincide, respectively, with the sensitivity and the complement to 1 of the specificity.

Actually, another measure which aggregates TPR and FPR , namely the area under the curve (AUC), can be also used for evaluating the accuracy of a single binary classifier. The AUC is computed as:

$$AUC = \frac{1 + TPR - FPR}{2}. \quad (18)$$

As regards the interpretability measure, a common approach is to distinguish between interpretability of the RB, also known as complexity, and interpretability of the fuzzy partitions, also known as integrity of the DB.

The most used measures of complexity are the *total number of rules* in the RB (Alcalá *et al.*, 2007; Ishibuchi and Nojima, 2007) and the *sum of the conditions* which compose the antecedents of the rules in the RB (Antonelli *et al.*, 2009; Cococcioni *et al.*, 2007; Ishibuchi and Nojima, 2007), also known as *total rule length*. As regards the integrity of the DB, there is no agreement regarding the choice of an appropriate measure. A deep discussion regarding this topic can be found in Gacto *et al.* (2011).

For the sake of brevity, in this chapter we just discuss a simple integrity index (I_{int}) that we have introduced in Antonelli *et al.* (2011b). I_{int} measures the similarity between the partitions learned during the evolutionary process and the initial interpretable partitions defined by an expert. Let $b_{f,j}$ and $\tilde{b}_{f,j}$ be the cores of the fuzzy sets of, respectively, the current partition $P_f = \{A_{f,1}, \dots, A_{f,T_f}\}$ and the initial partition $\tilde{P}_f = \{\tilde{A}_{f,1}, \dots, \tilde{A}_{f,T_f}\}$ defined by an expert for the linguistic variable X_f . The index is defined as follows:

$$I_{int} = 1 - \frac{D}{D_{max}}, \quad (19)$$

where $D = \frac{1}{F+1} \sum_{f=1}^{F+1} \sum_{j=1}^{T_f-1} |b_{f,j} - \tilde{b}_{f,j}|$ is a dissimilarity measure which expresses how much, on average, the partitions generated by the multi-objective evolutionary optimization are different from the initial partitions, and D_{max} is the possible maximum value of D . We suppose that $\tilde{b}_{f,j}$ identifies the core of the j th fuzzy set of the f th initial partition $\tilde{P}_f = \{\tilde{A}_{f,1}, \dots, \tilde{A}_{f,T_f}\}$ defined by an expert for the linguistic variable X_f .

The value of I_{int} ranges in $[0, 1]$. If the transformed partitions are equal to the initial partitions, then $I_{int} = 1$. The higher the difference between the initial partitions and the transformed partitions, the lower the value of I_{int} (at the minimum, $I_{int} = 0$). If all the partitions are uniform and $T_f = \hat{T}$ for each linguistic variable X_f , from simple mathematical considerations, we derive that $D_{max} = \frac{1}{2}(\hat{T} - 2)$.

18.4.4. An Example of MOEFS Implementation

In [Figure 18.9](#), we show a pseudo-code which describes an example of implementation of an MOEFS. The pseudo-code basically follows the $(2 + 2)M$ -PAES optimization scheme.

At the beginning, the archive is empty and two initial current solutions c_1 and c_2 are randomly generated. The core of the pseudo-code shows how, at each iteration, two candidate solutions s_1 and s_2 are generated from current solutions c_1 and c_2 by using crossover and mutation operators. In particular, we show a general scheme in which we suppose that both the RB and the DB are concurrently optimized and a chromosome C composed of two parts (C_{RB} , C_{DB}) is exploited. In the example, we suppose that two different crossover operators are applied to the C_{RB} and C_{DB} , with probabilities equal to P_{CRB} and P_{CDB} , respectively. Further, H specific mutation operators are supposed to be independently applied to C_{RB} . The h th mutation operators, with $h \in [1, H]$, is applied with a specific probability set to P_{MRBh} . As regards C_{DB} , we suppose that just one mutation operator is applied with a probability equal to P_{MDB} . Obviously, additional mutation operators for C_{DB} can be easily added to the learning scheme.

```

(2+2)M-PAES(TrainingSet)
begin
    initialize(archive);
    [c1, c2] = random_generation();
    loop i=1,...,z
        s1 = c1
        s2 = c2
        if (rand() < PCRB)
            [s1.CBB,s2.CBB]=crossover_CBB (c1.CBB,c2.CBB);
        endif
        if (rand() < PCDB)
            [s1.CDB,s2.CDB]=crossover_CDB (c1.CDB,c2.CDB);
        endif
        loop i=1,2
            if (rand() < PMRBI)
                s1.CBB=RB_mutation_operator_1(s1.CBB);
            endif
            .
            .
            .
            if (rand() < PMRBH)
                s1.CBB=RB_mutation_operator_H(s1.CBB);
            endif
            if (rand() < PMDB)
                s1.CDB = DB_mutation_operator(s1.CDB);
            endif
            evaluate(si,TrainingSet)
            if (!dominate(archive,si));
                insert_into_archive(si);
            endif
        endloop
    endloop
    return archive
end

```

Figure 18.9: A pseudo-code which describes an example of MOEFS implementation.

The function *evaluate ()* implements a set of fitness functions for evaluating both the accuracy and the interpretability of each solution. The candidate solutions are added to the archive only if they are dominated by no solution contained in the archive; possible solutions in the archive dominated by the candidate solutions are removed. Typically, the size of the archive is fixed at the beginning of the execution of the (2 + 2)M-PAES. In this case, when the archive is full and a new solution s_i , where $i = 1, 2$, has to be added to the archive, if it dominates no solution in the archive, then we insert s_i into the archive and remove the solution (possibly s_i itself) that belongs to the region with the highest crowding degree (these steps are implemented by the function *insert_into_archive ()*).

18.5. Analyzing and Comparing the Performances of MOEFSs

To evaluate the performance of an MOEFS, typically a number of benchmark datasets are considered. For each dataset, an n-fold cross validation is performed and a number of trials are executed for each fold with different seeds for the random function generator. Then, the results are evaluated along two different perspectives: the effectiveness of the evolutionary process and the generalization capabilities of the generated solutions. The former is generally evaluated by using well-known metrics, such as the hypervolume and the epsilon dominance (Zitzler *et al.*, 2003), commonly used in the literature to assess the quality of Pareto front approximations. Hypervolume and epsilon dominance represent a means to quantify the differences between Pareto front approximations by mapping each set of solutions to a real number. The hypervolume indicator measures the hypervolume of the portion of the objective space that is weakly dominated by a Pareto front approximation. In order to compute this indicator, the objective space must be either bounded or a bounding reference point, that is (at least weakly) dominated by all the solutions, must be defined. The epsilon dominance was proposed in Zitzler *et al.* (2003) and makes direct use of the concept of Pareto dominance. Let us consider two Pareto front approximations A and B, where A dominates B. The epsilon dominance is a measure of the smallest distance one would need to translate every solution in B so that B dominates A. If C is chosen to be a reference Pareto front approximation, such that it dominates the A and B Pareto front approximations, then A and B can be directly compared with each other on the basis of the epsilon dominance with respect to C.

In Antonelli *et al.* (2013b), the computations of the two indicators were performed using the performance assessment package provided in the PISA toolkit Bleuler *et al.* (2003). First, the maximum values of accuracy and total rule length of the Pareto front approximations generated in the 30 trials executed for each of the 3 MOEAs (90 Pareto front approximations in total) are computed in order to obtain the bounds for normalizing in [0, 1] the objectives of each approximation. Then, the objectives are normalized. The hypervolume is calculated by using (1, 1) as reference point. To compute the epsilon dominance, the reference Pareto front approximation consists of the non-dominated solutions among the solutions on the 90 Pareto front approximations. As a consequence of the normalization, the value of the two indicators is normalized in [0, 1].

The generalization capabilities of the generated solutions are evaluated by considering the accuracies on the test set. Since several solutions can lie on the Pareto front approximations, typically only some representative solutions are considered. In our previous papers (Alcalá *et al.*, 2009; Antonelli *et al.*, 2011b) and also in Gacto *et al.* (2010), for each fold and each trial, the Pareto front approximations of each algorithm are computed and the solutions are sorted in each approximation for decreasing accuracies on training set. Then, for each approximation, we select the first (the most accurate), the median, and the last (the least accurate) solutions. We denote these solutions as FIRST,

MEDIAN, and LAST, respectively. Finally, for the three solutions, we compute the mean values over all the folds and trials of the accuracy on the training and test sets, and of the interpretability. On the one side, the three solutions allow us to graphically show the average trend of the Pareto front approximations obtained in the executions performed on the different folds. On the other side, we can analyze how these solutions are able to generalize when applied to the test set.

[Figure 18.10](#) shows an example of FIRST, MEDIAN, and LAST solutions obtained by the execution of an MOEFS for classification problems (Antonelli *et al.*, 2012b). The algorithm has been executed for 3 trials for each fold in a 10-fold cross validation. In the figure, the complexity has been calculated as the total rule length and the accuracy is expressed in terms of *CR*. We can realize how the three solutions allow us to visualize the trend of the average Pareto front approximations. Further, by comparing the accuracies of the three solutions on the training and test sets, we can appreciate whether these solutions, especially the FIRST solution, suffer from overfitting. Indeed, the FIRST solution is in general the most prone to overfitting since it achieves the higher accuracy on the training set.

The use of numeric values for evaluating the effectiveness and quality of the evolutionary process and the use of the three representative solutions for assessing the generalization capabilities of the solutions on the Pareto front approximations allow comparing different MOEFSs along these two perspectives by using statistical tests. We have to distinguish two different cases: pairwise comparisons and multiple comparisons. In both the cases, the MOEFSs are executed on a number of datasets by employing an *n*-fold cross validation for each dataset. As regards the first case, the Wilcoxon signed-rank test (Sheskin, 2003; Wilcoxon, 1945), a non-parametric statistical test that detects significant differences between two sample means (Wilcoxon, 1945), has been extensively used. Let d_i be the difference between the performance scores of the two algorithms on the i th out of N_{ds} datasets. The differences are ranked according to their absolute values. Ranks of $d_i = 0$ are split evenly among the sums; if there is an odd number of them, one is ignored. Let $R^+ = \sum_{d_i > 0} rank(d_i) + \frac{1}{2} \sum_{d_i=0} rank(d_i)$ and $R^- = \sum_{d_i < 0} rank(d_i) + \frac{1}{2} \sum_{d_i=0} rank(d_i)$ be the sum of ranks for the datasets on which the first algorithm outperforms the second, and the sum of ranks for the opposite case, respectively. If $\min(R^+, R^-)$ is lower than, or equal to, the value of the distribution of Wilcoxon for N_{ds} degrees of freedom (Table B.12 in Zar (1999)), the null hypothesis of equality is rejected.

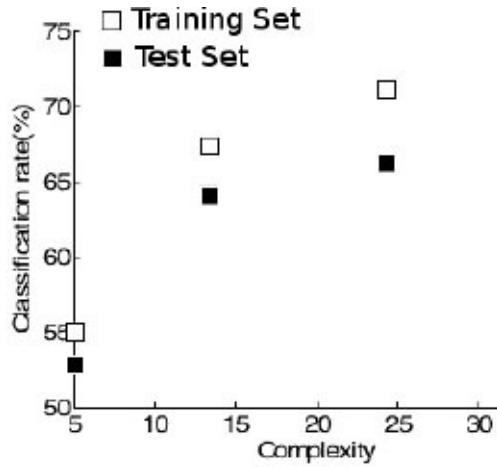


Figure 18.10: An example of Pareto front approximation visualized by using the three representative points.

The Wilcoxon signed-rank test is more sensible than the *t*-test. From the statistical point of view, the test is safer than *t*-test since it does not assume normal distributions. Also, the outliers have less effect on the Wilcoxon test than on the *t*-test.

As regards non-parametric statistical test for multiple comparisons, first a distribution consisting of the mean values of the metric (hypervolume, epsilon dominance, or accuracy) under consideration is generated for each comparison algorithm by using all the datasets. Then, in the MOEFS literature, the Friedman test is generally applied in order to compute a ranking among the distributions (Friedman, 1937), and the Iman and Davenport test (Iman and Davenport, 1980) is used to evaluate whether there exists a statistical difference among the distributions. If the Iman and Davenport *p*-value is lower than the level of significance α (in the experiments $\alpha = 0.05$ is generally adopted), we can reject the null hypothesis and affirm that there exist statistical differences between the multiple distributions associated with each approach. Otherwise, no statistical difference exists. If there exists a statistical difference, a *post hoc* procedure, for instance, the Holm test (Holm, 1979), is applied. This test allows detecting effective statistical differences between the control approach, i.e., the one with the lowest Friedman rank, and the remaining approaches.

18.6. Some Experimental Results

In this section, we show and discuss some experimental results obtained in regression and classification problems. These results are extracted from two (Antonelli *et al.* (2013b) and Antonelli *et al.* (2013a) for, respectively, regressions and classification problems) of our recent papers.

18.6.1. Example of MOEFS Application to Regression Problems

In Antonelli *et al.* (2013b), we have discussed and compared the results achieved by two different MOEFSs based on the $(2 + 2)M$ -PAES algorithm which learn concurrently the RB and the membership function parameters of a set of FRBSs in the same multi-objective evolutionary framework. To this aim, each MOEFS exploits a chromosome C composed of two parts (C_{RB}, C_{DB}), which define the RB and the membership function parameters of the linguistic variables, respectively. We denoted these two MOEFSs as PAES–RCS and PAES–RL, respectively. Both algorithms exploit the C_{DB} chromosome part shown in Figure 18.8. The only difference between the two MOEFSs lies on the generation of the RB: PAES–RCS uses a rule and condition selection scheme and is based on the C_{RB} shown in Figure 18.5, while PAES–RL employs a classical rule learning approach and is based on the C_{RB} shown in Figure 18.3. Further, both MOEFSs consider the half of the MSE , calculated as in Equation (14), and the *total rule length*, calculated as the sum of the conditions which compose the antecedents of the rules in the RB, as objectives to be optimized during the evolutionary process.

We tested the different MOEFSs on twelve regression datasets extracted from the Keel (available at <http://sci2s.ugr.es/keel/datasets.php>) and TorgoâLTMs (available at <http://www.liaad.up.pt/ltorgo/Regression/DataSets.html>) repositories. As shown in Table 18.1, the datasets are characterized by different numbers (from 7 to 40) of input variables and different numbers (from 4052 to 40768) of input/output instances.

Table 18.1: Regression datasets used in the experiments.

Dataset	#Instance	#Input Variables	Repository
Delta Ailerons (DA)	7129	6	Torgo
Delta Elevators (DE)	9517	6	Torgo
Analyzing Categorical Data (AN)	4052	7	Keel
Kinematics (KI)	8192	8	Torgo
Pumadyn (PM)	8192	8	Torgo
California Housing (CH)	20460	8	Keel
Abalone (AB)	4177	9	Keel
MV Artificial Domain (MV)	40768	10	Keel
House_16H (HO)	22784	16	Keel
Elevators (EL)	16559	18	Keel
Computer activity (CA)	8192	21	Keel
Ailerons (AI)	13750	40	Keel

For each dataset and for each algorithm, we carried out a five-fold cross-validation and executed 6 trials for each fold with different seeds for the random function generator

(30 trials in total). The values used in the experiments for the parameters of PAES–RCS and PAES–RL can be found in Antonelli *et al.* (2013b). In particular, the archive size, the number of fuzzy sets for each variable, the minimum and maximum numbers of rules were the same for both PAES–RCS and PAES–RL.

We executed 50,000 and 300,000 fitness evaluations of PAES–RCS and PAES–RL, respectively. In the comparison, we also considered the results generated by PAES–RL after 50,000 fitness evaluations. We denoted PAES–RL executed for 50,000 and 300,000 evaluations as PAES–RL50 and PAES–RL300, respectively.

To compare PAES–RCS with PAES–RL we adopted the epsilon dominance and the hypervolume. The aim of this analysis was to show that PAES–RCS, though performing only 50,000 fitness evaluations, achieved Pareto front approximations comparable with the ones achieved by PAES–RL300, thus confirming the effectiveness of the RCS approach in speeding up the convergence. Further, the solutions were also compared in terms of accuracy and complexity by using the FIRST, MEDIAN, and LAST solutions.

[Table 18.2](#) shows the results of the non-parametric statistical tests executed on the epsilon dominance and the hypervolume: for each indicator and for each MOEFS, we show the Iman and Davenport *p-value* and the Friedman rank.

For both the indicators, the Iman and Davenport statistical hypothesis of equivalence is rejected and so statistical differences among the three MOEFSs are detected. Thus, we applied the Holm *post hoc* procedure considering PAES–RL300 as control algorithm (associated with the lowest rank and in bold in the Table). For both the indicators, we observed that the statistical hypothesis of equivalence was rejected for PAES–RL50, but not for PAES–RCS.

By analyzing the results of the statistical tests, we deduced that PAES–RL needs a larger number of fitness evaluations to converge to Pareto front approximations comparable to the ones achieved by PAES–RCS after 50,000 evaluations. On the other hand, these results can be explained by analyzing the different sizes of the search spaces handled by the two MOEFSs. Indeed, PAES–RCS selects rules and conditions of these rules from an initial set of candidate rules generated by the Wang and Mendel algorithm: thus, rules are learned in a very constrained space. Unlike PAES–RCS, PAES–RL learns the rules by exploring a space consisting of all the possible valid combinations of propositions which can be generated from the linguistic values defined on the universes of the input and output variables. This space rapidly becomes difficult to manage with the increase of the number of input variables.

Table 18.2: Results of the statistical tests on epsilon dominance and hypervolume among PAES–RL50, PAES–RCS, and PAES–RL300.

Epsilon dominance				
	Algorithm	Friedman Rank	Iman and Davenport p-value	Hypothesis
	PAES-RL50	2.75		
	PAES-RCS	1.916	4.19E-04	Rejected
	PAES-RL300	1.333		
Holm Post hoc Procedure				
i	Algorithm	z-value	p-value	alpha/i
2	PAES-RL50	3.47	5.20E-04	0.025
1	PAES-RCS	1.429	1.53E-01	0.05
Hypervolume				
	Algorithm	Friedman Rank	Iman and Davenport p-value	Hypothesis
	PAES-RL50	2.667		
	PAES-RCS	1.917	3.91E-03	Rejected
	PAES-RL300	1.417		
Holm Post hoc Procedure				
i	Algorithm	z-value	p-value	alpha/i
2	PAES-RL50	3.061	2.20E-03	0.025
1	PAES-RCS	1.224	2.21E-01	0.05

To evaluate the generalization capabilities of the FRBSs on the Pareto front approximation, we considered the accuracy of the three representative solutions. In Figures 18.11 and 18.12, we plot the mean values of the MSE/2 and the complexity measured in terms of total rule length for the FIRST, MEDIAN, and LAST solutions for all the datasets, on both the training and test sets. We observe that PAES- RL300 generates more complex solutions than PAES-RL50 and PAES-RCS, except for datasets with a high number of input variables, namely EL, CA, and AI. For almost all the datasets the Pareto front approximations generated by PAES-RL50 are dominated by the Pareto front approximations generated by both PAES-RL300 and PAES-RCS, thus highlighting that PAES-RCS generates better trade-off solutions than PAES-RL, both on training and test sets, when the same number of fitness evaluations are performed.

In order to assess if there exist statistical differences among the MSE/2 corresponding to the representative solutions generated by PAES-RCS and the two versions of PAES-RL, we applied non-parametric statistical tests by combining all the datasets: for each approach and for each of the three representative solutions we generated a distribution consisting of the mean values of the MSE/2 on the test set. Table 18.3 shows the results of the statistical tests. For all the three solutions, the Iman and Davenport test rejected the statistical hypothesis of equivalence. Then, the Holm *post hoc* procedure was performed by considering as control algorithms PAES-RL300 for the FIRST and MEDIAN solutions and PAES-RCS for the LAST solutions. For all the three solutions, the statistical hypothesis of equivalence was rejected only for PAES-RL50. Thus, we concluded that PAES-RL300 and PAES- RCS result to be statistically equivalent in terms of MSE/2

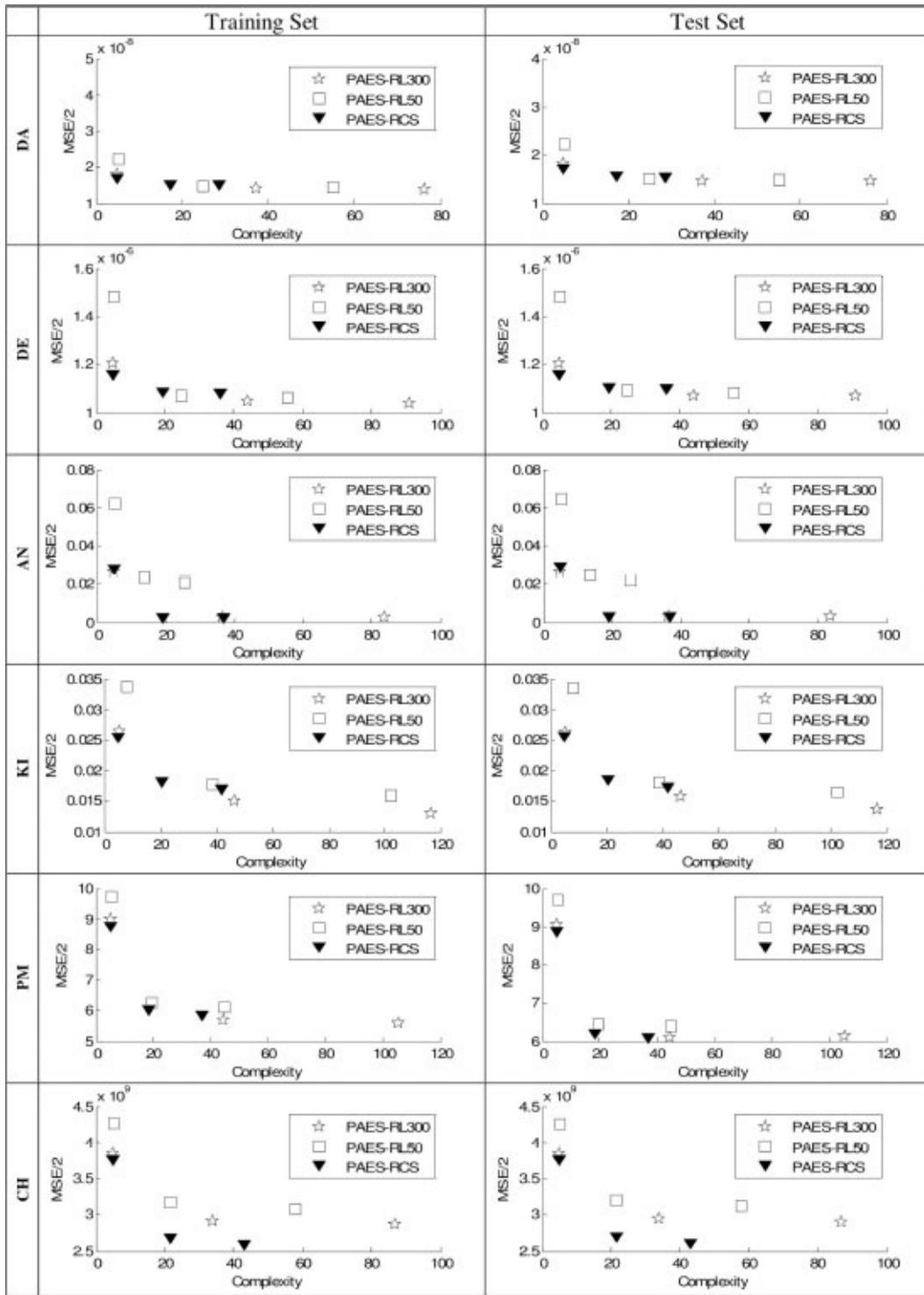


Figure 18.11: Plots of the FIRST, MEDIAN, and LAST solutions onto the Complexity-MSE/2 plane (first six datasets).

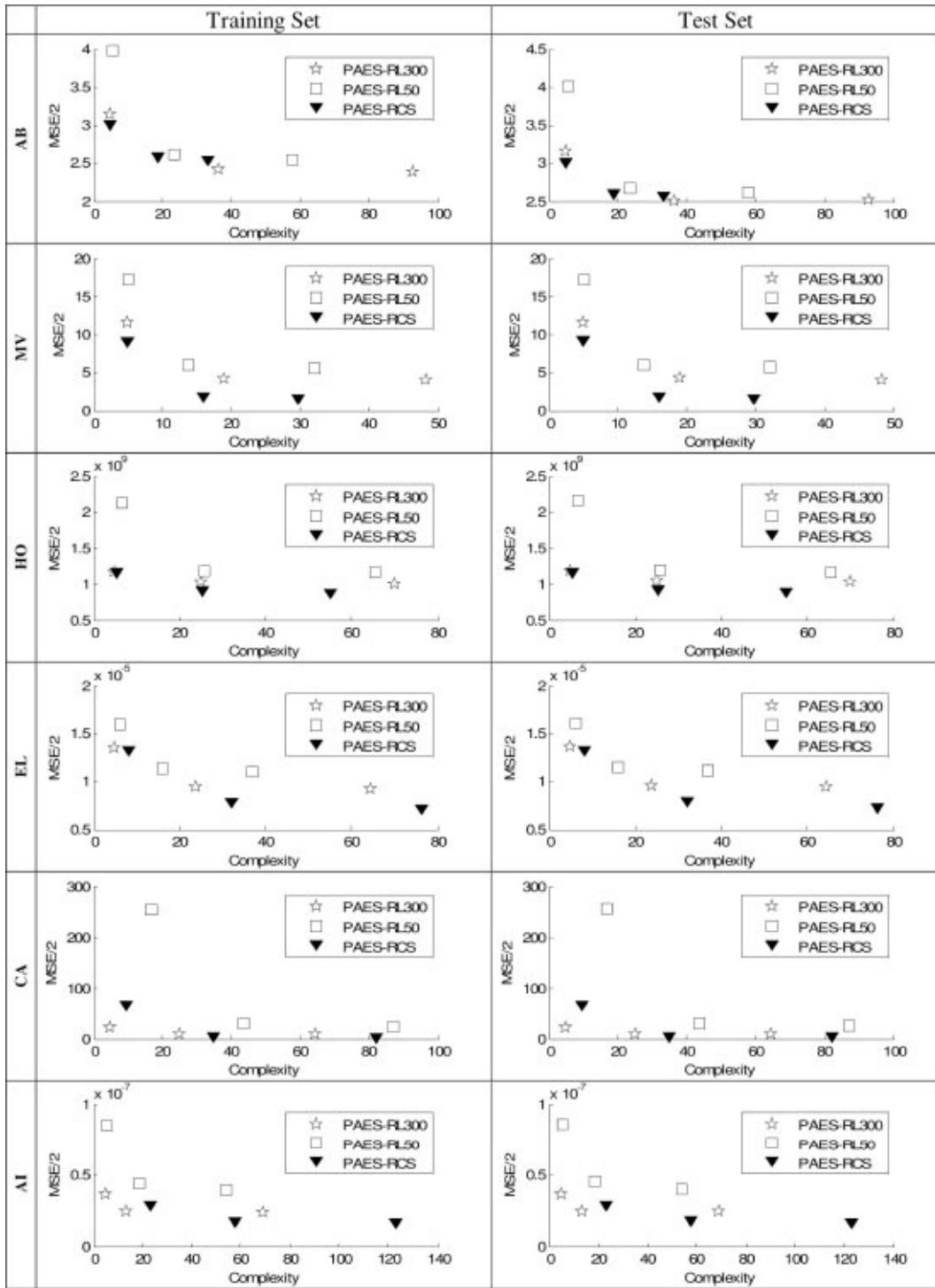


Figure 18.12: Plots of the FIRST, MEDIAN, and LAST solutions onto the Complexity-MSE/2 plane (last six datasets).

Table 18.3: Results of the statistical tests on the FIRST, MEDIAN, and LAST solutions applied to the test sets for PAES-RL50, PAES-RCS, and PAES-RL300.

FIRST					
	Algorithm	Friedman Rank	Iman and Davenport p-value	Hypothesis	
	PAES-RL50	2.75			
	PAES-RCS	1.667	2.33E-03	Rejected	
	PAES-RL300	1.583			
<i>Holm Post hoc Procedure</i>					
<i>i</i>	Algorithm	<i>z-value</i>	<i>p-value</i>	<i>alpha/i</i>	<i>Hypothesis</i>
2	PAES-RL50	2.858	4.27E-03	0.025	Rejected
1	PAES-RCS	0.204	8.38E-01	0.05	Not Rejected
MEDIAN					
	Algorithm	Friedman Rank	Iman and Davenport p-value	Hypothesis	
	PAES-RL50	2.75			
	PAES-RCS	1.749	1.78E-03	Rejected	
	PAES-RL300	1.5			
<i>Holm Post hoc Procedure</i>					
<i>i</i>	Algorithm	<i>z-value</i>	<i>p-value</i>	<i>alpha/i</i>	<i>Hypothesis</i>
2	PAES-RL50	3.061	2.20E-03	0.025	Rejected
1	PAES-RCS	0.612	5.40E-01	0.05	Not Rejected
LAST					
	Algorithm	Friedman Rank	Iman and Davenport p-value	Hypothesis	
	PAES-RL50	3			
	PAES-RCS50	1.167	3.71E-10	Rejected	
	PAES-RL300	1.833			
<i>Holm Post hoc Procedure</i>					
<i>i</i>	Algorithm	<i>z-value</i>	<i>p-value</i>	<i>alpha/i</i>	<i>Hypothesis</i>
2	PAES-RL50	4.49	7.10E-06	0.025	Rejected
1	PAES-RL300	1.632	1.03E-01	0.05	Not Rejected

The results on the test set confirm that PAES–RCS after 50,000 fitness evaluations generates on average solutions with MSE/2 similar to the ones generated by PAES–RL after 300,000 evaluations and that the MSE/2 of the solutions achieved by PAES–RL after 50,000 fitness evaluations are not statistically equivalent to the ones obtained by PAES–RCS and PAES–RL300.

18.6.2. Example of MOEFs Application to Classification Problems with Imbalanced Datasets

In Antonelli *et al.* (2013a), we have proposed the application of a multi-objective evolutionary learning scheme for generating FRBCs for imbalanced datasets. In particular, the RB and the membership function parameters of a set of FRBCs are concurrently learned by optimizing the sensitivity, the specificity, and the complexity. We recall that imbalanced datasets are characterized by classification categories not equally represented. These datasets are particularly interesting since lots of real world classification problems are often associated with data samples which consist of a large percentage of normal examples (negative classes) and a small percentage of atypical or relevant examples (positive classes).

The proposed MOEFS, denoted as PAES-3obj in Antonelli *et al.* (2013a), for imbalanced datasets is based on (2 + 2)M-PAES and exploits a chromosome C composed of two parts (C_{RB}, C_{DB}), where C_{RB} and C_{DB} are shown, respectively, in Figures 18.4 and 18.8. The MOEFS concurrently optimizes three objectives, namely TPR , FPR , calculated as in Equation (17), and complexity, expressed in terms of total rule length.

We compared the results achieved by PAES-3obj with the ones achieved by two recent single-objective evolutionary fuzzy classifiers (EFCs) appropriately designed for imbalanced datasets. The first EFC (Villar *et al.*, 2012) performs an evolutionary database learning with an embedded rule-base generation. The second EFC (Lopez *et al.*, 2013) builds a hierarchical FRBC: first, a genetic programming algorithm is used to learn the rule-base and then a post-process, which includes a genetic rule selection and a membership function parameters tuning, is applied to the generated FRBC. We denote these two comparison approaches as GA-FS + GL and GP-COACH-H, respectively. We recall that both the comparison approaches rebalance the training set using a state-of-the-art oversampling method, namely SMOTE (Chawla *et al.*, 2002).

We evaluated the performance of the three algorithms on the 22 classification datasets used in Villar *et al.* (2012) and extracted from the KEEL repository (available at <http://sci2s.ugr.es/keel/datasets.php>). We used a five-fold cross validation as in Villar *et al.* (2012) and in Lopez *et al.* (2013). As shown in Table 18.4, the datasets are characterized by different numbers of input variables (from 7 to 13) and input/output instances (from 184 to 4174), and by different values of the imbalance ratio (IR). The IR is defined as the ratio between the number of instances of the majority and minority classes.

In order to analyze the results of PAES-3obj, each three-dimensional Pareto front approximation was projected onto the FPR–TPR plane: each FRBC of the Pareto front approximation was represented as a point corresponding to its pair (FPR, TPR). These pairs give origin to the so-called ROC curve [Fawcett (2006)]. We recall that the ROC analysis is typically adopted to compare different binary classifiers with each other. In particular, one classifier in the ROC space is better than (dominates) another if it is located more North–West (higher TPR and/or lower FPR) than the other. For this reason, in order to select a set of potentially optimal FRBCs, we extracted the non-dominated solutions obtained on the training set in the FPR–TPR plane and used only these solutions for plotting the ROC curve. Further, to evaluate the overall quality of our approach, we calculated the AUC.

Table 18.4: Datasets used in the experiments for classification problems.

Dataset	# Instances	# Variables	IR
Abalone19	4174	8	128.87
Abalone9vs18	731	8	16.68
<i>E.coli</i> 0137vs26	281	7	39.15
<i>E.coli</i> 4	336	7	13.84
Glass016vs2	192	9	10.29
Glass016vs5	184	9	19.44
Glass2	214	9	10.39
Glass4	214	9	15.47
Glass5	214	9	22.81
Page-blocks13vs2	472	10	15.85
Shuttle0vs4	1829	9	13.87
Shuttle2vs4	129	9	20.5
Vowel0	988	13	10.1
Yeast05679vs4	528	8	9.35
Yeast1289vs7	947	8	30.56
Yeast1458vs7	693	8	22.1
Yeast1vs7	459	8	13.87
Yeast2vs4	514	8	9.08
Yeast2vs8	482	8	23.1
Yeast4	1484	8	28.41
Yeast5	1484	8	32.78
Yeast6	1484	8	39.15

As regards GA-FS + GL and GP-COACH-H, they do not generate a set of FRBCs characterized by different values of TPR and FPR but they just provide one classifier with its associated sensitivity and specificity. In this case, the ROC curve is generated by connecting the point corresponding to the unique FRBC to (0, 0) and (1, 1), respectively. The corresponding AUC is computed as in Equation (18).

[Table 18.5](#) shows, for each dataset, the average values of the AUCs computed on the test set for the solutions generated by GA-FS + GL, GP-COACH-H, and PAES-3obj. We point out that the ROC curve on the test set is obtained by connecting the points selected in the training set. For each dataset, the values of the highest AUCs are shown in bold. We observe that in most of the datasets, the solutions generated by PAES-3obj are associated with the highest values of AUC. Further, the solutions generated by GA-FS + GL and GP-COACH-H are characterized, on average, by similar values of AUC.

Table 18.5: Average values of the AUC computed on the test set.

Dataset	GA-FS + GL	GP-COACH-H	PAES-3obj
Abalone19	0.699	0.594	0.727
Abalone9-18	0.628	0.780	0.811
<i>E.coli</i> 0137vs26	0.790	0.898	0.892
<i>E.coli</i> 4	0.860	0.934	0.909
glass016vs2	0.626	0.646	0.662
glass016vs5	0.867	0.871	0.885
Glass2	0.729	0.609	0.705
Glass4	0.853	0.793	0.864
Glass5	0.755	0.856	0.898
Page-Blocks13vs4	0.937	0.884	0.951
shuttle0vs4	0.999	0.996	0.996
shuttle2vs4	0.992	1.000	0.937
Vowel0	0.928	0.946	0.903
Yeast05679vs4	0.793	0.785	0.835
Yeast1289vs7	0.750	0.696	0.688
Yeast1458vs7	0.641	0.588	0.645
yeastB1vs7	0.754	0.681	0.775
Yeast2vs4	0.884	0.897	0.944
Yeast2vs8	0.710	0.731	0.814
Yeast4	0.835	0.813	0.868
Yeast5	0.935	0.928	0.952
Yeast6	0.870	0.880	0.888
Mean	0.811	0.809	0.843

To statistically validate the previous observations, we applied non-parametric statistical tests for multiple comparisons among the three algorithms by combining all the datasets. [Table 18.6](#) shows the results of the non-parametric statistical tests. With a significance level lower than 0.05, we observed that the Iman and Davenport statistical hypothesis of equivalence is rejected and so statistical differences among the three algorithms were detected. Thus, we applied the Holm post-hoc procedure considering the PAES-3obj as control algorithm (associated with the lowest rank and in bold in the Table). The statistical hypothesis of equivalence can be rejected for both GA-FS + GL and GP-COACH-H, thus confirming that PAES-3obj outperforms, in terms of AUC on the test set, the other algorithms.

The main advantage of PAES-3obj is that, unlike GA-FS + GL and GP-COACH-H, the training set does not need to be re-balanced by using the SMOTE oversampling algorithm. This advantage stands out especially when dealing with datasets characterized by a large number of instances and a high IR. Indeed, the oversampling of the minority class becomes computationally expensive, the size of the dataset almost doubles and, consequently, the learning time increases.

Table 18.6: Results of the non-parametric statistical tests on the AUC computed on the test set.

<i>Algorithm</i>	<i>Friedman rank</i>	<i>Iman and Davenport p-value</i>	<i>Hypothesis</i>
GA-FS + GL	2.272		
GP-COACH-H	2.227	0.0126	Rejected
PAES-3obj	1.499		
		<i>Holm post-hoc procedure</i>	
<i>i</i>	<i>Algorithm</i>	<i>z-value</i>	<i>p-value</i>
2	GA-FS + GL	2.562	0.0103
1	GP-COACH-H	1.809	0.0158

Table 18.7: Average values of the complexity and number of rules for GA-FS + GL and PAES-3obj.

Dataset	GA-FS + GL	PAES-3obj		PAES-3obj		Comp
		#Rules	Comp	(Min)	#Rules	
Abalone19	7.40	14.80	2.70	8.30	12.87	66.60
Abalone9-18	4.40	12.32	2.60	8.03	12.10	59.70
<i>E.coli</i> 0137vs26	4.20	10.08	2.43	3.17	7.63	24.93
<i>E.coli</i> 4	58.20	162.96	2.33	3.53	6.67	25.00
Glass016vs2	3.60	10.80	2.93	8.73	11.00	58.43
Glass016vs5	4.00	12.00	2.10	2.57	5.00	14.97
Glass2	8.20	24.60	2.93	8.73	8.87	44.27
Glass4	8.40	26.88	2.83	6.60	7.20	30.30
Glass5	5.20	15.60	2.43	4.37	5.77	19.47
Page-Blocks13vs4	6.60	21.12	2.63	7.80	6.53	34.17
shuttle0vs4	14.80	32.56	2.00	2.00	3.27	10.70
shuttle2vs4	4.20	12.60	2.00	2.00	2.10	2.13
Vowel0	4.40	22.88	2.30	8.77	8.37	58.97
Yeast05679vs4	9.40	20.68	2.47	6.60	11.33	53.60
Yeast1289vs7	13.80	41.40	3.00	10.10	11.60	56.07
Yeast1458vs7	3.80	8.36	2.73	8.73	11.00	55.80
yeast1vs7	6.80	19.04	2.53	6.60	10.93	47.77
Yeast2vs4	2.20	6.60	2.57	6.17	8.83	40.90
Yeast2vs8	5.00	27.00	2.53	5.80	8.57	40.43
Yeast4	6.40	17.92	2.23	5.47	10.50	51.67
Yeast5	4.80	14.40	3.00	8.67	9.33	44.13
Yeast6	5.00	14.00	3.40	10.70	11.40	55.37
Mean	8.67	24.94	2.58	6.52	8.68	40.70

As an example of the FRBCs that can be generated by using PAES-3obj, we extracted the solution with the highest AUC from the Pareto front approximation of a specific fold of the *E.coli*4 dataset. In [Figure 18.13](#), we show the overall KB composed by the linguistic fuzzy partitions, associated with the input variables, and the fuzzy RB. In this figure, we represent the initial and the optimized fuzzy partitions by gray dashed lines and solid black lines, respectively. We denote as L, M, and H, the labels of the fuzzy sets which represent, respectively, the low, medium and high levels of each linguistic variable universe. Further, “1” and “2” are the labels of, respectively, the minority and the majority classes, and “-” represents the “don’t care” condition.

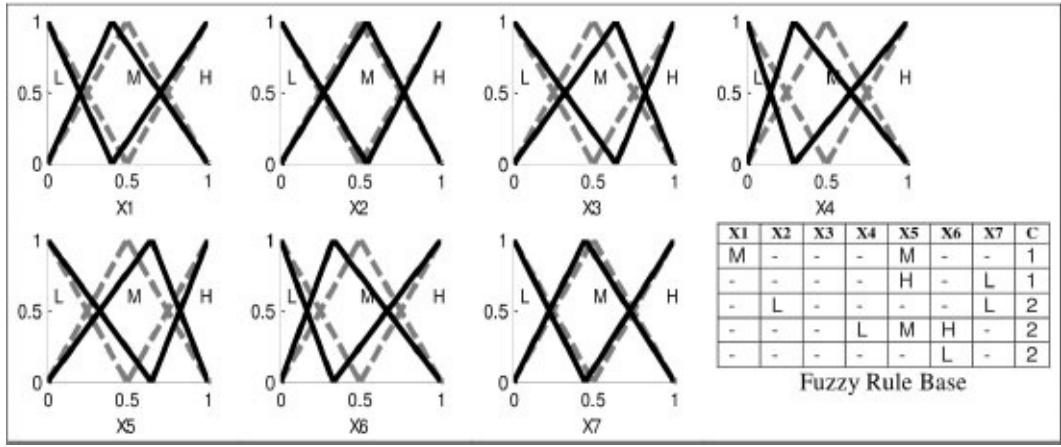


Figure 18.13: An example of database and rule-base generated by PAES-3obj.

We observe that, even though the optimized fuzzy partitions are slightly different from the initial ones, they continue to be interpretable (order, coverage and distinguishability are maintained). As regards the RB, we note that it is characterized by a reduced number of rules and conditions in the rules, thus ensuring a very high readability of the final RB.

18.7. Conclusions and Current Research Trends

In this chapter, we have presented the main features of MOEFSs, which have been proposed in the last years for both classification and regression problems. In particular, we have discussed the chromosome coding, mating operators, objective functions, and multi-objective evolutionary algorithms that have been used in most of these systems. Further, we have shown how the results of MOEFSs can be analyzed and compared. Finally, we have discussed some experimental results obtained by applying MOEFSs to both regression and classification datasets. Although MOEFSs can be considered as a mature research area, there still exist some challenges which have to be investigated. One of the most critical aspects, which limits the use of MOEFSs, is the effort needed for their execution. This effort is strongly affected by the computation of the fitness, especially when the dataset is large. To reduce the computational cost, some solutions have been investigated such as employing parallel evolutionary algorithms, exploiting fitness approximation (Cococcioni *et al.*, 2011) and adopting instance selection techniques (Antonelli *et al.*, 2012a; Fazzolari *et al.*, 2013b). Another critical aspect is to manage high-dimensional datasets since the search space increases with the number of features. To limit this drawback, feature selection during the evolutionary process and ad-hoc modified MOEAs have being explored (Alcalá *et al.*, 2011). Finally, classical MOEAs, which have been used so far in MOEFSs, are not suitable for managing more than three objectives. Since interpretability is now evaluated by using a number of indexes, which can be used as different objectives in the optimization process, appropriate algorithms recently proposed in the multi-objective evolutionary framework for dealing with a large number of objectives will have to be investigated and experimented.

References

- Alcalá, R., Alcalá-Fdez, J. and Herrera, F. (2007). A proposal for the genetic lateral tuning of linguistic fuzzy systems and its interaction with rule selection. *IEEE T. Fuzzy Syst.*, 15(4), pp. 616–635.
- Alcalá, R., Ducange, P., Herrera, F., Lazzerini, B. and Marcelloni, F. (2009). A multiobjective evolutionary approach to concurrently learn rule and data bases of linguistic fuzzy-rule-based systems. *IEEE T. Fuzzy Syst.*, 17(5), pp. 1106–1122.
- Alcalá, R., Gacto, M. J., Herrera, F. and Alcalá-Fdez, J. (2007). A multi-objective genetic algorithm for tuning and rule selection to obtain accurate and compact linguistic fuzzy rule-based systems. *Int. J. Uncertain. Fuzz.*, 15(5), pp. 539–557.
- Alcalá, R., Gacto, M. J. and Herrera, F. (2011). A fast and scalable multiobjective genetic fuzzy system for linguistic fuzzy modeling in high-dimensional regression problems. *IEEE T. Fuzzy Syst.*, 19(4), pp. 666–681.
- Alonso, J. M., Magdalena, L. and González-Rodríguez, G. (2009). Looking for a good fuzzy system interpretability index: An experimental approach. *Int. J. Approx. Reason.*, 51(1), pp. 115–134.
- Antonelli, M., Ducange, P., Lazzerini, B. and Marcelloni, F. (2009). Multi-objective evolutionary learning of granularity, membership function parameters and rules of Mamdani fuzzy systems, *Evol. Intell.*, 2(1–2), pp. 21–37.
- Antonelli, M., Ducange, P., Lazzerini, B. and Marcelloni, F. (2011a). Learning concurrently data and rule bases of Mamdani fuzzy rule-based systems by exploiting a novel interpretability index. *Soft Comput.*, 15(10), pp. 1981–1998.
- Antonelli, M., Ducange, P., Lazzerini, B. and Marcelloni, F. (2011b). Learning knowledge bases of multi-objective evolutionary fuzzy systems by simultaneously optimizing accuracy, complexity and partition integrity. *Soft Comput.*, 15(12), pp. 2335–2354.
- Antonelli, M., Ducange, P. and Marcelloni, F. (2012a). Genetic training instance selection in multi-objective evolutionary fuzzy systems: A co-evolutionary approach. *IEEE T. Fuzzy Syst.*, 20(2), pp. 276–290.
- Antonelli, M., Ducange, P. and Marcelloni, F. (2012b). Multi-objective evolutionary rule and condition selection for designing fuzzy rule-based classifiers. In *2012 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1–7.
- Antonelli, M., Ducange, P., Marcelloni, F. and Segatori, A. (2013a). Evolutionary fuzzy classifiers for imbalanced datasets: An experimental comparison. In *2013 Joint IFSA World Congress and NAFIPS Annual Meeting*, pp. 13–18.
- Antonelli, M., Ducange, P. and Marcelloni, F. (2013b). An efficient multi-objective evolutionary fuzzy system for regression problems. *Int. J. Approx. Reason.*, 54(9), pp. 1434–1451.
- Bleuler, S., Laumanns, M., Thiele, L. and Zitzler, E. (2003). Pisa a platform and programming language independent interface for search algorithms. In Fonseca, C., Fleming, P., Zitzler, E., Thiele, L. and Deb, K. (eds.), *Evolutionary Multi-Criterion Optimization*, Vol. 2632, *Lect. Notes Comput. Sci.* Berlin, Heidelberg: Springer, pp. 494–508.
- Botta, A., Lazzerini, B., Marcelloni, F. and Stefanescu, D. C. (2009). Context adaptation of fuzzy systems through a multi-objective evolutionary approach based on a novel interpretability index. *Soft Comput.*, 13(5), pp. 437–449.
- Casillas, J., Martínez, P. and Benítez, A. (2009). Learning consistent, complete and compact sets of fuzzy rules in conjunctive normal form for regression problems. *Soft Comput.*, 13(5), pp. 451–465.
- Chawla, N. V., Bowyer, K. W., Hall, L. O. and Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.*, 16, pp. 321–357.
- Cococcioni, M., Ducange, P., Lazzerini, B. and Marcelloni, F. (2007). A Pareto-based multi-objective evolutionary approach to the identification of Mamdani fuzzy systems. *Soft Comput.*, 11(11), pp. 1013–1031.
- Cococcioni, M., Lazzerini, B. and Marcelloni, F. (2011). On reducing computational overhead in multi-objective genetic Takagi–Sugeno fuzzy systems. *Appl. Soft Comput.*, 11(1), pp. 675–688.
- Coello, C. A. C., Lamont, G. B. and Van Veldhuisen, D. A. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer.
- Cordon, O., del Jesus, M. J. and Herrera, F. (1999). A proposal on reasoning methods in fuzzy rule-based classification

- systems. *Int. J. Approx. Reason.*, 20(1), pp. 21–45.
- Cordón, O. (2011). A historical review of evolutionary learning methods for Mamdani-type fuzzy rule-based systems: Designing interpretable genetic fuzzy systems. *Int. J. Approx. Reason.*, 52(6), pp. 894–913.
- Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE T. Evol. Comput.*, 6(2), pp. 182–197.
- de Oliveira, J. (1999). Semantic constraints for membership function optimization. *IEEE T. Syst. Man Cy. A*, 29(1), pp. 128–138.
- Ducange, P., Lazzerini, B. and Marcelloni, F. (2010). Multi-objective genetic fuzzy classifiers for imbalanced and cost-sensitive datasets. *Soft Comput.*, 14(7), pp. 713–728.
- Ducange, P. and Marcelloni, F. (2011). Multi-objective evolutionary fuzzy systems. In Fanelli, A., Pedrycz, W. and Petrosino, A. (eds.), *Fuzzy Logic and Applications*, Vol. 6857, *Lect. Notes Comput. Sci.*, Berlin, Heidelberg: Springer, pp. 83–90.
- Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognit. Lett.*, 27(8), pp. 861–874.
- Fazzolari, M., Alcalá, R., Nojima, Y., Ishibuchi, H. and Herrera, F. (2013a). A review of the application of multiobjective evolutionary fuzzy systems: Current status and further directions. *IEEE T. Fuzzy Syst.*, 21(1), pp. 45–65.
- Fazzolari, M., Giglio, B., Alcalá, R., Marcelloni, F. and Herrera, F. (2013b). A study on the application of instance selection techniques in genetic fuzzy rule-based classification systems: Accuracy-complexity trade-off. *Know.-Based Syst.*, 54, pp. 32–41.
- Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *J. Am. Stat. Assoc.*, 32(200), pp. 675–701.
- Gacto, M. J., Alcalá, R. and Herrera, F. (2009). Adaptation and application of multi-objective evolutionary algorithms for rule reduction and parameter tuning of fuzzy rule-based systems. *Soft Comput.*, 13(5), pp. 419–436.
- Gacto, M. J., Alcalá, R. and Herrera, F. (2010). Integration of an index to preserve the semantic interpretability in the multiobjective evolutionary rule selection and tuning of linguistic fuzzy systems. *IEEE T. Fuzzy Syst.*, 18(3), pp. 515–531.
- Gacto, M. J., Alcalá, R. and Herrera, F. (2011). Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures. *Inform. Sci.*, 181(20), pp. 4340–4360.
- Guillaume, S. (2001). Designing fuzzy inference systems from data: An interpretability-oriented review. *IEEE T. Fuzzy Syst.*, 9(3), pp. 426–443.
- Herrera, F. and Martínez, L. (2000). A 2-tuple fuzzy linguistic representation model for computing with words. *IEEE T. Fuzzy Syst.*, 8(6), pp. 746–752.
- Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scand. J. Stat.*, 6(2), pp. 65–70.
- Horn, J., Nafpliotis, N. and Goldberg, D. E. (1994). A niched Pareto genetic algorithm for multiobjective optimization. In *Proc. First IEEE C. Evol. Computat.*, pp. 82–87.
- Iman, R. L. and Davenport, J. M. (1980). Approximations of the critical region of the fbietkan statistic. *Common Stat.*, 9(6), pp. 571–595.
- Ishibuchi, H., Murata, T. and Türksen, I. B. (1997). Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems. *Fuzzy Set. Syst.*, 89(2), pp. 135–150.
- Ishibuchi, H., Nakashima, T. and Murata, T. (2001). Three-objective genetics-based machine learning for linguistic rule extraction. *Inform. Sci.*, 136(1–4), pp. 109–133.
- Ishibuchi, H., Nakashima, T. and Nii, M. (2004). *Classification and Modeling with Linguistic Information Granules: Advanced Approaches to Linguistic Data Mining (Advanced Information Processing)* Secaucus, NJ, USA: Springer-Verlag.
- Ishibuchi, H. and Nojima, Y. (2007). Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning. *Int. J. Approx. Reason.*, 44(1), pp. 4–31.
- Ishibuchi, H. and Yamamoto, T. (2004). Fuzzy rule selection by multi-objective genetic local search algorithms and rule

- evaluation measures in data mining. *Fuzzy Set. Syst.*, 141(1), pp. 59–88.
- Klir, G. J. and Yuan, B. (1995). *Fuzzy Sets and Fuzzy Logic*. New Jersey: Prentice Hall.
- Knowles, J. D. and Corne, D. W. (2000). Approximating the nondominated front using the Pareto archived evolution strategy. *Evol. Comput.*, 8(2), pp. 149–172.
- Lopez, V., Fernandez, A., del Jesus, M. J. and Herrera, F. (2013). A hierarchical genetic fuzzy system based on genetic programming for addressing classification with highly imbalanced and borderline data-sets. *Knowl.-Based Syst.*, 38, pp. 85–104.
- Mencar, C. and Fanelli, A. (2008). Interpretability constraints for fuzzy information granulation. *Inform. Sci.*, 178(24), pp. 4585–4618.
- Nguyen, C. H., Pedrycz, W., Duong, T. L. and Tran, T. S. (2013). A genetic design of linguistic terms for fuzzy rule based classifiers. *Int. J. Approx. Reason.*, 54(1), pp. 1–21.
- Pulkkinen, P. and Koivisto, H. (2010). A dynamically constrained multiobjective genetic fuzzy system for regression problems. *IEEE T. Fuzzy Syst.*, 18(1), pp. 161–177.
- Sheskin, D. (2003). *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall/CRC.
- Villar, P., Fernandez, A., Carrasco, R. A. and Herrera, F. (2012). Feature selection and granularity learning in genetic fuzzy rule-based classification systems for highly imbalanced data-sets. *Int. J. Uncertain. Fuzz.*, 20(03), pp. 369–397.
- Wang, L.-X. and Mendel, J. M. (1992). Generating fuzzy rules by learning from examples. *IEEE T. Syst. Man Cyb.*, 22(6), pp. 1414–1427.
- Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics*, 1, pp. 80–83.
- Zadeh, L. (1965). Fuzzy sets. *Inform. Control*, 8(3), pp. 338–353.
- Zar, J. (1999). *Biostatistical Analysis*. Upper Saddle River, NJ: Prentice-Hall.
- Zhou, S.-M. and Gan, J. Q. (2008). Low-level interpretability and high-level interpretability: A unified view of data-driven interpretable fuzzy system modelling. *Fuzzy Set. Syst.*, 159(23), pp. 3091–3131.
- Zitzler, E., Laumanns, M. and Bleuler, S. (2004). A tutorial on evolutionary multiobjective optimization. In *Metaheuristics for Multiobjective Optimisation*, pp. 3–37. Springer.
- Zitzler, E., Laumanns, M. and Thiele, L. (2001). SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In Giannakoglou, K. C., Tsahalis, D. T., Périoux, J., Papailiou, K. D. and Fogarty, T. (eds.), *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*. International Center for Numerical Methods in Engineering, pp. 95–100.
- Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. and da Fonseca, V. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE T. Evol. Comput.*, 7(2), pp. 117–132.

Chapter 19

Bio-Inspired Optimization of Interval Type-2 Fuzzy Controllers

Oscar Castillo

This chapter presents a general framework for designing interval type-2 fuzzy controllers (FCs) based on bio-inspired optimization techniques. The problem of designing optimal type-2 FCs for complex nonlinear plants under uncertain environments is of crucial importance in achieving good results for real-world applications. Traditional approaches have been the use of genetic algorithms (GAs) or trial and error approaches; however results tend to be not optimal or require very large design times. More recently, bio-inspired optimization techniques, like ant colony optimization (ACO) or particle swarm intelligence, have also been tried on optimal design of FCs. In this chapter, we show how the use of bio-inspired optimization techniques can be used to obtain results that outperform traditional approaches in the design of optimal type-2 FCs.

19.1. Introduction

We describe in this chapter, new methods for building intelligent systems using type-2 fuzzy logic and bio-inspired optimization techniques. Bio-inspired optimization includes techniques, such as particle swarm optimization (PSO), ant colony optimization (ACO) and genetic algorithms (GAs), have been applied in numerous optimization problems. In this paper, we are extending the use of fuzzy logic to a higher order, which is called type-2 fuzzy logic (Castillo and Melin, 2008; Mendel, 2000). Combining type-2 fuzzy logic with bio-inspired optimization techniques, we can build powerful hybrid intelligent systems that can use the advantages that each technique offers in solving complex control problems.

Fuzzy logic is an area of soft computing that enables a computer system to reason with uncertainty (Zadeh, 1975). A fuzzy inference system consists of a set of if-then rules defined over fuzzy sets (Castillo and Melin, 2008; Mendel, 2000). Fuzzy sets generalize the concept of a traditional set by allowing the membership degree to be any value between 0 and 1. This corresponds, in the real world, to many situations where it is difficult to decide in an unambiguous manner if something belongs or not to a specific class. Fuzzy expert systems, for example, have been applied with some success to problems of decision, control, diagnosis and classification, just because they can manage the complex expert reasoning involved in these areas of application. The main disadvantage of fuzzy systems is that they cannot adapt to changing situations. For this reason, it is a good idea to combine fuzzy logic with neural networks or GAs, because either one of these last two methodologies could give adaptability to the fuzzy system (Castro *et al.*, 2009; Wang *et al.*, 2004). On the other hand, the knowledge that is used to build these fuzzy rules is uncertain. Such uncertainty leads to rules whose antecedents or consequents are uncertain, which translates into uncertain antecedent or consequent membership functions (Hagras, 2004; Mendel, 2000). Type-1 fuzzy systems, like the ones mentioned above, whose membership functions are type-1 fuzzy sets, are unable to directly handle such uncertainties. Type-2 fuzzy sets are fuzzy sets whose membership grades themselves are type-1 fuzzy sets; they are very useful in circumstances where it is difficult to determine an exact membership function for a fuzzy set (Biglarbegian *et al.*, 2011; Castillo *et al.*, 2008a, 2008b; Melin and Castillo, 2004, 2007).

Uncertainty is an inherent part of intelligent systems used in real-world applications (Bingül and Karahan, 2011; Castillo *et al.*, 2007). The use of new methods for handling incomplete information is of fundamental importance (Cazarez-Castro *et al.*, 2008; Dereli *et al.*, 2011; Martinez-Marroquin *et al.*, 2009; Starczewski, 2009; Yager, 1980). Type-1 fuzzy sets used in conventional fuzzy systems cannot fully handle the uncertainties present in intelligent systems (Cervantes and Castillo, 2010; Chaoui and Gueaieb, 2008). Type-2 fuzzy sets that are used in type-2 fuzzy systems can handle such uncertainties in a better way because they provide us with more parameters (Castillo *et al.*, 2012; Galluzzo and

Cosenza, 2010; Hsiao *et al.*, 2008). This chapter deals with the design of intelligent systems using interval type-2 fuzzy logic for minimizing the effects of uncertainty produced by the instrumentation elements, environmental noise, etc. Experimental results include simulations of feedback control systems for nonlinear plants using type-1 and type-2 fuzzy logic controllers (FLCs); a comparative analysis of the systems' response is performed, with and without the presence of uncertainty (Martinez *et al.*, 2009, 2010). The main contribution of the chapter is the proposed approach for the design of type-2 FLCs using bio-inspired optimization algorithms (Astudillo *et al.*, 2010; Castillo *et al.*, 2012).

We describe the use of ACO for the problem of finding the optimal intelligent controller for an autonomous wheeled mobile robot, in particular for the problem of tuning a fuzzy controller (FC) of Sugeno type. In our case study the controller has four inputs, each of them with two membership functions and we consider the interpolation point for every pair of membership function as the main parameter and their individual shape as secondary ones in order to achieve the tuning of the FC by using an ACO algorithm (Martinez-Marroquin *et al.*, 2009). Simulation results show that using ACO and coding the problem with just three parameters instead of six, allows us to find an optimal set of membership function parameters for the fuzzy control system with less computational effort needed (Castillo *et al.*, 2012).

This chapter also describes the application of the optimization algorithm for particle swarm known by its acronym as PSO, used to adjust the parameters of membership functions of a FLC to find the optimal intelligent control for a wheeled autonomous mobile robot. Results of several simulations show that the PSO is able to optimize the type-1 and type-2 FLC for this specific application (Martinez *et al.*, 2010). We use the PSO method to find the parameters of the membership functions of a type-2 FLC in order to minimize the state error for nonlinear systems. PSO is used to find the optimal type-2 FLC to achieve regulation of the output and stability of the closed-loop system (Martinez *et al.*, 2009). For this purpose, we change the values of the cognitive, social and inertia variables in the PSO. Simulation results, with the optimal FLC implemented in Simulink, show the feasibility of the proposed approach.

In general, the above mentioned applications of type-2 fuzzy logic in intelligent control are representative of the state of art in the area. However, we also have to mention that there exist applications of type-2 fuzzy logic in pattern recognition (Huang *et al.*, 2009), time series prediction (Castro *et al.*, 2009), and classification (Dereli *et al.*, 2011), which have been successful in the real world, but are not the main concern in this chapter. There have also been important theoretical advances on type-2 fuzzy logic that have enable more efficient processing and type reduction (Mendel, 2000; Starczewski, 2009; Yager, 1980), which have helped obtaining solutions to real world problems (Kim *et al.*, 2010; Koca *et al.*, 2011; Poornaselvan *et al.*, 2008; Sudha and Vijaya Santhi, 2011; Wagner and Hagras, 2007a, 2007b; Wu and Tan, 2004, 2006).

19.2. Design of Interval Type-2 FCs

Uncertainty is an inherent part of intelligent systems used in real-world applications (Wagner and Hagras, 2007a; Wu and Tan, 2006). The use of new methods for handling incomplete information is of fundamental importance (Castillo and Melin, 2008; Mendel, 2000). Type-1 fuzzy sets used in conventional fuzzy systems cannot fully handle the uncertainties present in intelligent systems. Type-2 fuzzy sets that are used in type-2 fuzzy systems can handle such uncertainties in a better way because they provide us with more parameters (Biglarbegian *et al.*, 2011; Castro *et al.*, 2007). This section deals with the design of intelligent systems using interval type-2 fuzzy logic for minimizing the effects of uncertainty produced by the instrumentation elements, environmental noise, etc. Experimental results include simulations of feedback control systems for nonlinear plants using type-1 and type-2 FLCs; a comparative analysis of the systems' response is performed, with and without the presence of uncertainty (Castillo *et al.*, 2012).

19.2.1. Introduction to the Design

Uncertainty affects decision-making and appears in a number of different forms. The concept of information is fully connected with the concept of uncertainty. The most fundamental aspect of this connection is that the uncertainty involved in any problem solving situation is a result of some information deficiency, which may be incomplete, imprecise, fragmentary, not fully reliable, vague, contradictory, or deficient in some other way. Uncertainty is an attribute of information. The general framework of fuzzy reasoning allows handling much of this uncertainty; fuzzy systems employ type-1 fuzzy sets, which represent uncertainty by numbers in the range [0, 1]. When something is uncertain, like a measurement, it is difficult to determine its exact value, and of course type-1 fuzzy sets make more sense than using sets. However, it is not reasonable to use an accurate membership function for something uncertain, so in this case what we need is another type of fuzzy sets, those which are able to handle these uncertainties, the so called type-2 fuzzy sets. So, the amount of uncertainty in a system can be reduced by using type-2 fuzzy logic because it offers better capabilities to handle linguistic uncertainties by modeling vagueness and unreliability of information (Castillo and Melin, 2008; Mendel, 2000).

In this section, we deal with the application of interval type-2 fuzzy control to nonlinear dynamic systems (Juang and Hsu, 2009b; Juang *et al.*, 2009; Mohammadi *et al.*, 2010). It is a well-known fact, that in the control of real systems, the instrumentation elements (instrumentation amplifier, sensors, digital to analog, analog to digital converters, etc.) introduce some sort of unpredictable values in the information that has been collected (Castillo *et al.*, 2012). So, the controllers designed under idealized conditions tend to behave in an inappropriate manner. Since, uncertainty is inherent in the design of controllers for real world applications; we are presenting how to deal with this problem using type-2 FLC, to reduce the effects of imprecise information (Mendel, 2000).

We are supporting this statement with experimental results, qualitative observations, and quantitative measures of errors. For quantifying the errors, we utilized three widely used performance criteria, these are: Integral of square error (ISE), integral of the absolute value of the error (IAE), and integral of the time multiplied by the absolute value of the error (ITAE).

19.2.2. Fuzzy Logic Systems (FLSs)

In this section, a brief overview of type-1 and type-2 fuzzy systems is presented. This overview is considered as necessary to understand the basic concepts needed to understand the methods and algorithms presented later in the paper (Castillo and Melin, 2008; Mendel, 2000).

19.2.2.1. Type-1 FLSs

In the 40s and 50s, many researchers proved that dynamic systems could be mathematically modeled using differential equations. In these works we have the foundations of the Control Theory, which in addition with the Transform Theory (Laplace's Theory), provided an extremely powerful means of analyzing and designing control systems. These theories were developed until the 70s, when the area was called Systems Theory to indicate its definitiveness.

Soft computing techniques have become an important research topic, which can be applied in the design of intelligent controllers. These techniques have tried to avoid the abovementioned drawbacks, and they allow us to obtain efficient controllers, which utilize the human experience in a more natural form than the conventional mathematical approach. In the cases in which a mathematical representation of the controlled system is difficult to obtain, the process operator has the knowledge, the experience to express the relationships existing in the process behavior.

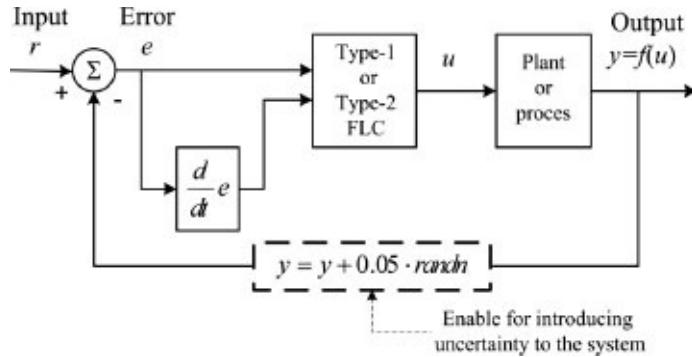
A FLS, described completely in terms of type-1 fuzzy sets is called a type-1 FLS. It is composed by a knowledge base, which comprises the information given by the process operator in form of linguistic control rules, a fuzzification interface, which has the effect of transforming crisp data into fuzzy sets, and inference system, which uses the fuzzy sets in conjunction with the knowledge base to make inferences by means of a reasoning method. Finally, a defuzzification interface, which translates the fuzzy control action so obtained to a real control action using a defuzzification method.

In this section, the implementation of the FC in terms of type-1 fuzzy sets, has two input variables, which are the error $e(t)$, the difference between the reference signal and the output of the process, as well as the error variation $\Delta e(t)$,

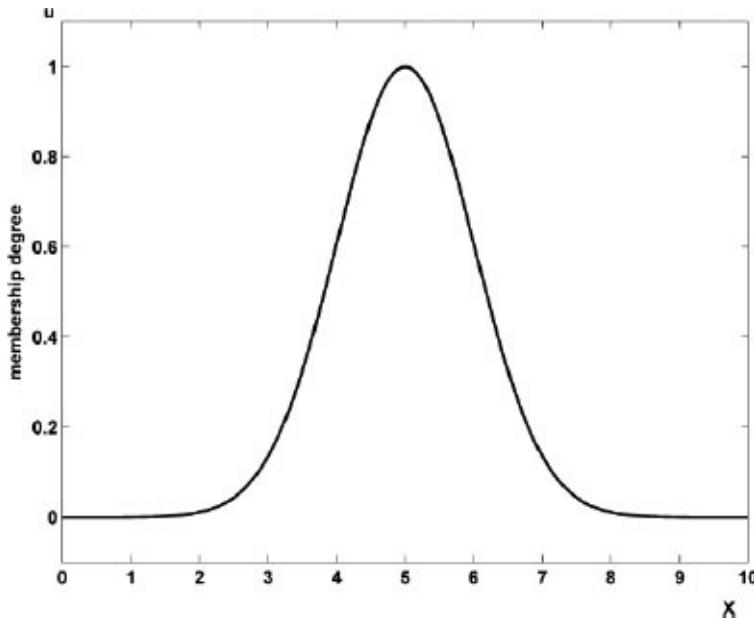
$$e(t) = r(t) - y(t), \quad (1)$$

$$\Delta e(t) = e(t) - e(t - 1), \quad (2)$$

so the control system can be represented as in [Figure 19.1](#).



[Figure 19.1](#): System used for obtaining the experimental results for control.



[Figure 19.2](#): Type-1 membership function.

19.2.2.2. Type-2 FLSs

If for a type-1 membership function, as in [Figure 19.2](#), we blur it to the left and to the right, as illustrated in [Figure 19.3](#), then a type-2 membership function is obtained. In this case, for a specific value x' , the membership function (u'), takes on different values, which are not all weighted the same, so we can assign an amplitude distribution to all of those points.

Doing this for all $x \in X$, we create a three-dimensional membership function—a type-2 membership function—that characterizes a type-2 fuzzy set. A type-2 fuzzy set \tilde{A} , is characterized by the membership function:

$$\tilde{A} = \{(x, u), \mu_{\tilde{A}}(x, u)\} \mid \forall x \in X, \forall u \in J_x \subseteq [0, 1]\}, \quad (3)$$

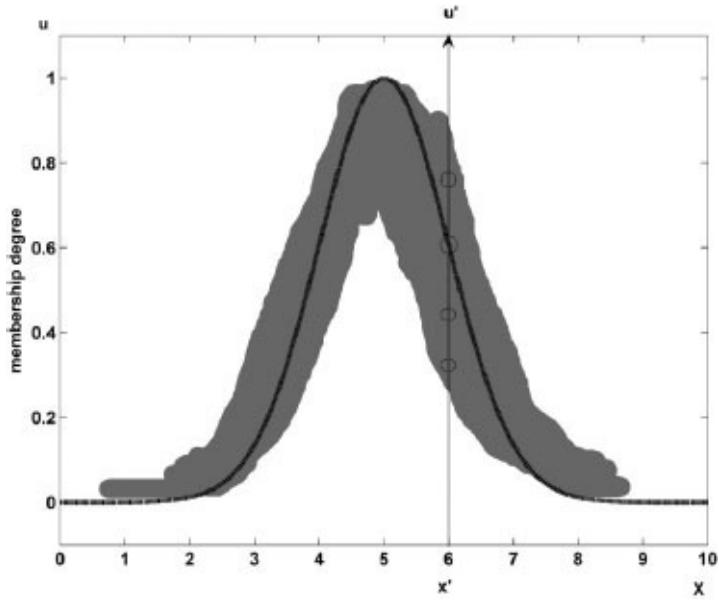


Figure 19.3: Blurred type-1 membership function.

in which $0 \leq \mu_{\tilde{A}}(x, u) \leq 1$. Another expression for \tilde{A} is

$$\tilde{A} = \iint_{x \in X, u \in J_x} \frac{\mu_{\tilde{A}}(x, u)}{(x, u) J_x} \subseteq [0, 1], \quad (4)$$

where \iint denotes the union over all admissible input variables x and u . For discrete universes of discourse \iint is replaced by \sum (Mendel, 2000). In fact $J_x \subseteq [0, 1]$ represents the primary membership of x , and $\mu_{\tilde{A}}(x, u)$ is a type-1 fuzzy set known as the secondary set. Hence, a type-2 membership grade can be any subset in $[0, 1]$, the primary membership, and corresponding to each primary membership, there is a secondary membership (which can also be in $[0, 1]$) that defines the possibilities for the primary membership. Uncertainty is represented by a region, which is called the footprint of uncertainty (FOU). If $\mu_{\tilde{A}}(x, u) = 1, \forall u \in J_x \subseteq [0, 1]$ then we have an interval type-2 membership function, as shown in [Figure 19.4](#). The uniform shading for the FOU represents the entire interval type-2 fuzzy set and it can be described in terms of an upper membership function $\bar{\mu}_{\tilde{A}}(x)$ and a lower membership function $\underline{\mu}_{\tilde{A}}(x)$.

A FLS described using at least one type-2 fuzzy set is called a type-2 FLS. Type-1 FLSs are unable to directly handle rule uncertainties, because they use type-1 fuzzy sets that are certain. On the other hand, type-2 FLSs, are very useful in circumstances where it is difficult to determine an exact membership function, and there are measurement uncertainties.

It is known that type-2 fuzzy sets enable modeling and minimizing the effects of uncertainties in rule-based FLS. Unfortunately, type-2 fuzzy sets are more difficult to use and understand than type-1 fuzzy sets; hence, their use is not widespread yet. As a justification for the use of type-2 fuzzy sets, are mentioned at least four sources of uncertainties not considered in type-1 FLSs:

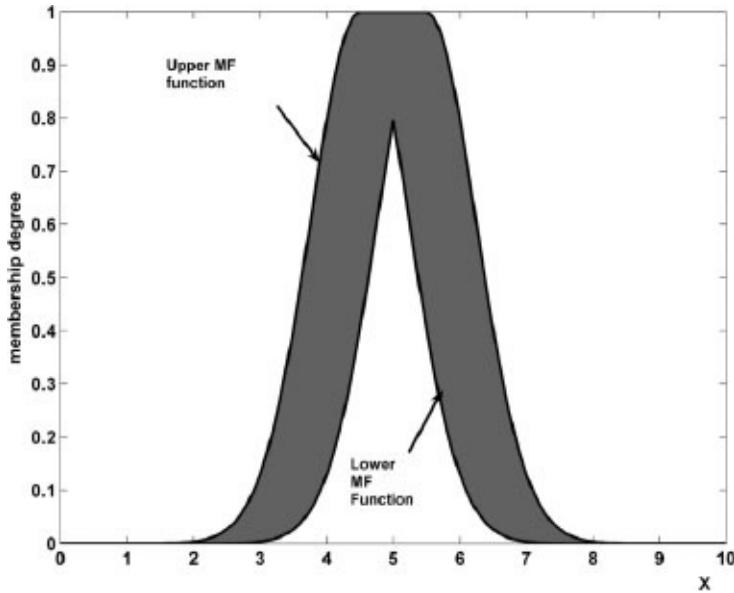


Figure 19.4: Interval type-2 membership function.

1. The meanings of the words that are used in the antecedents and consequents of rules can be uncertain (words mean different things to different people).
2. Consequents may have histogram of values associated with them, especially when knowledge is extracted from a group of experts who do not all agree.
3. Measurements that activate a type-1 FLS may be noisy and therefore uncertain.
4. The data used to tune the parameters of a type-1 FLS may also be noisy.

All of these uncertainties translate into uncertainties about fuzzy set membership functions. Type-1 fuzzy sets are not able to directly model such uncertainties because their membership functions are totally crisp. On the other hand, type-2 fuzzy sets are able to model such uncertainties because their membership functions are themselves fuzzy. A type-1 fuzzy set is a special case of a type-2 fuzzy set; its secondary membership function is a subset with only one element, unity.

A type-2 FLS is again characterized by IF-THEN rules, but its antecedent or consequent sets are now of type-2. Type-2 FLSs, can be used when the circumstances are too uncertain to determine exact membership grades such as when the training data is corrupted by noise. Similar to a type-1 FLS, a type-2 FLS includes a fuzzifier, a rule-base, fuzzy inference engine, and an output processor, as we can see in [Figure 19.5](#). The output processor includes type-reducer and defuzzifier; it generates a type-1 fuzzy set output (from the type-reducer) or a crisp number (from the defuzzifier). Now, we will explain each of the blocks of [Figure 19.5](#).

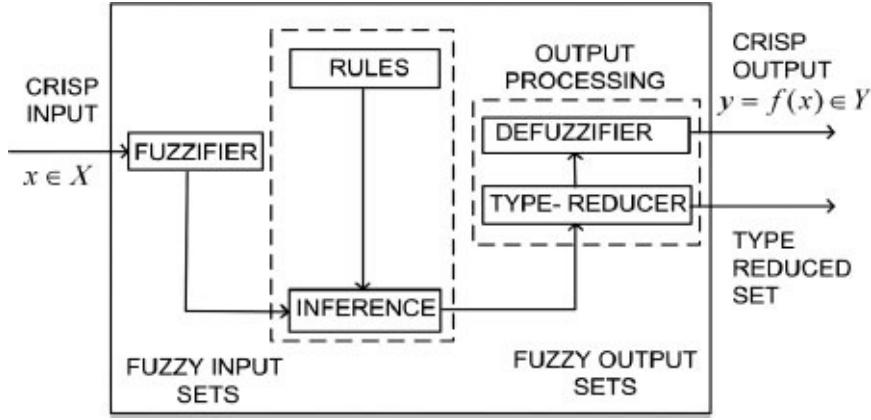


Figure 19.5: Type-2 FLS.

19.2.2.2. 1. Fuzzifier

The fuzzifier maps a crisp point $\mathbf{x} = (x_1, \dots, x_p)^T \in X_1 \times X_2 \times \dots \times X_p \equiv \mathbf{X}$ into a type-2 fuzzy set \tilde{A}_x in \mathbf{X} , interval type-2 fuzzy sets in this case. We will use type-2 singleton fuzzifier, in a singleton fuzzification, the input fuzzy set has only a single point on non-zero membership. \tilde{A}_x is a type-2 fuzzy singleton if $\mu_{\tilde{A}_x}(x) = 1/1$ for $\mathbf{x} = \mathbf{x}'$ and $\mu_{\tilde{A}_x}(x) = 1/0$ for all other $\mathbf{x} \neq \mathbf{x}'$.

19.2.2.2. 2. Rules

The structure of rules in a type-1 FLS and a type-2 FLS is the same, but in the latter the antecedents and the consequents will be represented by type-2 fuzzy sets. So for a type-2 FLS with p inputs $x_1 \in X_1, \dots, x_p \in X_p$ and one output $y \in Y$, multiple input single output (MISO), if we assume there are M rules, the l th rule in the type-2 FLS can be written as follows:

$$R^l : \text{IF } x_1 \text{ is } \tilde{F}_1^l \text{ and } x_p \text{ is } \tilde{F}_p^l, \quad \text{THEN } y \text{ is } \tilde{G}^l \quad l = 1, \dots, M. \quad (5)$$

19.2.2.2. 3. Inference

In the type-2 FLS, the inference engine combines rules and gives a mapping from input type-2 fuzzy sets to output type-2 fuzzy sets. It is necessary to compute the join \sqcup , (unions) and the meet \sqcap (intersections), as well as extended sup-star compositions (sup star compositions) of type-2 relations. If $\tilde{F}_1^l \times \dots \times \tilde{F}_p^l = \tilde{A}^l$, Equation (5) can be rewritten as

$$R^l : \tilde{F}_1^l \times \dots \times \tilde{F}_p^l \rightarrow \tilde{G}^l = \tilde{A}^l \rightarrow \tilde{G}^l \quad l = 1, \dots, M. \quad (6)$$

R^l is described by the membership function $\mu_{R^l}(\mathbf{x}, y) = \mu_{R^l}(x_1, \dots, x_p, y)$, where

$$\mu_{R^l}(\mathbf{x}, y) = \mu_{\tilde{A}^l \rightarrow \tilde{G}^l}(\mathbf{x}, y), \quad (7)$$

can be written as:

$$\begin{aligned}\mu_{R^l}(\mathbf{x}, y) &= \mu_{\tilde{A}^l \rightarrow \tilde{G}^l}(\mathbf{x}, y) = \mu_{\tilde{F}_1^l}(x_1) \prod \cdots \prod \mu_{\tilde{F}_p^l}(x_p) \prod \mu_{\tilde{G}^l}(y) \\ &= \left[\prod_{i=1}^p \mu_{\tilde{F}_i^l}(x_i) \right] \prod \mu_{\tilde{G}^l}(y).\end{aligned}\quad (8)$$

In general, the p -dimensional input to R^l is given by the type-2 fuzzy set \tilde{A}_x whose membership function is

$$\mu_{\tilde{A}_x}(\mathbf{x}) = \mu_{\tilde{x}_1}(x_1) \prod \cdots \prod \mu_{\tilde{x}_p}(x_p) = \prod_{i=1}^p \mu_{\tilde{x}_i}(x_i), \quad (9)$$

where $\tilde{X}_i (i = 1, \dots, p)$ are the labels of the fuzzy sets describing the inputs. Each rule R^l determines a type-2 fuzzy set $\tilde{B}^l = \tilde{A}_x \circ R^l$ such that:

$$\mu_{\tilde{B}^l}(y) = \mu_{\tilde{A}_x \circ R^l} = \sqcup_{x \in \mathbf{X}} \left[\mu_{\tilde{A}_x}(\mathbf{x}) \prod \mu_{R^l}(\mathbf{x}, y) \right] \quad y \in Y \quad l = 1, \dots, M. \quad (10)$$

This equation is the input/output relation in [Figure 19.5](#) between the type-2 fuzzy set that activates one rule in the inference engine and the type-2 fuzzy set at the output of that engine.

In the FLS we used interval type-2 fuzzy sets and meet under product t-norm, so the result of the input and antecedent operations, which are contained in the firing set $\prod_{i=1}^p \mu_{\tilde{F}_i^l}(x'_i \equiv F^l(\mathbf{x}'))$, is an interval type-1 set,

$$F^l(\mathbf{x}') = [\underline{f}^l(\mathbf{x}'), \bar{f}^l(\mathbf{x}')] \equiv [\underline{f}^l, \bar{f}^l], \quad (11)$$

where

$$\underline{f}^l(\mathbf{x}') = \underline{\mu}_{\tilde{F}_1^l}(x'_1) * \cdots * \underline{\mu}_{\tilde{F}_p^l}(x'_p) \quad (12)$$

and

$$\bar{f}^l(\mathbf{x}') = \bar{\mu}_{\tilde{F}_1^l}(x'_1) * \cdots * \bar{\mu}_{\tilde{F}_p^l}(x'_p), \quad (13)$$

where $*$ is the product operation.

19.2.2.4. Type reducer

The type-reducer generates a type-1 fuzzy set output, which is then converted in a crisp output through the defuzzifier. This type-1 fuzzy set is also an interval set, for the case of our FLS we used center of sets (COSSs) type reduction, Y_{cos} which is expressed as

$$Y_{\cos}(\mathbf{x}) = [y_l, y_r] = \int_{y^1 \in [y_l^1, y_r^1]} \dots \int_{y^M \in [y_l^M, y_r^M]} \times \int_{f^1 \in [f_l^1, f_r^1]} \dots \int_{f^M \in [f_l^M, f_r^M]} \frac{1}{\sum_{i=1}^M f^i y^i}. \quad (14)$$

This interval set is determined by its two end points, y_l and y_r , which corresponds to the centroid of the type-2 interval consequent set \tilde{G}^i ,

$$C_{\tilde{G}^i} = \int_{\theta_1 \in J_{y1}} \dots \int_{\theta_N \in J_{yN}} \frac{1}{\sum_{i=1}^N y_i \theta_i} = [y_l^i, y_r^i], \quad (15)$$

before the computation of $Y_{\cos}(\mathbf{x})$, we must evaluate Equation (15), and its two end points, y_l and y_r . If the values of f_i and y_i that are associated with y_l are denoted f_l^i and y_l^i , respectively, and the values of f_i and y_i that are associated with y_r are denoted f_r^i and y_r^i , respectively, from Equation (14), we have

$$y_l = \frac{\sum_{i=1}^M f_l^i y_l^i}{\sum_{i=1}^M f_l^i}, \quad (16)$$

$$y_r = \frac{\sum_{i=1}^M f_r^i y_r^i}{\sum_{i=1}^M f_r^i}. \quad (17)$$

19.2.2.2. 5. Defuzzifier

From the type-reducer we obtain an interval set Y_{\cos} , to defuzzify it we use the average of y_l and y_r , so the defuzzified output of an interval singleton type-2 FLS is

$$y(\mathbf{x}) = \frac{y_l + y_r}{2}. \quad (18)$$

In this paper, we are simulating the fact that the instrumentation elements (instrumentation amplifier, sensors, digital to analog, analog to digital converters, etc.) are introducing some sort of unpredictable values in the collected information. In the case of the implementation of the type-2 FLC, we have the same characteristics as in type-1 FLC, but we used type-2 fuzzy sets as membership functions for the inputs and for the output.

19.2.2.3. Performance criteria

For evaluating the transient closed-loop response of a computer control system we can use the same criteria that normally are used for adjusting constants in proportional integral derivative (PID) controllers. These are:

1. ISE.

$$\text{ISE} = \int_0^{\infty} [e(t)]^2 dt. \quad (19)$$

2. IAE.

$$\text{IAE} = \int_0^{\infty} |e(t)| dt. \quad (20)$$

3. ITAE.

$$\text{ITAE} = \int_0^{\infty} t|e(t)| dt. \quad (21)$$

The selection of the criteria depends on the type of response desired, the errors will contribute different for each criterion, so we have that large errors will increase the value of ISE more heavily than to IAE. ISE will favor responses with smaller overshoot for load changes, but ISE will give longer settling time. In ITAE, time appears as a factor, and therefore, ITAE will penalize heavily errors that occur late in time, but virtually ignore errors that occur early in time. Designing using ITAE will give us the shortest settling time, but it will produce the largest overshoot among the three criteria considered. Designing considering IAE will give us an intermediate result, in this case, the settling time will not be so large than using ISE or so small than using ITAE, and the same applies for the overshoot response. The selection of a particular criterion is depending on the type of desired response.

19.3. Optimization of FCs using ACO

In this section, we describe the application of a simple ACO (S-ACO) as a method of optimization for membership functions' parameters of a FLC in order to find the optimal intelligent controller for an Autonomous Wheeled Mobile Robot (Castillo *et al.*, 2012). Simulation results show that ACO outperforms a GA in the optimization of FLCs for an autonomous mobile robot (Castillo *et al.*, 2012).

19.3.1. Introduction

Nowadays, fuzzy logic is one of the most used methods of computational intelligence and with the best future; this is possible thanks to the efficiency and simplicity of Fuzzy Systems since they use linguistic terms similar to those that human beings use (Zadeh, 1975).

The complexity for developing fuzzy systems can be found at the time of deciding which are the best parameters of the membership functions, the number of rules or even the best granularity that could give us the best solution for the problem that we want to solve (Castillo and Melin, 2008).

A solution for the above mentioned problem is the application of evolutionary algorithms for the optimization of fuzzy systems (Castillo *et al.*, 2008a; Cervantes and Castillo, 2010; Cordon *et al.*, 2004; Martinez *et al.*, 2009; Wagner and Hagras, 2007a; Wu and Tan, 2006). Evolutionary algorithms can be a useful tool since its capabilities of solving nonlinear problems, well-constrained or even NP-hard problems. However, recently there have also been proposals of new optimization techniques based on biological or natural phenomena that have achieved good results in real world problems, for example, ACO, the bat algorithm, firefly algorithm, chemical optimization, and others (Astudillo *et al.*, 2010).

This section describes the application of one recent bio-inspired algorithms, such as the ACO (Juang and Hsu, 2009b) as a method of optimization of the parameters of the membership functions of the FLC in order to find the best intelligent controller for an Autonomous Wheeled Mobile Robot (Castillo *et al.*, 2012; Martinez-Marroquin *et al.*, 2009).

19.3.2. S-ACO Algorithm

ACO is a probabilistic technique that can be used for solving problems that can be reduced to finding good path along graphs. This method is inspired on the behavior presented by ants in finding paths from the nest or colony to the food source (Juang and Hsu, 2009b).

The S-ACO is an algorithmic implementation that adapts the behavior of real ants to solutions of minimum cost path problems on graphs (Poornaselvan *et al.*, 2008). A number of artificial ants build solutions for a certain optimization problem and exchange

information about the quality of these solutions making allusion to the communication systems of the real ants.

Let us define the graph $G = (V, E)$, where V is the set of nodes and E is the matrix of the links between nodes. G has $n_G = |V|$ nodes. Let us define L^k as the number of hops in the path built by the ant k from the origin node to the destiny node. Therefore, it is necessary to find:

$$Q = \{q_a, \dots, q_f \mid q_i \in C\}, \quad (22)$$

where Q is the set of nodes representing a continuous path with no obstacles; q_a, \dots, q_f are former nodes of the path and C is the set of possible configurations of the free space. If $x^k(t)$ denotes a Q solution in time t , $f(x^k(t))$ expresses the quality of the solution. The general steps of S-ACO are as follows:

- Each link (i, j) is associated with a pheromone concentration denoted as τ_{ij} .
- A number $k = 1, 2, \dots, n_k$ are placed in the nest.
- On each iteration all ants build a path to the food source (destiny node). For selecting the next node a probabilistic equation is used:
- Remove cycles and compute each route weight $f(x^k(t))$. A cycle could be generated when there are no feasible candidates nodes, that is, for any i and any k , $N_i^k = \emptyset$; then the predecessor of that node is included as a former node of the path.
- Pheromone evaporation is calculated with equation:

$$\tau_{ij}(t) \leftarrow (1 - \rho)\tau_{ij}(t), \quad (23)$$

where $\rho \in [0, 1]$ is the evaporation rate value of the pheromone trail. The evaporation is added to the algorithm in order to force the exploration of the ants, and avoid premature convergence to sub-optimal solutions. For $\rho = 1$ the search becomes completely random.

- The update of the pheromone concentration is realized using equation:

$$\tau_{ij}(t + 1) = \tau_{ij}(t) + \sum_{k=1}^{n_k} \Delta \tau_{ij}^k(t), \quad (24)$$

where $\Delta \tau_{ij}^k$ is the amount of pheromone that an ant k deposits in a link ij in a time t .

- Finally, the algorithm can be ended in three different ways:
 - When a maximum number of epochs has been reached.
 - When it has been found an acceptable solution, with $f(x_k(t)) < \varepsilon$.
 - When all ants follow the same path.

19.3.3. Problem Statement

The model of the robot considered in this paper is a unicycle mobile robot, that consists of two driving wheels mounted of the same axis and a front free wheel (Castillo *et al.*, 2012; Castro *et al.*, 2007b; Martinez *et al.*, 2009; Martinez-Marroquin *et al.*, 2009).

A unicycle mobile robot is an autonomous, wheeled vehicle capable of performing missions in fixed or uncertain environments. The robot body is symmetrical around the perpendicular axis and the center of mass is at the geometrical center of the body. It has two driving wheels that are fixed to the axis that passes through C and one passive wheel prevents the robot from tipping over as it moves on a plane. In what follows, it is assumed that motion of the passive wheel can be ignored in the dynamics of the mobile robot presented by the following set of equations:

$$M(q)\dot{\vartheta} + C(q, \dot{q})\dot{\vartheta} + D\vartheta = \tau + F_{ext}(t), \quad (25)$$

where $q = (x, y, \theta)^T$ is the vector of the configuration coordinates; $\vartheta = (v, w)^T$ is the vector of linear and angular velocities; $\tau = (\tau_1, \tau_2)$ is the vector of torques applied to the wheels of the robot where τ_1 and τ_2 denote the torques of the right and left wheel respectively (Figure 19.6); $F_{ext} \in \mathbb{R}^2$ uniformly bounded disturbance vector; $M(q) \in \mathbb{R}^{2 \times 2}$ is the positive-definite inertia matrix; $C(q, \dot{q})\dot{\vartheta}$ is the vector of centripetal and Coriolis forces; and $D \in \mathbb{R}^{2 \times 2}$ is a diagonal positive-definite damping matrix.

Furthermore, the Equation (25) has the following non-holonomic constraint:

$$\dot{y} \cos \theta - \dot{x} \sin \theta = 0, \quad (26)$$

which corresponds to a no-slip wheel condition preventing the robot from moving sideways (Chaoui and Gueaieb, 2008).

The control objective is to design a FLC of τ that ensures:

$$\lim_{t \rightarrow \infty} \|q_d(t) - q(t)\| = 0, \quad (27)$$

for any continuously, differentiable, bounded desired trajectory $q_d \in \mathbb{R}^3$ while attenuating external disturbances.

19.3.4. Fuzzy Logic Control Design

In order to satisfy the control objective it is necessary to design a FLC for the real velocities of the mobile robot. To do that, a Takagi–Sugeno FLC was designed, using linguistic variables in the input and mathematical functions in the output. The error of the linear and angular velocities (v_d, w_d respectively), were taken as inputs variables, while the right (τ_1) and left (τ_2) torques as outputs. The membership functions used on the input are trapezoidal for the negative (N) and positive (P), and a triangular was used for the zero (C)

linguistic terms. The interval used for this FC is $[-50, 50]$ (Castro *et al.*, 2007b).

The rule set of the FLC contains nine rules, which governs the input–output relationship of the FLC and this adopts the Takagi–Sugeno style inference engine, and it is used with a single point in the outputs, this mind that the outputs are constant values, obtained using weighted average defuzzification procedure. In [Table 19.1](#), we present the rule set whose format is established as follows:

Rule i : if e_v is G_1 and e_w is G_2 then F is G_3 and N is G_4

where G_1, \dots, G_4 are the fuzzy set associated to each variable $I = 1, 2, \dots, 9$.

Table 19.1: Fuzzy rules set.

e_v/e_w	N	Z	P
N	N/N	N/Z	N/P
Z	Z/N	Z/Z	Z/P
P	P/N	P/Z	P/P

Table 19.2: Parameters of the membership functions.

MF Type	Point	Minimal value	Maximal value
Trapezoidal	a	-50	-50
	b	-50	-50
	c	-15	-5.1
	d	-1.5	-0.5
Triangular	a	-5	-1.8
	b	0	0
	c	1.8	5
Trapezoidal	a	0.5	1.5
	b	5.1	15
	c	50	50
	d	50	50
Constant (N)	a	-50	-50
Constant (C)	a	0	0
Constant (P)	a	50	50

To find the best FLC, we used a S-ACO to find the parameters of the membership functions. [Table 19.2](#) shows the parameters of the membership functions, the minimal and maximum values in the search range for the S-ACO algorithm to find the best FLC.

It is important to remark that values shown in [Table 19.2](#) are applied to both inputs and both outputs of the FLC.

19.3.5. ACO Architecture

The S-ACO algorithm was applied for the optimization of the membership functions for the FLC. For developing the architecture of the algorithm it was necessary to follow the next steps:

1. Marking the limits of the problem in order to eliminate unnecessary complexity.

2. Representing the architecture of the FLC as a graph that artificial ants could traverse.
3. Achieving an adequate handling of the pheromone but permitting the algorithm to evolve by itself.

Table 19.3: Parameters of membership functions included in S-ACO search.

MF type	Point	Minimal value	Maximal value
Trapezoidal	c	-15	-5.1
	d	-1.5	-0.5
Triangular	a	-5	-1.8
	c	1.8	5
Trapezoidal	a	0.5	1.5
	b	5.1	15

Table 19.4: Number of possible values of the parameters of membership functions.

MF Type	Point	Combinations
Trapezoidal	c	100
	d	15
Triangular	a	33
	c	33
Trapezoidal	a	15
	b	100

19.3.5.1. Limiting the problem and graph representation

One of problems found on the development of the S-ACO algorithm was to make a good representation of FLC. First we reduced the number of elements that the method needed to find by deleting the elements whose minimal value and maximal values are the same (see [Table 19.2](#)) and therefore if they were included they will not change any way. [Table 19.3](#) shows the parameters of the membership functions included in the search.

The next step was to represent those parameters shown in [Table 19.3](#); to that, was necessary to discretize the parameters in a range of possible values in order to represent every possible value as a node in the graph of search. The level of discretization between minimal and maximal value was of 0.1 (by example: -1.5, -1.4, -1.3, ..., -0.5).

[Table 19.4](#) shows the number of possible values that each parameter can take.

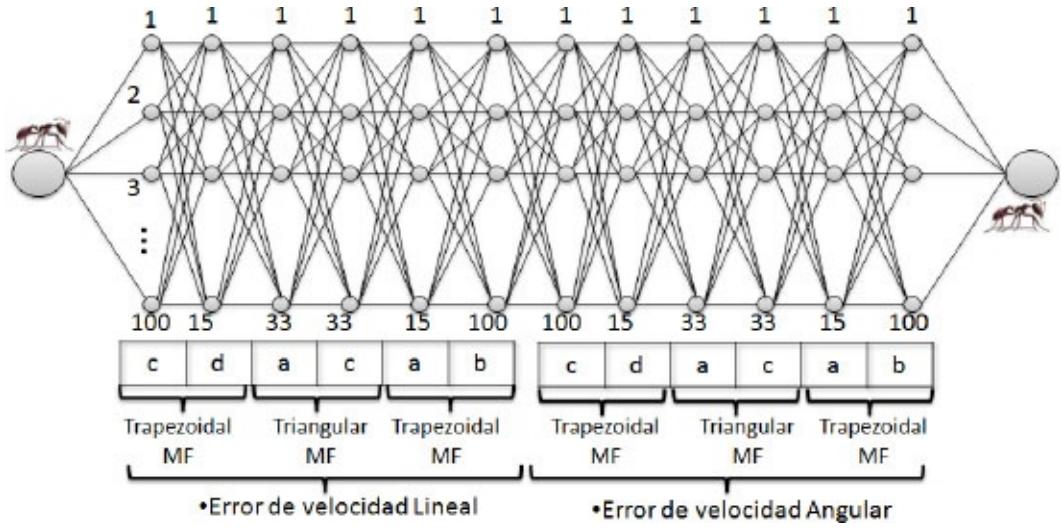


Figure 19.6: S-ACO search graph.

Figure 19.6 shows the search graph for the proposed S-ACO algorithm, the graph can be viewed as a tree where the root is the nest and the last node is the food source.

19.3.6. Simulation Results

Table 19.5 shows the results of the FLC, obtained varying the values of maximum iterations and number of artificial ants, where the highlighted row shows the best result obtained with the method. Figure 19.10 shows the evolving of the method.

A trajectory tracking controller has been designed based on the dynamics and kinematics of the autonomous mobile robot through the application of ACO for the optimization of membership functions for the FLC with good results obtained after simulations.

19.4. Optimization of FCs using PSO

This section describes the application of the optimization algorithm for particle swarm known by its acronym as PSO (Engelbrecht, 2005), used to adjust the parameters of membership functions of a FLC to find the optimal intelligent control for a wheeled autonomous mobile robot (Martinez *et al.*, 2010). Results of several simulations show that the PSO is able to optimize the type-1 and type-2 FLC for this specific application (Castillo *et al.*, 2012).

This section describes the application of particle swarm algorithm (PSO) as a method of optimizing the parameters of membership functions of the proposed FLC in order to find the best intelligent controller for an autonomous mobile robot.

Table 19.5: S-ACO results of simulations for FLC optimization.

Iterations	Ants	α	ρ	Average error	Time
20	10	0.2	Random	1.5589	00:01:30
20	10	0.2	Random	1.451	00:01:34
25	10	0.2	Random	1.5566	00:01:46
25	10	0.2	Random	1.4767	00:01:51
25	10	0.2	Random	1.4739	00:02:05
25	10	0.2	Random	1.6137	00:02:08
25	10	0.2	Random	1.6642	00:01:54
25	100	0.2	Random	1.3484	00:20:30
25	100	0.2	Random	1.3413	00:18:44
25	100	0.2	Random	1.3360	00:18:31
25	100	0.2	Random	1.2954	00:18:32
25	100	0.2	Random	1.4877	00:18:41
25	100	0.2	Random	1.2391	00:18:31
10	15	0.2	Random	1.6916	00:01:14
10	15	0.2	Random	1.4256	00:01:09
40	65	0.2	Random	1.2783	00:19:17
40	65	0.2	Random	1.4011	00:19:45
40	65	0.2	Random	1.2216	00:19:33
40	65	0.2	Random	1.2487	00:19:49
50	70	0.2	Random	1.3782	00:26:09
50	70	0.2	Random	1.0875	00:27:35
50	70	0.2	Random	1.4218	00:33:45
50	70	0.2	Random	1.475	01:08:48
25	80	0.2	Random	1.4718	00:14:55
25	80	0.2	Random	1.4212	00:15:00
25	80	0.2	Random	1.3221	00:14:52
25	80	0.2	Random	1.1391	00:15:41
50	80	0.2	Random	1.2148	00:28:43
62	50	0.2	Random	1.0322	00:24:49
50	80	0.2	Random	1.1887	00:29:55
50	80	0.2	Random	1.2158	00:29:56
60	90	0.2	Random	1.3493	00:41:56
60	90	0.2	Random	1.3060	00:39:48
60	90	0.2	Random	1.3161	00:40:00

19.4.1. PSO Algorithm

Optimizer by clouds or swarm of particles (PSO) is a relatively new technique that is slowly taking rise and recognition as an effective and efficient algorithm (Engelbrecht,

2005). While PSO algorithm shares similarities with other evolutionary computation techniques while also differs in certain respects and needs no evolution operators such as crossover and mutation (Cordon *et al.*, 2004).

PSO emulates the swarm behavior of insects, a herd of grazing animals, a cloud of birds, a host of fish in these swarms or clouds that made the search for food in a collaborative manner. Each member of a swarm adapts its search patterns, learning from his own experience and experiences of other members, i.e., taking into account their cognitive beliefs and social beliefs.

These phenomena are discussed in the algorithm and the mathematical models are built on the method for updating the positions of each particle.

In the PSO algorithm, a member in the swarm, called a particle, represents a possible solution is a point in the search space. The global optimum is considered as the location of food, the problem would be implemented as the optimal solution found. Each particle has a fitness value and a speed to adjust the flying direction according to the best.

The general formula [Equation (28)] for determining the motion of particles which are presented below, is shown in two parts the cognitive and social part of which are crucial to identify the type of algorithm that is being implemented in our case we used the Full GBEST i.e., both C_1 and C_2 must have values greater than 0 but less than 4, respectively.

$$V_{id} = V_{id}(t + 1) + C_1 r_{id}(t)[Y_{id}(t) - X_{id}(t)] + C_2 r_2(t)[Y_{id}(t) - X_{id}(t)]. \quad (28)$$

There is another formula [Equation (33)] that is critical for the update of each particle, this assesses the current position of the particle and the previous position to choose which is the most appropriate to find the result more quickly this position is recalculated at each new iterations that is the algorithm.

$$X_i(t + 1) = X_i(t) + V_i(t + 1). \quad (29)$$

19.4.2. Design of the FC

As determined by one of the most used and effective techniques to solve control problems is to use fuzzy systems to meet the control objective, so it is necessary to design a FC for the actual speed of the mobile robot. In this research work, a type inference systems using Takagi–Sugeno type fuzzy system in order both 1 and 2, the use of linguistic variables in the input and output functions is applied. The error of linear and angular velocities (respectively) was taken as input variables, while the left and right pairs are taken as outputs. The membership functions used in the entry are trapezoidal for negative (N) and positive (P), and a triangle was used to zero (C) linguistic terms. [Figure 19.7](#) shows the input and output variables used, these are used for both types of fuzzy logic (1 and 2).

The FLC has nine rules, which are adapted to the style of Takagi–Sugeno controller,

so the output has a single point, so the results are constants values P, C, N), which are obtained through a procedure using a weighted average defuzzification by:

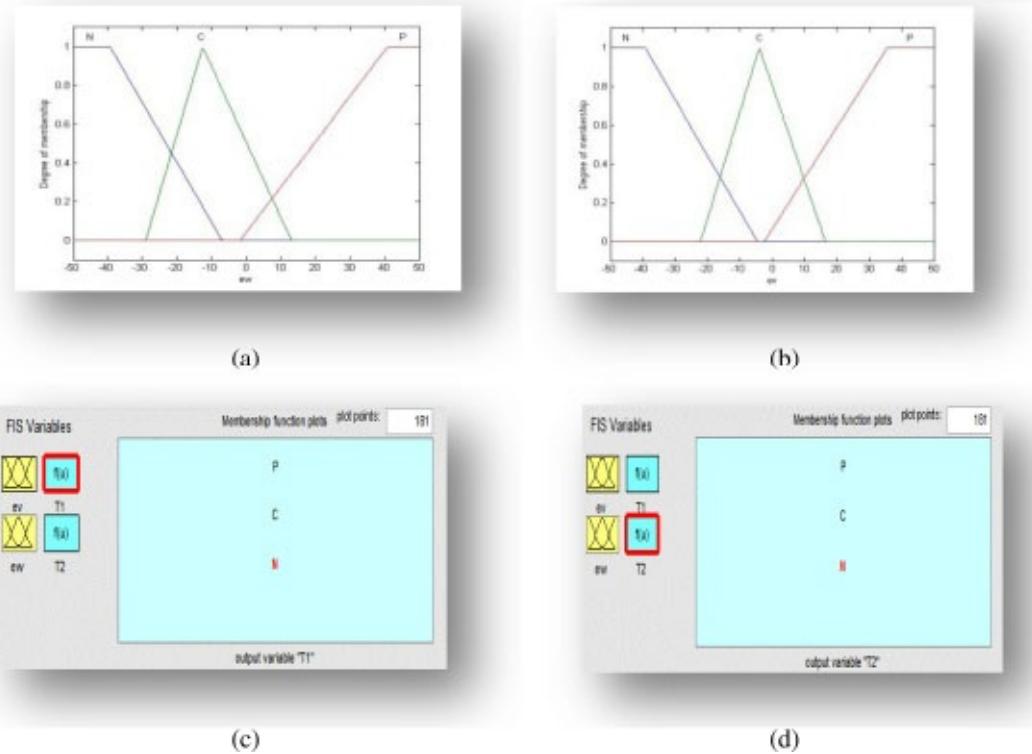


Figure 19.7: Variable input/output: (a) error of the linear velocity (ev). (b) Angular velocity error (ew). (c) Exit fee (τ 1). (d) Exit left (τ 2).

- ☞ R1: If Vangular is C and VLinear is C, then τ 1 is C and τ 2 is C
- ☞ R2: If Vangular is C and VLinear is P, then τ 1 is C and τ 2 is P
- ☞ R3: If Vangular is C and VLinear is N, then τ 1 is C and τ 2 is N
- ☞ R4: If Vangular is P and VLinear is C, then τ 1 is P and τ 2 is C
- ☞ R5: If Vangular is P and VLinear is P, then τ 1 is P and τ 2 is P
- ☞ R6: If VLinear is P and Vangular is N, then τ 1 is P and τ 2 is N
- ☞ R7: If VLinear is N and Vangular is C, then τ 1 is N and τ 2 is C
- ☞ R8: If VLinear is N and Vangular is P, then τ 1 is N and τ 2 is P
- ☞ R9: If Vangular VLinear is N and is N, then τ 1 is N and τ 2 is N.

The linguistic terms of input variables are shown in the first row and column of [Table 19.6](#), the rest of the content corresponds to the linguistic terms of output variables.

Table 19.6: FLC rules.

Ev/ew	N	C	P
N	N/N	N/C	N/P
C	C/N	C/C	C/P
P	P/N	P/C	P/P

Table 19.7: Results for PSO using the constriction coefficient for type-1.

Experiment No.	Iterations	Swarm	Coefficient constriction	Average error	Runtime
1	50	50	1.0	0.0606	00:20:05
2	100	200	1.0	0.2670	02:44:53
3	100	100	1.0	0.0301	01:49:55
4	100	150	1.0	0.0315	01:54:31
5	100	50	1.0	0.0266	03:16:34
6	100	57	1.0	0.0211	00:41:38
7	100	300	1.0	0.0276	04:04:54
8	100	80	1.0	0.0527	01:03:45
9	150	80	1.0	0.0260	01:35:17
10	300	150	1.0	0.0307	04:30:16
11	500	200	1.0	0.0529	39:56:23
12	200	90	1.0	0.0345	01:54:59
13	150	100	1.0	0.0496	01:36:37
14	100	53	1.0	0.0230	00:39:29

19.4.3. Results of the Simulations

This section presents the results of proposed controller to stabilize the autonomous mobile robot. The evaluation is done through computer simulation done in MATLAB® and Simulink® (Wagner and Hagras, 2007b).

[Table 19.7](#) shows the results of the FLC, obtained by varying the values of maximum iterations and the number of particles, where the highlighted row shows the best result obtained with the method.

As you can see, the above results are acceptable for type-1 FLC obtaining a final result of 0.0211 using 57 particles and 100 iterations in a time of 47 minutes and 38 seconds for this experiment, but as previously mentioned were also simulations with type-2, taking into account the same parameters and conditions of Takagi–Sugeno controller, the results of these simulations are presented below in [Table 19.8](#).

The above results are acceptable for type-2 fuzzy logic control because we obtain a final result of 0.0500 using 380 particles and 300 iterations in a time of seven hours 55 minutes and 08 seconds for this experiment, which cannot be considered high due to the complexity of the problem.

Table 19.8: Results of PSO using the constriction coefficient for type-2.

Experiment no.	Iterations	Swarm	Coefficient constriction	Average error	Runtime
1	100	150	1.0	0.0659	02:20:31
2	100	200	1.0	0.0675	02:56:21
3	100	250	1.0	0.0666	03:05:31
4	200	150	1.0	0.0663	03:55:06
5	200	200	1.0	0.0651	04:07:14
6	200	250	1.0	0.0642	04:20:10
7	200	300	1.0	0.0536	04:54:43
8	200	350	1.0	0.0554	05:12:09
9	250	350	1.0	0.0600	05:38:20
10	300	300	1.0	0.0531	06:12:57
11	300	350	1.0	0.0503	07:30:28
12	300	380	1.0	0.0500	07:55:08
13	350	400	1.0	0.0501	08:10:15
14	350	450	1.0	0.0503	08:59:01
15	400	300	1.0	0.0502	12:31:11

Table 19.9: Comparison of results of the PSO algorithm for type-1 and type-2.

	Iterations	Swarm	Average error	Runtime
PSO with type-1 fuzzy logic	100	57	0.0211	00:41:38
PSO with type-2 fuzzy logic	300	380	0.0050	07:55:08

The type-2 FCs were implemented with the type-2 fuzzy logic toolbox that was developed previously by our research group (Castro *et al.*, 2007a, 2007b).

With the results of the experiments shown in [Table 19.9](#), we can determine that for this particular problem the optimized type-2 FLC clearly outperform its type-1 counterpart.

The trajectory tracking controller is designed based on the dynamics and kinematics of mobile autonomous robot through the application of PSO for the optimization of membership functions of FC, both type-1 and type-2 with the good results obtained after simulations.

19.5. Conclusions

This chapter has presented a general framework for designing interval type-2 FCs based on bio-inspired optimization techniques. A trajectory tracking controller has been designed based on the dynamics and kinematics of the autonomous mobile robot through the application of ACO for the optimization of membership functions for the FLC with good results obtained after simulations. Also, a trajectory tracking controller is designed based on the dynamics and kinematics of mobile autonomous robot through the application of PSO for the optimization of membership functions of the FC for both type-1 and type-2 with the good results obtained after simulations. As future work we plan to test other bio-inspired optimization techniques, like the bat or firefly algorithms for this particular robotic application and in other different kind of applications.

References

- Astudillo, L., Melin, P. and Castillo, O. (2010). A new optimization method based on a paradigm inspired by nature. *Stud. Comput. Intell.*, 312, pp. 277–283.
- Biglarbegian, M., Melek, W. W. and Mendel, J. M. (2011). Design of novel interval type-2 fuzzy controllers for modular and reconfigurable robots: Theory and experiments. *IEEE Trans. Ind. Electron.*, 58, pp. 1371–1384.
- Bingül, Z. and Karahan, O. (2011). A fuzzy logic controller tuned with PSO for 2 DOF robot trajectory control. *Expert Syst. Appl.*, 38, pp. 1017–1031.
- Castillo, O., Martinez, A. I. and Martinez, A. C. (2007). Evolutionary computing for topology optimization of type-2 fuzzy systems. *Adv. Soft Comput.*, 41, pp. 63–75.
- Castillo, O. and Melin, P. (2008). *Type-2 Fuzzy Logic: Theory and Applications*. Heidelberg, Germany: Springer-Verlag.
- Castillo, O., Huesca, G. and Valdez, F. (2008a). Evolutionary computing for topology optimization of type-2 fuzzy controllers. *Stud. Fuzziness Soft Comput.*, 208, pp. 163–178.
- Castillo, O., Aguilar, L. T., Cazarez-Castro, N. R. and Cardenas, S. (2008b). Systematic design of a stable type-2 fuzzy logic controller. *Appl. Soft Comput. J.*, 8, pp. 1274–1279.
- Castillo, O., Martinez-Marroquin, R., Melin, P., Valdez, F. and Soria, J. (2012). Comparative study of bio-inspired algorithms applied to the optimization of type-1 and type-2 fuzzy controllers for an autonomous mobile robot. *Inf. Sci.*, 192, pp. 19–38.
- Castro, J. R., Castillo, O. and Melin, P. (2007a). An interval type-2 fuzzy logic toolbox for control applications. *Proc. FUZZ-IEEE 2007*. London, UK, pp. 1–6.
- Castro, J. R., Castillo, O. and Martinez, L. G. (2007b). Interval type-2 fuzzy logic toolbox. *Eng. Lett.*, 15(1), p. 14.
- Castro, J. R., Castillo, O., Melin, P., Martinez, L. G., Escobar, S. and Camacho, I. (2007c). Building fuzzy inference systems with the interval type-2 fuzzy logic toolbox. *Adv. Soft Comput.*, 41, pp. 53–62.
- Castro, J. R., Castillo, O., Melin, P. and Rodriguez-Diaz, A. (2009). A hybrid learning algorithm for a class of interval type-2 fuzzy neural networks. *Inf. Sci.*, 179, pp. 2175–2193.
- Cazarez-Castro, N. R., Aguilar, L. T. and Castillo, O. (2008). Hybrid genetic-fuzzy optimization of a type-2 fuzzy logic controller. *Proc. Eighth Int. Conf. Hybrid Intell. Syst., HIS 2008*. Barcelona, Spain, pp. 216–221.
- Cervantes, L. and Castillo, O. (2010). Design of a fuzzy system for the longitudinal control of an F-14 airplane. *Stud. Comput. Intell.*, 318, pp. 213–224.
- Chaoui, H. and Gueaieb, W. (2008). Type-2 fuzzy logic control of a flexible-joint manipulator. *J. Intell. Robot. Syst.: Theory Appl.*, 51, pp. 159–186.
- Cordon, O., Gomide, F., Herrera, F., Hoffmann, F. and Magdalena, L. (2004). Ten years of genetic fuzzy systems: Current framework and new trends. *Fuzzy Sets Syst.*, 141, pp. 5–31.
- Dereli, T., Baykasoglu, A., Altun, K., Durmusoglu, A. and Turksen, I. B. (2011). Industrial applications of type-2 fuzzy sets and systems: A concise review. *Comput. Ind.*, 62, pp. 125–137.
- Engelbrecht, A. P. (2005). *Fundamentals of Computational Swarm Intelligence*. England: Wiley.
- Galluzzo, M. and Cosenza, B. (2010). Adaptive type-2 fuzzy logic control of a bioreactor. *Chem. Eng. Sci.*, 65, pp. 4208–4221.
- Hagras, H. (2004). Hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots. *IEEE Trans. Fuzzy Syst.*, 12, pp. 524–539.
- Hsiao, M., Li, T. H. S., Lee, J. Z., Chao, C. H. and Tsai, S. H. (2008). Design of interval type-2 fuzzy sliding-mode controller. *Inf. Sci.*, 178, pp. 1686–1716.
- Huang, H. C., Chu, C. M. and Pan, J. S. (2009). The optimized copyright protection system with genetic watermarking. *Soft Comput.*, 13, pp. 333–343.
- Juang, C.-F. and Hsu, C.-H. (2009a). Reinforcement ant optimized fuzzy controller for mobile-robot wall-following control. *IEEE Trans. Ind. Electron.*, 56, pp. 3931–3940.
- Juang, C.-F. and Hsu, C.-H. (2009b). Reinforcement interval type-2 fuzzy controller design by online rule generation and

- Q-value-aided ant colony optimization. *IEEE Trans. Syst., Man, Cybern., Part B Cybern.*, 39, pp. 1528–1542.
- Juang, C.-F., Hsu, C.-H. and Chuang, C.-F. (2009). Reinforcement self-organizing interval type-2 fuzzy system with ant colony optimization. *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, San Antonio, pp. 771–776.
- Kim, W.-D., Jang, H.-J. and Oh, S.-K. (2010). The design of optimized fuzzy cascade controller: focused on type-2fuzzy controller and HFC-based genetic algorithms. *Trans. Korean Inst. Electr. Eng.*, 59, pp. 972–980.
- Koca, G. O., Akpolat, Z. H. and Özdemir, M. (2011). Type-2 fuzzy sliding mode control of a four-bar mechanism. *Int. J. Model. Simul.*, 31, pp. 60–68.
- Martinez, R., Castillo, O. and Aguilar, L. T. (2009). Optimization of interval type-2 fuzzy logic controllers for a perturbed autonomous wheeled mobile robot using genetic algorithms. *Inf. Sci.*, 179, pp. 2158–2174.
- Martinez, R., Rodriguez, A., Castillo, O. and Aguilar, L. T. (2010). Type-2 fuzzy logic controllers optimization using genetic algorithms and particle swarm optimization. *Proc. IEEE Int. Conf. Granular Comput., GrC 2010*, San Jose, California, USA, pp. 724–727.
- Martinez-Marroquin, R., Castillo, O. and Soria, J. (2009). Parameter tuning of membership functions of a type-1 and type-2 fuzzy logic controller for an autonomous wheeled mobile robot using ant colony optimization. *Proc. IEEE Int. Conf. Syst., Man, Cybern.* San Antonio, pp. 4770–4775.
- Melin, P. and Castillo, O. (2004). A new method for adaptive control of nonlinear plants using type-2 fuzzy logic and neural networks. *Int. J. Gen. Syst.*, 33, pp. 289–304.
- Melin, P. and Castillo, O. (2007). An intelligent hybrid approach for industrial quality control combining neural networks, fuzzy logic and fractal theory. *Inf. Sci.*, 177, pp. 1543–1557.
- Mendel, J. M. (2000). Uncertainty, fuzzy logic, and signal processing. *Signal Process. J.*, 80, pp. 913–933.
- Mohammadi, S. M. A., Gharaveisi, A. A. and Mashinchi, M. (2010). An evolutionary tuning technique for type-2fuzzy logic controller in a nonlinear system under uncertainty. *Proc. 18th Iran. Conf. Electr. Eng., ICEE 2010*, Tehran, Iran, pp. 610–616.
- Poornaselvan, K. J., Gireesh Kumar, T. and Vijayan, V. P. (2008). Agent based ground flight control using type-2 fuzzy logic and hybrid ant colony optimization to a dynamic environment. *Proc. First Int. Conf. Emerg. Trends Eng. Technol., ICETET 2008*, Nagpur, Maharashtra, India, pp. 343–348.
- Starczewski, J. T. (2009). Efficient triangular type-2 fuzzy logic systems. *Int. J. Approx. Reason.*, 50, pp. 799–811.
- Sudha, K. R. and Vijaya Santhi, R. (2011). Robust decentralized load frequency control of interconnected power system with generation rate constraint using type-2 fuzzy approach. *Int. J. Electr. Power Energy Syst.*, 33, pp. 699–707.
- Wagner, C. and Hagras, H. (2007a). A genetic algorithm-based architecture for evolving type-2 fuzzy logic controllers for real world autonomous mobile robots. *Proc. IEEE Conf. Fuzzy Syst.*, London, UK.
- Wagner, C. and Hagras, H. (2007b). Evolving type-2 fuzzy logic controllers for autonomous mobile robots. *Adv. Soft Comput.*, 41, pp. 16–25.
- Wang, C.-H., Cheng, C.-S. and Lee, T.-T. (2004). Dynamical optimal training for interval type-2 fuzzy neural network (T2FNN). *IEEE Trans. Syst., Man, Cybern., Part B Cybern.*, 34(3), pp. 1462–1477.
- Wu, D. and Tan, W.-W. (2004). A type-2 fuzzy logic controller for the liquid level process. *Proc. IEEE Conf. Fuzzy Systems*. Budapest, Hungary, pp. 953–958.
- Wu, D. and Tan, W.-W. (2006). Genetic learning and performance evaluation of interval type-2 fuzzy logic controllers. *Eng. Appl. Artif. Intell.*, 19, pp. 829–841.
- Yager, R. R. (1980). Fuzzy subsets of type II in decisions. *J. Cybern.*, 10, pp. 137–159.
- Zadeh, L. A. (1975). The concept of a linguistic variable and its application to approximate reasoning. *Inf. Sci.*, 8, pp. 43–80.

Chapter 20

Nature-Inspired Optimization of Fuzzy Controllers and Fuzzy Models

Radu-Emil Precup and Radu-Codruț David

This chapter deals with the nature-inspired optimal tuning of fuzzy controllers (FCs) and of fuzzy models. In the first part, the charged system search (CSS) algorithm, a novel nature-inspired optimization algorithm, is employed for the optimal tuning of Takagi–Sugeno proportional-integral (PI) FCs for nonlinear servo systems, minimizing the objective functions defined as the sum of time multiplied by squared error and the sum of output sensitivity function of the state sensitivity model with respect to process parametric variations; optimal fuzzy control systems with reduced parametric sensitivity are derived. The second part presents the simulated annealing (SA) algorithm, a representative nature-inspired optimization algorithm, which carries out the optimal tuning of Takagi–Sugeno fuzzy model parameters for anti-lock braking systems (ABS), derived on the basis of the modal equivalence principle by placing a set of linearized process models at several operating points in the rule consequents; the objective functions expressed as the root mean square of modeling errors are minimized. Both parts focus on performance improvements of fuzzy control systems and fuzzy models.

20.1. Introduction

The majority of fuzzy control applications uses fuzzy controllers (FCs) for direct feedback control or placed at the low level in hierarchical control system structures (Precup and Hellendoorn, 2011). However, fuzzy control can be used at the supervisory level, for example, in adaptive control system structures. Nowadays fuzzy control is no longer only used to directly express the knowledge on the process or, in other words, to carry out model-free fuzzy control. A fuzzy controller can be tuned from a fuzzy model obtained in terms of system identification techniques, and thus it can be viewed in the framework of model-based fuzzy control or in the framework of data-driven control (Hjalmarsson *et al.*, 1998; Radac *et al.*, 2011; Owens *et al.*, 2013; Hou and Wang, 2013; Formentin *et al.*, 2013a, 2013b).

The following categories of FCs are most often used:

- Mamdani FCs, referred to also as linguistic FCs, with either fuzzy consequents that represent type-I fuzzy systems (Sugeno, 1999) or singleton consequents which belong to type-II fuzzy systems. These FCs are usually used as direct closed-loop controllers.
- Takagi–Sugeno FCs, known also as type-III fuzzy systems especially when affine consequents are employed, and typically used as supervisory controllers.

A systematic way to meet the performance specifications of fuzzy control systems is the optimization in terms of optimization problems with variables represented by the parameters of FCs or fuzzy models. Nature-inspired algorithms can solve the optimization problems, and they ensure the optimal tuning of FCs and fuzzy models in order to meet the performance specifications expressed by adequately defined objective functions and constraints. The motivation for nature-inspired optimization algorithms in combination with fuzzy control and fuzzy modeling concerns their ability to cope with non-convex or non-differentiable objective functions due to controllers' structures and nonlinearities, and to process complexity, which can lead to multi-objective optimization problems.

The latest nature-inspired optimization algorithms applied to the parameter tuning of FCs include genetic algorithms (Onieva *et al.*, 2012; Das *et al.*, 2013; Pérez *et al.*, 2013), simulated annealing (SA) (Haber *et al.*, 2009; Jain *et al.*, 2011; Precup *et al.*, 2012a), particle swarm optimization (PSO) (Bingül and Karahan, 2011; Oh *et al.*, 2011; Precup *et al.*, 2013a), Gravitational Search Algorithms (GSAs) (Precup *et al.*, 2013a, 2013b; David *et al.*, 2013), Ant Colony Optimization (Kumaresan, 2011; Chang *et al.*, 2012; Lu and Liu, 2013), cross-entropy (Haber *et al.*, 2010), chemical optimization (Melin *et al.*, 2013), charged system search (CSS) algorithms (Precup *et al.*, 2014), harmony search (Wang *et al.*, 2013), etc., in several fuzzy control system structures. The appropriate tools specific to the analysis of fuzzy systems must be accounted for in all applications (Preitl and Precup, 1997; Batyrshin *et al.*, 2002; Škrjanc *et al.*, 2002; Baranyi *et al.*, 2003, 2013; Škrjanc and Blažič, 2005; Johanyák, 2010; Topalov *et al.*, 2011; Tikk *et al.*, 2011; Angelov and Yager,

2012, 2013; Obradović *et al.*, 2013; Feriyonika and Dewantoro, 2013).

Some popular applications on the nature-inspired optimal tuning of fuzzy models concern the optimization of the parameters in the rule antecedents and consequents of Takagi–Sugeno fuzzy models by SA (Almaraashi *et al.*, 2010), the SA-based tuning of the parameters of input membership functions (m.f.s) in fuzzy control systems for general nonlinear systems (Liu and Yang, 2000), chaotic time series (Yanara and Akyürek, 2011), ABS (David *et al.*, 2014), servo systems (Precup *et al.*, 2013c), and magnetic levitation systems (David *et al.*, 2012; Dragos *et al.*, 2013). A hybrid PSO-based algorithm is suggested for Takagi–Sugeno fuzzy model identification (Wang *et al.*, 2011). The input m.f.s of Takagi–Sugeno fuzzy models are also tuned by a combination of Island Model Parallel Genetic Algorithm and space search memetic algorithm (Ho and Garibaldi, 2013). The migration algorithms (Vaščák, 2012; Vaščák and Pal'a, 2012), the PSO-based optimization of linguistic fuzzy models (Chen *et al.*, 2012; Jiang *et al.*, 2012), the Variable string length Artificial Bee Colony algorithm (Su *et al.*, 2012), the genetic algorithms (Pedrycz and Song, 2012) and the chaotic GSA (Li *et al.*, 2013) give good results in fuzzy model optimization.

This chapter presents two original solutions for the optimal tuning of fuzzy control systems and fuzzy models. This methodology of parameter tuning is important as it represents a proficient approach to the design of optimal control systems with a reduced parametric sensitivity. The included digital and experimental results validate new important contributions, as the nature-inspired optimization algorithms provide simplified process models in the design and tuning of fuzzy control systems with reduced sensitivity in regards to the process parameters involved in the sensitivity models, and the relatively small number of tuning parameters used in the objective functions, which facilitate the design of low-cost optimal fuzzy control systems.

The next section gives aspects concerning the design and tuning of Mamdani and Takagi–Sugeno FCs with dynamics with focus on PI-FCs and the Takagi–Sugeno fuzzy models. [Section 20.3](#) discusses the CSS algorithm-based optimal tuning of FCs for nonlinear servo systems. [Section 20.4](#) presents the SA algorithm-based optimal tuning of the parameters of input m.f.s of Takagi–Sugeno fuzzy models for anti-lock braking systems (ABSs). The concluding remarks are highlighted in [Section 20.5](#).

20.2. FCs and Fuzzy Models

20.2.1. Structures and Design Methodology of FCs

The FCs without dynamics offer nonlinear input–output static maps, which can be modified by the proper tuning of all parameters in the FC structure (the shapes and scaling factors of m.f.s, the rule-base, and the defuzzification method). The introduction of dynamics of integral and/or derivative type in the FC structure can be carried out on either the inputs or the outputs of the FC resulting in two versions of PI-FCs (Preitl and Precup, 1997):

- The PI-FC with integration of controller output (PI-FC-OI),
- The PI-FC with integration of controller input (PI-FC-II)

and both versions of PI-FCs, with the structures given in [Figure 20.1](#), are type-2 fuzzy systems according to Sugeno and Kóczy (Sugeno, 1999; Kóczy, 1996).

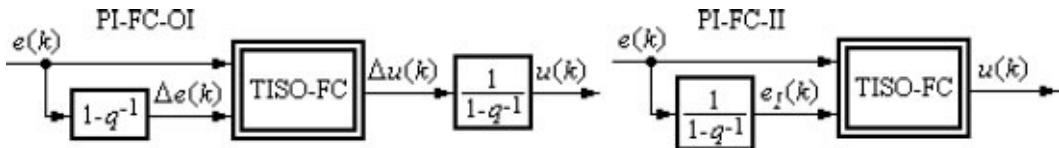


Figure 20.1: Structures of PI-FCs.

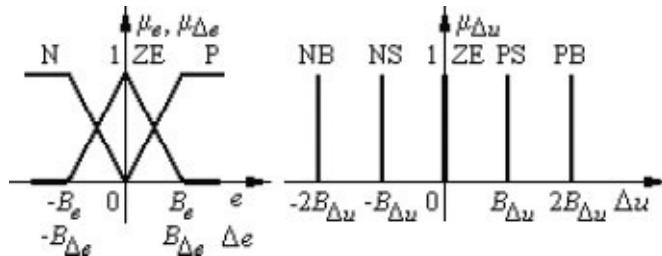


Figure 20.2: Input and output membership functions of Mamdani PI-FCs with integration on controller output.

As shown in [Figure 20.1](#), the two PI-FCs are built around the two inputs-single output fuzzy controller (TISO-FC) without dynamics. The presentation is focused on PI-FCs as they can be easily extended to PI-derivative FCs (PID-FCs) and transformed in particular forms of proportional-derivative FCs (PD-FCs). In addition, the structures presented in [Figure 20.1](#) can be processed such that to be extended to two-degree-of-freedom (2-DOF) FCs.

The dynamics of the PI-FC-OI is obtained by the numerical differentiation of the control error $e(k)$, which results in the increment of control error $\Delta e(k) = e(k) - e(k - 1)$, and by the numerical integration of the increment of control signal $\Delta u(k) = u(k) - u(k - 1)$ which leads to the control signal $u(k)$. The variable k in [Figure 20.1](#) is the current discrete time index, and q indicates the forward time shift operator. The dynamics of the PI-FC-II is obtained by the numerical integration of $e(k)$, which gives the integral of control error $e_I(k) = e_I(k - 1) + e(k)$. It is assumed here that the eventually nonlinear scaling factors of the input and output variables of TISO-FC are inserted in the controlled process.

The fuzzification in the TISO-FC block that belongs to PI-FC-OI is performed by means of the input m.f.s illustrated in [Figure 20.2](#) for the Mamdani PI-FCs. The fuzzification in the TISO-FC that belongs to the PI-FC-II is performed by the same m.f.s as those of the PI-FC-OI, but the input variable $e_I(k)$ is used instead of $\Delta e(k)$, and the output variable $u(k)$ is used instead of $\Delta u(k)$. The relatively simple shapes of m.f.s presented in [Figure 20.2](#) and their reduced numbers ensure the cost-effective implementation of the FCs. Other distributions of the membership functions can modify in a desired way the FC nonlinearities. The Takagi–Sugeno PI-FCs employ only the input m.f.s taken from [Figure 20.2](#).

Both versions of Mamdani PI-FCs, namely the PI-FC-OI and the PI-FC-II, employ Mamdani’s MAX-MIN compositional rule of inference assisted by the rule-base exemplified for the PI-FC-OI

- Rule 1: IF $e(k)$ IS N AND $\Delta e(k)$ IS P THEN $\Delta u(k)$ IS ZE,
 - Rule 2: IF $e(k)$ IS ZE AND $\Delta e(k)$ IS P THEN $\Delta u(k)$ IS PS,
 - Rule 3: IF $e(k)$ IS P AND $\Delta e(k)$ IS P THEN $\Delta u(k)$ IS PB,
 - Rule 4: IF $e(k)$ IS N AND $\Delta e(k)$ IS ZE THEN $\Delta u(k)$ IS NS,
 - Rule 5: IF $e(k)$ IS ZE AND $\Delta e(k)$ IS ZE THEN $\Delta u(k)$ IS ZE,
 - Rule 6: IF $e(k)$ IS P AND $\Delta e(k)$ IS ZE THEN $\Delta u(k)$ IS PS,
 - Rule 7: IF $e(k)$ IS N AND $\Delta e(k)$ IS N THEN $\Delta u(k)$ IS NB,
 - Rule 8: IF $e(k)$ IS ZE AND $\Delta e(k)$ IS N THEN $\Delta u(k)$ IS NS,
 - Rule 9: IF $e(k)$ IS P AND $\Delta e(k)$ IS N THEN $\Delta u(k)$ IS ZE
- (1)

and the center of gravity method for singletons is used for defuzzification. The inference engines of both versions of Takagi–Sugeno PI-FCs are based on the SUM and PROD operators assisted by the rule-base exemplified for the PI-FC-OI.

- Rule 1: IF $e(k)$ IS N AND $\Delta e(k)$ IS P THEN $\Delta u(k) = K_p^1[\Delta e(k) + \mu^1 e(k)]$,
 - Rule 2: IF $e(k)$ IS ZE AND $\Delta e(k)$ IS P THEN $\Delta u(k) = K_p^2[\Delta e(k) + \mu^2 e(k)]$,
 - Rule 3: IF $e(k)$ IS P AND $\Delta e(k)$ IS P THEN $\Delta u(k) = K_p^3[\Delta e(k) + \mu^3 e(k)]$,
 - Rule 4: IF $e(k)$ IS N AND $\Delta e(k)$ IS ZE THEN $\Delta u(k) = K_p^4[\Delta e(k) + \mu^4 e(k)]$,
 - Rule 5: IF $e(k)$ IS ZE AND $\Delta e(k)$ IS ZE THEN $\Delta u(k) = K_p^5[\Delta e(k) + \mu^5 e(k)]$,
 - Rule 6: IF $e(k)$ IS P AND $\Delta e(k)$ IS ZE THEN $\Delta u(k) = K_p^6[\Delta e(k) + \mu^6 e(k)]$,
 - Rule 7: IF $e(k)$ IS N AND $\Delta e(k)$ IS N THEN $\Delta u(k) = K_p^7[\Delta e(k) + \mu^7 e(k)]$,
 - Rule 8: IF $e(k)$ IS ZE AND $\Delta e(k)$ IS N THEN $\Delta u(k) = K_p^8[\Delta e(k) + \mu^8 e(k)]$,
 - Rule 9: IF $e(k)$ IS P AND $\Delta e(k)$ IS N THEN $\Delta u(k) = K_p^9[\Delta e(k) + \mu^9 e(k)]$
- (2)

and the weighted average method is employed for defuzzification.

The PI-FC structures and the rule-bases given in Equations (1) and (2) make these controllers behave like bumpless interpolators between separately designed PI controllers. The maximum number of such controllers is nine, and the following conditions ensure the interpolation between only two separately designed PI controllers in case of Takagi–

Sugeno PI-FCs:

$$\begin{aligned} K_P^1 &= K_P^2 = K_P^4 = K_P^5 = K_P^6 = K_P^8 = K_P^9 = K_P, \\ K_P^3 &= K_P^7 = \eta K_P, \\ \mu^1 &= \mu^2 = \mu^3 = \mu^4 = \mu^5 = \mu^6 = \mu^7 = \mu^8 = \mu^9 = \mu. \end{aligned} \quad (3)$$

The unified design methodology consists of the following design steps for Mamdani and Takagi–Sugeno PI-FCs in their PI-FC-OI versions:

Step 1. The continuous-time design of the linear PI controller with the transfer function $C(s)$

$$C(s) = \frac{k_c(1 + sT_i)}{s} = k_C \left[1 + \frac{1}{(sT_i)} \right], \quad k_C = k_c T_i, \quad (4)$$

is carried out, leading to the controller gain k_c (or k_C) and to the integral time constant T_i .

Step 2. The sampling period T_s is set according to the requirements of quasi-continuous digital control. Tustin’s method is then applied to discretize the continuous-time linear PI controller, and the recurrent equation of the incremental digital PI controller is

$$\Delta u(k) = K_P [\Delta e(k) + \mu e(k)], \quad (5)$$

where the expressions of the parameters K_P and μ in Equations (3) and (5) are

$$K_P = k_c \left(\frac{T_i - T_s}{2} \right), \quad \mu = \frac{2T_s}{(2T_i - T_s)}. \quad (6)$$

The parameter η , with typical values $0 < \eta < 1$, is introduced in Equation (3) to alleviate the overshoot of the fuzzy control system when both inputs have the same sign. Therefore, these PI-FCs can also be applied to the control of non-minimum phase systems with right half-plane zeros when such rule-bases produce the alleviation of the downshoot.

Step 3. The modal equivalence principle (Galichet and Foulloy, 1995) is applied to map the linear controller parameters onto the PI-FC ones. Therefore, the tuning condition for the Takagi–Sugeno PI-FC-OI is

$$B_{\Delta e} = \mu B_e \quad (7)$$

and the tuning conditions for the Mamdani PI-FC-OI is

$$B_{\Delta e} = \mu B_e, \quad B_{\Delta e} = K_P \mu B_e. \quad (8)$$

The tuning conditions for the Mamdani PI-FC-II (Preitl and Precup, 1997) are

$$B_{\Delta e} = \left(\frac{1}{\mu} \right) B_e, \quad B_{\Delta e} = K_P B_e \quad (9)$$

and the tuning condition for the Takagi–Sugeno PI-FC-II results as follows in a similar manner:

$$B_{\Delta e} = \left(\frac{1}{\mu} \right) B_e. \quad (10)$$

The value of the parameter B_e in Equations (7)–(10) must be set by the designer. This value can be set according to designer’s experience, and the stability analysis of the fuzzy control system can be conducted with this regard. However, the systematic tuning of the parameter B_e will be ensured as follows as solutions to the optimization problems as this parameter will be one of the elements of the vector variable of the objective functions.

The PD-FCs are designed using the methodology for PI-FCs. The design is carried out in terms of the PD-FC structure obtained from the PI-FC-OI structure given in [Figure 20.1](#) by dropping out the output integrator.

The PID-FC structure is presented in [Figure 20.3](#), where the TISO-FC^{PD} block and the TISO-FC^{PI} block are identical to the TISO-FC block given in [Figure 20.1](#), u_k^{PD} is the control signal of the PD controller, and Δu_k^{PI} is the increment of control signal calculated by the PI controller. The fuzzification is carried out by means of the input m.f.s exemplified in [Figure 20.3](#) for the Takagi–Sugeno PID-FC, where $g \in \{PI, PD\}$. [Figure 20.3](#) indicates that the PID-FC structure consists of a parallel connection of PD-FC and PI-FC.

The design of the PID-FC is carried out using the same methodology. However, the equivalence to the linear PID controller has to be ensured in terms of the manipulation (Precup *et al.*, 2008) of the input–output characterization of the linear PID controller which is transformed into the nonlinear parallel structure given in [Figure 20.3](#). The transfer function of the linear PID controller is $C(s)$:

$$C(s) = k_C \left[1 + \frac{1}{(sT_i)} + sT_d \right] = \frac{k_c(1 + sT_{c1})(1 + sT_{c2})}{s},$$

$$k_c = \frac{k_C}{T_i}, \quad T_{c1,2} = \frac{T_i \left(1 \pm \sqrt{1 - \frac{4T_d}{T_i}} \right)}{2}, \quad (11)$$

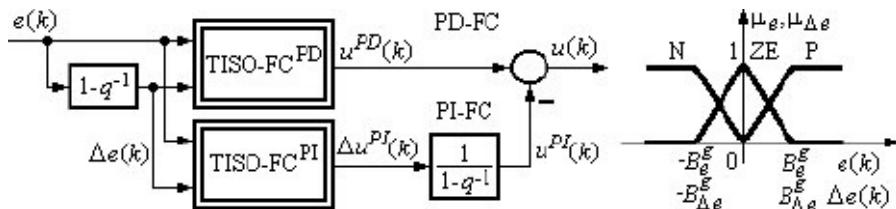


Figure 20.3: Structure and input membership functions of Takagi–Sugeno PID-FC.

where T_d is the derivative time constant and T_{c1} and T_{c2} are the controller time constants.

Setting the sampling period T_s , Tustin's method leads to the discrete-time PID controller with the transfer function $C(q^{-1})$

$$C(q^{-1}) = \frac{\rho_0 + \rho_1 q^{-1} + \rho_2 q^{-2}}{1 - q^{-1}}, \quad (12)$$

$$\rho_0 = k_c \left(\frac{1 + T_d}{T_s} \right), \quad \rho_1 = -k_c \left(\frac{1 + 2T_d}{T_s} - \frac{T_s}{T_i} \right), \quad \rho_2 = k_c \frac{T_d}{T_s}.$$

The expression (12) is organized as the following parallel connection of a discrete-time PD controller and a discrete-time PI controller with the superscripts PD and PI, respectively, associated to [Figure 20.3](#):

$$C(q^{-1}) = K_P^{PD}(1 - q^{-1} + \mu^{PD}) - \frac{K_P^{PI}(1 - q^{-1} + \mu^{PI})}{(1 - q^{-1})}, \quad (13)$$

$$K_P^{PD} = \rho_2, \quad \mu^{PD} = -\frac{\rho_1}{\rho_2}, \quad K_P^{PI} = \rho_2 - \rho_1 - \rho_0,$$

$$\mu^{PI} = \frac{2\rho_2}{(\rho_2 - \rho_1 - \rho_0)}.$$

Therefore, it is justified to carry out the separate design of the PI-FC and of the PD-FC shown in [Figure 20.3](#) using the same methodology.

The PI-FC can be involved in 2-DOF FCs by the fuzzification of some of the controller blocks in 2-DOF linear controllers in order to improve the control system performance. The 2-DOF PI-FCs exhibit performance improvements over the one-DOF FCs (Precup *et al.*, 2009; Preitl *et al.*, 2010a, 2010b), and similar controller structures can be formulated as state-feedback control systems.

The extensions of FCs to type-2 FCs prove additional performance improvement. The tuning of type-2 FCs is mainly based on optimization algorithms (Linda and Manic, 2011a, 2011b; Kayacan *et al.*, 2011; Castillo and Melin, 2012a, 2012b).

20.2.2. Takagi–Sugeno Fuzzy Models

The general rule-base of the continuous-time Takagi–Sugeno fuzzy model (of the process) consists of the rules (Takagi and Sugeno, 1985);

$$\begin{aligned} \text{Rule } i: & \text{ IF } z_1(t) \text{ IS } T_{1,k_1}^i \text{ AND } z_2(t) \text{ IS } T_{2,k_2}^i \text{ AND } \dots \text{ AND } z_s(t) \text{ IS } T_{s,k_s}^i \\ & \text{ THEN } y_m(t) = \varphi_i(\mathbf{x}(t)), \quad i = 1, \dots, n_R, \end{aligned} \quad (14)$$

where t is the continuous time variable, n_R is the number of rules, $y_m(t)$ is the continuous-time Takagi–Sugeno fuzzy model output, the notations T_{1,k_1}^i , T_{2,k_2}^i , and T_{s,k_s}^i are used for the linguistic terms of the input variables $z_1(t)$, $z_2(t)$, and $z_s(t)$, respectively, referred to also as scheduling variables, with the number and ranges of indices $k_1 = 1 \dots f_1$, $k_2 = 1 \dots f_2$, and

$k_s = 1 \dots f_s$, $\mathbf{x}(t) = [x_1(t) \ x_2(t) \dots x_n(t)]^T$ is the input vector to the fuzzy model, namely an additional input vector which appears in the rule consequent and it can represent the state vector of the process, the superscript T indicates the matrix transposition, and the functions φ_i can be general arbitrary smooth functions, with particular forms as linear functions, affine functions or crisp models with dynamics including linear or nonlinear state-space models placed in the rule consequents. The discrete-time Takagi–Sugeno fuzzy model is similar to Equation (14), but the discrete time variable k is used instead of the continuous time variable t .

20.3. CSS Algorithm-Optimal Tuning of FCs for Nonlinear Servo Systems

The process is characterized by the following nonlinear continuous-time time-invariant single input-single output (SISO) state-space model which defines a general class of nonlinear servo systems:

$$m(t) = \begin{cases} -1, & \text{if } u(t) \leq -u_b, \\ \frac{[u(t) + u_c]}{(u_b - u_c)}, & \text{if } -u_b < u(t) < -u_c, \\ 0, & \text{if } -u_c \leq |u(t)| \leq u_a, \\ \frac{[u(t) - u_a]}{(u_b - u_a)}, & \text{if } u_a < u(t) < u_b, \\ 1, & \text{if } u(t) \geq u_b, \end{cases} \quad (15)$$

$$\dot{\mathbf{x}}_P(t) = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{T_\Sigma} \end{bmatrix} \mathbf{x}_P(t) + \begin{bmatrix} 0 \\ \frac{k_p}{T_\Sigma} \end{bmatrix} m(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} d(t),$$

$$y(t) = [1 \ 0] \ \mathbf{x}_P(t),$$

where k_p is the process gain, T_Σ is the small time constant, the control signal u is a pulse width modulated duty cycle, and m is the output of the saturation and dead zone static nonlinearity specific to the actuator. The nonlinearity is modeled by the first equation in Equation (15), with the parameters u_a , u_b , and u_c fulfilling the conditions $0 < u_a < u_b$, $0 < u_c < u_b$. The state-space model (15) includes the actuator and measuring element dynamics. The state vector $\mathbf{x}_P(t)$ is expressed as follows in (angular) position applications:

$$\mathbf{x}_P(t) = [x_{P,1}(t) \ x_{P,2}(t)]^T = [\alpha(t) \ \omega(t)]^T, \quad (16)$$

where $\alpha(t)$ is the angular position, $\omega(t)$ is the angular speed, and the controlled output is $y(t) = x_{P,1}(t) = \alpha(t)$.

The nonlinearity in (15) is neglected in the following simplified model of the process expressed as the transfer function $P(s)$:

$$P(s) = \frac{k_{EP}}{[s(1 + T_\Sigma s)]}. \quad (17)$$

This transfer function is considered for u as input and y as output. The equivalent process gain is k_{EP} :

$$k_{EP} = \begin{cases} \frac{k_p}{(u_b - u_c)}, & \text{if } -u_b < u(t) < -u_c, \\ \frac{k_p}{(u_b - u_a)}, & \text{if } u_a < u(t) < u_b. \end{cases} \quad (18)$$

Therefore, $P(s)$ can be used in the controller design and tuning in two cases out of the five cases concerning the nonlinearity in Equation (15).

The process models, Equations (15) and (17) can be employed in the control designs of servo systems in various applications accepting that the parameters k_P and T_Σ depend on the operating point. That is the reason why the design of control systems with a reduced parametric sensitivity with respect to k_P and T_Σ is justified. In this context, the minimization of the discrete-time objective function $J(\boldsymbol{\rho})$ ensures the sensitivity reduction with respect to the modifications of k_P or T_Σ :

$$J(\boldsymbol{\rho}) = \sum_{k=0}^{\infty} \{k|e(k)| + \gamma^2[\sigma(k)]^2\}, \quad (19)$$

where γ is the weighting parameter, $\boldsymbol{\rho}$ is the vector variable of the objective function and also the parameter vector of the FC, σ is the output sensitivity function defined in the state sensitivity model with respect to k_P or T_Σ , and the variables in the sum depend on $\boldsymbol{\rho}$.

The objective function $J(\boldsymbol{\rho})$ is referred to as the sum of absolute control errors multiplied by time and squared output sensitivity function. The output sensitivity function σ is introduced in Equation (19) because one of the goals of the FC controller design is to reduce the parametric variations effects of the controlled plant on the controlled output. Other objective functions can be defined to include the state sensitivity functions if the goal is to reduce the effects of the parametric variations of the controlled plant on the state variables.

The discrete-time objective function given in Equation (19) can be defined either directly or from continuous-time objective functions with the sampling period included in the consideration of the weighting parameter. The dynamic regimes with regard to step-type modifications of the reference and disturbance inputs should be considered.

Making use of Equation (19), the optimization problem which ensures the optimal design of the controllers resulting in fuzzy control system performance enhancement and a reduced parametric sensitivity, is defined as follows:

$$\boldsymbol{\rho}^* = \arg \min_{\boldsymbol{\rho} \in D_{\boldsymbol{\rho}}} J(\boldsymbol{\rho}), \quad (20)$$

where $\boldsymbol{\rho}^*$ is the optimal value of the vector $\boldsymbol{\rho}$ and $D_{\boldsymbol{\rho}}$ is the feasible domain of $\boldsymbol{\rho}$. The stability of the fuzzy control system should first be taken into consideration when setting the domain $D_{\boldsymbol{\rho}}$.

The PI controllers can cope with the process modeled in Equation (17) if they are inserted in 2-DOF linear control system structures (Precup *et al.*, 2009; Åström and Hägglund, 1995; Preitl and Precup, 1996, 1999). The transfer function of the PI controller is given in Equation (4). The PI controllers can be tuned by the extended symmetrical optimum (ESO) method (Preitl and Precup, 1996, 1999) to guarantee a desired tradeoff to the performance specifications (i.e., maximum values of control system performance indices) imposed to the control system using a single design parameter referred to as β ,

with the recommended values within $1 < \beta < 20$. The diagrams presented in Figure 20.4 can be used in setting the value of the design parameter β and, therefore, the tradeoff to the linear control system performance indices expressed as percent overshoot $\sigma_1[\%]$, normalized settling time $\hat{t}_s = t_s/T_\Sigma$, normalized rise time $\hat{t}_r = t_r/T_\Sigma$ and phase margin $\varphi_m[\circ]$.

The PI tuning conditions specific to the ESO method ((Preitl and Precup, 1996, 1999) are

$$k_c = \frac{1}{(\beta\sqrt{\beta}k_{EP}T_\Sigma^2)}, \quad T_i = \beta T_\Sigma, \quad k_C = \frac{1}{(\sqrt{\beta}k_{EP}T_\Sigma)}. \quad (21)$$

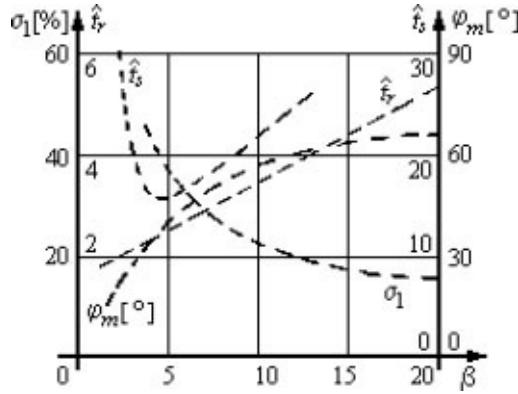


Figure 20.4: Linear control system performance indices versus the parameter β specific to the ESO method for the step-type modification of the reference input.

Figure 20.4 is important because both possible values of k_{EP} according to Equation (18) should be used in setting certain values of β , which ensure the fulfillment of the performance specifications imposed to the control system. A simple version of set-point filter which ensures the performance improvement of the linear control system by the cancellation of a zero in the closed-loop transfer function with respect to the reference input (Preitl and Precup, 1996, 1999) is

$$F(s) = \frac{1}{(1 + \beta T_\Sigma s)}. \quad (22)$$

The PI-FCs are designed starting with the linear PI controllers such that to ensure the further improvement of the control system performance indices for the nonlinear process modeled in Equation (15). The presentation and the results will be focused on the Takagi–Sugeno PI-FC-OI, but the presentation for the Mamdani PI-FC-OI and for both versions of PI-FC-II is straightforward.

In the context of the previous section, the controller parameter vector, to be determined by the CSS algorithm that solves the optimization problem, Equation (20), is

$$\rho = [\rho_1 \ \rho_2 \ \rho_3]^T = [\beta \ B_e \ \eta]^T \in \mathbf{R}^3. \quad (23)$$

The application of the CSS algorithm described as follows will give the optimal parameter vector ρ^*

$$\rho^* = [\rho_1^* \ \rho_2^* \ \rho_3^*]^T = [\beta^* \ B_e^* \ \eta^*]^T. \quad (24)$$

Equation (7) is next applied to obtain the optimal value of the last parameter of the PI-FC-OI, $B_{\Delta e}^*$. Equations (8), (9), and (10) are applied to obtain the optimal values of the parameters of other versions of PI-FCs.

20.3.1. Application of CSS Algorithm

The specific features of CSS algorithms concern the random determination of the initial positions of charged particles (CPs) and the initial velocities of CPs set to zero. Each CP has an associated magnitude of charge $q_{c,i}$ and as a result it creates an electrical field around its space. The magnitude of the charge is defined considering the quality of its solution as

$$q_{c,i} = \frac{(g_i - g_{best})}{(g_{best} - g_{worst})}, \quad i = 1, \dots, N, \quad (25)$$

where g_{best} and g_{worst} are the so far best and the worst fitness of all CPs, g_i represents the objective function value or the fitness of i th CP, and N is the total number of CPs. The separation distance r_{ij} between two CPs is defined (Precup *et al.*, 2011) as

$$r_{ij} = \frac{(\|\mathbf{X}_i - \mathbf{X}_j\|)}{\left(\left\|\frac{(\mathbf{X}_i + \mathbf{X}_j)}{2} - \mathbf{X}_{best}\right\| + \varepsilon\right)}, \quad \mathbf{X}_o \in \mathbf{R}^{q_s}, \quad o \in \{i, j, best\}, \quad (26)$$

where q_s is the dimension of the search space, \mathbf{X}_i and \mathbf{X}_j are the position vectors of i th and j th CP, respectively, \mathbf{X}_{best} is the position of the best current CP, ε , $\varepsilon > 0$, is a constant introduced to avoid singularities, and the Euclidean norm is considered in Equation (26).

The electric forces between any two CPs are used in increasing CSS algorithm's exploitation ability. The good CPs can attract the other CPs and the bad ones repel the others, proportional to their rank c_{ij} (Precup *et al.*, 2011; Kaveh and Talatahari, 2010a, 2010b, 2010c)

$$c_{ij} = \begin{cases} -1, & \text{if } g_i < g_j, \\ 1, & \text{otherwise,} \end{cases} \quad (27)$$

where the parameter c_{ij} determines the type and the degree of influence of each CP on the other CPs, considering their fitness apart from their charges.

The value of the resultant electrical force \mathbf{F}_j acting on j th CP (Precup *et al.*, 2011) is

$$\mathbf{F}_j = q_j \sum_{i,i \neq j} \left(\frac{q_i r_{ij} i_1}{a^3} + \frac{q_i i_2}{r_{ij}^2} \right) c_{ij} (\mathbf{X}_i - \mathbf{X}_j), \quad (i_1, i_2) = \begin{cases} (0, 1), & \text{if } r_{ij} \geq a, \\ (1, 0), & \text{otherwise,} \end{cases} \quad (28)$$

where $i, j = 1 \dots N$. Equation (28) shows that each CP is considered as a charged sphere

with radius a having a uniform volume charge density.

The new position (vector) $\mathbf{X}_j(k_{iter} + 1)$ and velocity (vector) $\mathbf{V}_j(k_{iter} + 1)$ of each CP is determined (Precup *et al.*, 2014; Kaveh and Talatahari, 2010a, 2010b, 2010c) in terms of

$$\begin{aligned}\mathbf{X}_j(k_{iter} + 1) &= r_{j1}k_a \left(\frac{\mathbf{F}_j}{m_j} \right) (\Delta k)^2 + r_{j2}k_v \mathbf{V}_j(k_{iter}) \Delta k + \mathbf{X}_j(k_{iter}), \\ \mathbf{V}_j(k_{iter} + 1) &= \frac{[\mathbf{X}_j(k_{iter} + 1) - \mathbf{X}_j(k_{iter})]}{\Delta k},\end{aligned}\quad (29)$$

where k_{iter} is the current iteration index which is dropped out at certain variables for the sake of simplicity, k_a is the acceleration parameter, k_v is the velocity parameter which controls the influence of the previous velocity, r_{j1} and r_{j2} are two random numbers uniformly distributed in the range $0 < r_{j1}, r_{j2} < 1$, m_j is the mass of j th CP which is set here as equal to $q_{c,j}$, and Δk is the time step set to 1.

The effect of the previous velocity and the resultant force acting on a CP can be decreased or increased based on the values of k_a and k_v , respectively. Since k_a is the parameter related to the attracting forces; selecting a large value of k_a may cause a fast convergence and choosing a small value can increase the computational time. k_v controls the exploration process. The following modifications of k_a and k_v with respect to the iteration index are applied (Precup *et al.*, 2014):

$$k_a = 3 \left(\frac{1 - k_{iter}}{k_{max}} \right), \quad k_v = 0.5 \left(\frac{1 + k_{iter}}{k_{max}} \right), \quad (30)$$

where k_{max} is the maximum number of iterations.

The CSS algorithm consists of the following steps (Preitl and Precup, 1999):

Step A. Initialize the q_s = three-dimensional search space, the number of CPs N and randomly generate the CPs' position vector $\mathbf{X}_i \in \mathbf{R}^3$.

Step B. Evaluate the CPs' fitness using Equation (19) by simulations and/or experiments conducted with the fuzzy control system accounting for the following relationship that maps the CSS algorithm onto the optimization problem, Equation (20):

$$g_i = J(\rho), \quad \mathbf{X}_i = \rho, \quad i = 1 \dots N. \quad (31)$$

Step C. Update g_{best} and g_{worst} , and update $q_{c,i}$ using Equation (25) for $i = 1 \dots N$.

Step D. Compute the total force in different directions using Equations (26), (27) and (28), and update the CPs' velocities and positions using Equation (29).

Step E. Validate the obtained vector solution \mathbf{X}_i in terms of checking the following inequality-type constraint which guarantees that the fuzzy control system with the

obtained parameter vector $\rho = \mathbf{X}_i$ ensures the convergence of the objective function (Precup *et al.*, 2011, 2014)

$$|y(t_f) - r(t_f)| \leq 0.001|r(t_f) - r(0)|, \quad (32)$$

where t_f is the final time moment, theoretically ∞ according to (19), but practically finite such that to capture all control system transients.

Step F. Increment k_{iter} and go to Step B until the maximum number of iterations is reached, i.e., $k_{iter} = k_{\max}$, and the optimal parameter vector will be that last position vector out of the N position vectors which offers the minimum value of objective (fitness) function, $\rho^* = \mathbf{X}_i$.

20.3.2. Example Concerning a Laboratory Servo System

The CSS algorithm has been applied to exemplify the optimal tuning of a Takagi–Sugeno PI-FC-OI in the position control of a laboratory servo system in terms of solving the optimization problem, Equation (20). The experimental setup is illustrated in [Figure 20.5](#). An optical encoder is used for the measurement of the angle and a tacho-generator for the measurement of the angular speed. The speed can also be estimated from the angle measurements. The PWM signals proportional with the control signal are produced by the actuator in the power interface. The main features of the experimental setup (Precup *et al.*, 2012b; Radac *et al.*, 2012) are: rated amplitude of 24 V, rated current of 3.1 A, rated torque of 15 N cm, rated speed of 3000 rpm, and weight of inertial load of 2.03 kg. The nominal values of the parameters of the process model given in Equations (15) and (17), obtained by a least squares algorithm, are $u_a = 0.15$, $u_b = 1$, $u_c = 0.15$, $k_{P0} = k_{EP0} = 140$, and $T_{\Sigma 0} = 0.92$ s, and the subscript 0 points out the nominal value of a certain parameter.

The steps A to F of the CSS algorithm have been applied, and the main results are presented as follows. The weighting parameters for the objective function defined in Equation (19) have been set to the values $\gamma^2 \in \{0, 0.142, 1.42, 14.2\}$ for the PI-FC tuning such that to ensure a reduced sensitivity with respect to k_p , and $\gamma^2 = \{0, 0.15885, 1.5885, 15.885\}$ for the PI-FC tuning such that to ensure a reduced sensitivity with respect to T_Σ , in order to reflect a ratio of $\{0, 0.1, 1, 10\}$ of the initial values of the two terms in the sum given in Equation (19).

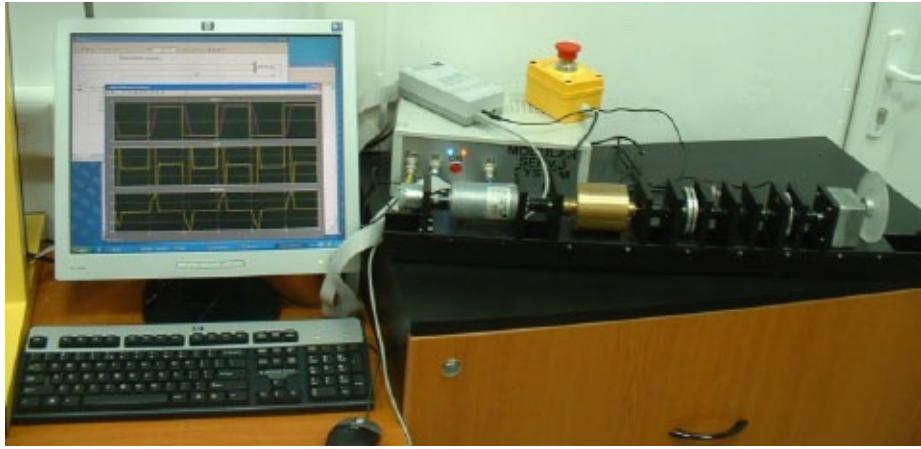


Figure 20.5: Servo system experimental setup in the Intelligent Control Systems Laboratory of the Politehnica University of Timisoara.

The variables of the objective function have been initialized taking into consideration the following boundaries which define the feasible domain of ρ

$$D_\rho = \{\beta | 3 \leq \beta \leq 17\} \times \{B_e | 20 \leq B_e \leq 40\} \times \{\eta | 0.25 \leq \eta \leq 0.75\}. \quad (33)$$

The CSS algorithm implementation is assisted by running the required digital simulations of the fuzzy control system behavior with respect to $r = 40$ rad step-type modification of the reference input. This dynamic regime has been considered in relation with the optimization problem, Equation (20).

In order to ensure a good convergence of the CSS algorithm the following parameters have been chosen on the basis of numerous trial runs and previous experience: $N = 20$, k_a and k_v computed according to Equation (30). The maximum number of iterations used as stop criterion of the CSS algorithm has been set to $k_{\max} = 100$. In order to reduce the complexity of the algorithm the volume charged density for each CP has been considered unitary, namely $a = 1$. In addition, $\varepsilon = 10^{-4}$ has been set in Equation (26).

The sampling period has been set to $T_s = 0.01$ s, which is acceptable for quasi-continuous digital control. The state sensitivity models of the fuzzy control systems with respect to k_P and T_Σ must be derived (Precup *et al.*, 2013a; David *et al.*, 2013) because they are involved in the step B of the CSS algorithm.

The optimal controller parameters and the minimum objective function values with respect for the sensitivity reduction with respect to k_P and T_Σ are summarized in [Table 20.1](#) and in [Table 20.2](#), respectively. The data contained by these tables demonstrate the algorithm's convergence to the optimum as they were obtained after repeated algorithm reiterations. All iterations have involved the evaluation of the minimum objective function using the position of each agent. All results are given as average values taken for the best five runs of the algorithm.

[Table 20.1:](#) Weighting parameter, controller parameters for the CSS algorithm-based tuning of PI-FC-OI with a reduced sensitivity with respect to k_P .

γ^2	$B_{\Delta e}^*$	B_e^*	η^*	β^*	k_c^*	T_i^*	$J(\rho^*)$
0	0.084519	40	0.75	5.14965	0.003421	4.73768	155252
0.142	0.084497	40	0.75	5.15098	0.003421	4.7389	354007
1.42	0.081739	40	0.75	5.32457	0.003365	4.8986	2149610
14.2	0.022529	20	0.25	9.65464	0.002499	8.88227	19379100

Table 20.2: Weighting parameter, controller parameters for the CSS algorithm-based tuning of PI-FC-OI with a reduced sensitivity with respect to T_Σ .

γ^2	$B_{\Delta e}^*$	B_e^*	η^*	β^*	k_c^*	T_i^*	$J(\rho^*)$
0	0.084519	40	0.75	5.14965	0.003421	4.73768	155252
0.15885	0.084735	40	0.75	5.13651	0.003426	4.72559	380819
1.5885	0.084982	40	0.75	5.12164	0.003431	4.7119	2419410
15.885	0.020722	20	0.25	10.4964	0.002396	9.65673	21711500

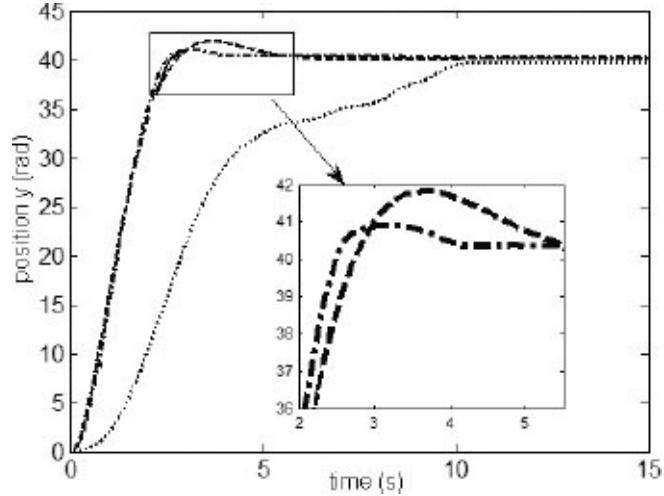


Figure 20.6: Real-time experimental results of control systems with initial PI controller (dotted), initial Takagi–Sugeno PI-FC-OI (dashed) and non-adaptive CSS algorithm-based tuned Takagi–Sugeno PI-FC-OI (dash-dotted).

Figure 20.6 illustrates the experimental results for three controller architectures, i.e., a linear control system with PI controller tuned by the ESO method for $\beta = 4$, a fuzzy control system with Takagi–Sugeno PI-FC-OI obtained from the PI controller by the modal equivalence principle, with the parameters $B_e = 40$, $\eta = 0.75$, and $B_{\Delta e} = 0.1$, and a fuzzy control system with Takagi–Sugeno PI-FC-OI tuned by the CSS algorithm. These experimental results illustrate the performance improvement offered by the CSS algorithm in the dynamic regime characterized by the $r = 40$ rad step-type modification of the reference input.

The comparisons of nature-inspired optimization algorithms can be carried out by means of several performance indices. Two such indices are the convergence speed defined as the number of evaluations of the objective functions until finding their minimum values (Precup *et al.*, 2013a), and the number of evaluations per iteration (David *et al.*, 2013). As these two indices point out how fast a solution is found, they can miss other relevant information on the overall solution’s quality. This limitation can be mitigated by another second performance index, namely the accuracy rate, defined as the standard deviation reported to the average of the solutions (Precup *et al.*, 2014).

Nevertheless, complexity analyses of the algorithms can be conducted. The results of the comparisons of the performance indices should be presented as average values for several runs of the algorithms in order to alleviate the impact of the random parameters in the algorithms.

20.4. Simulated Algorithm-Based Optimal Tuning of Input Membership Function Parameters of Takagi–Sugeno Fuzzy Models for ABS

The continuous-time state-space model of the ABS laboratory equipment is derived from the first-principle process model (David *et al.*, 2014)

$$\begin{aligned} J_1 \dot{x}_1 &= F_n r_1 \mu(\lambda) - d_1 x_1 - M_{10} - M_1, \\ J_2 \dot{x}_2 &= -F_n r_2 \mu(\lambda) - d_2 x_2 - M_{20}, \end{aligned} \quad (34)$$

where λ is the longitudinal slip (the wheel slip), J_1 and J_2 are the inertia moments of wheels, x_1 and x_2 are the angular velocities, d_1 and d_2 are the friction coefficients in wheels' axes, M_{10} and M_{20} are the static friction torques that oppose the normal rotation, M_1 is the brake torque, r_1 and r_2 are the radii of wheels, F_n is the normal force that the upper wheel pushes upon the lower wheel, $\mu(\lambda)$ is the friction coefficient, and \dot{x}_1 and \dot{x}_2 are the wheels' angular accelerations. The identification by experiment-based measurements leads to the following process parameter values (David *et al.*, 2014) that correspond to the experimental setup illustrated in Figure 20.7:

$$\begin{aligned} r_1 = r_2 &= 0.99 \text{ m}, \quad F_n = 58.214 \text{ N}, \quad J_1 = 7.53 \cdot 10^{-3} \text{ kg m}^2, \\ J_2 &= 25.6 \cdot 10^{-3} \text{ kg m}^2, \quad d_1 = 1.1874 \cdot 10^{-4} \text{ kg m}^2/\text{s}, \\ d_2 &= 2.1468 \cdot 10^{-4} \text{ kg m}^2/\text{s}, \quad M_{10} = 0.0032 \text{ N m}, \quad M_{20} = 0.0925 \text{ N m}. \end{aligned} \quad (35)$$



Figure 20.7: ABS experimental setup in the Intelligent Control Systems Laboratory of the Politehnica University of Timisoara.

The wheel slip and the nonlinear term $S(\lambda)$ are expressed as

$$\begin{aligned} \lambda &= \frac{(r_2 x_2 - r_1 x_1)}{(r_2 x_2)}, \quad x_2 \neq 0, \\ S(\lambda) &= \frac{\mu(\lambda)}{\{L[\sin \varphi - \mu(\lambda) \cos \varphi]\}}, \quad \tan \phi \neq \mu(\lambda), \end{aligned} \quad (36)$$

where $L = 0.37$ m is the arm's length which fixes the upper wheel and $\phi = 65.61^\circ$ is the angle between the normal direction in wheels' contact point and the direction of L .

The state-space equations of the nonlinear process specific to ABS (David *et al.*, 2014) are

$$\begin{aligned}\dot{x}_1 &= S(\lambda)(c_{11}x_1 + c_{12}) + c_{13}x_1 + c_{14} + (c_{15}S(\lambda) + c_{16})s_1 M_1, \\ \dot{x}_2 &= S(\lambda)(c_{21}x_1 + c_{22}) + c_{23}x_2 + c_{24} + c_{25}S(\lambda)s_1 M_1, \\ \dot{M}_1 &= c_{31}(b(u) - M_1),\end{aligned}\quad (37)$$

where u is the control signal applied to the actuator, i.e., the DC motor which drives the upper wheel, and the actuator's nonlinear model is highlighted in the third equation. The expressions of the parameters in Equation (37) (David *et al.*, 2014; Radac *et al.*, 2009) are

$$\begin{aligned}c_{11} &= \frac{r_1 d_1}{J_1}, \quad c_{12} = \frac{(M_{10} + M_g)r_1}{J_1}, \quad c_{13} = -\frac{d_1}{J_1}, \quad c_{14} = -\frac{M_{10}}{J_1}, \\ c_{15} &= \frac{r_1}{J_1}, \quad c_{16} = -\frac{1}{J_1}, \quad c_{21} = -\frac{r_2 d_1}{J_2}, \quad c_{22} = -\frac{(M_{10} + M_g)r_2}{J_2}, \\ c_{23} &= -\frac{d_2}{J_2}, \quad c_{24} = -\frac{M_{20}}{J_2}, \quad c_{25} = -\frac{r_2}{J_2}.\end{aligned}\quad (38)$$

The variable λ is considered as the controlled output of the process specific to ABS, and the notation $y = \lambda$ will be used in this context. The state variable x_1 is substituted from Equation (37) in the model, Equation (38), and the continuous-time state-space equations of the process specific to ABS (David *et al.*, 2014) is transformed into

$$\dot{\mathbf{x}} = \begin{bmatrix} z_1(\lambda, x_2) & 0 & z_3(\lambda, x_2) \\ 0 & z_{40}(\lambda) & z_5(\lambda) \\ 0 & 0 & -c_{31} \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 0 \\ c_{31} \end{bmatrix} b(u) + \mathbf{d}(\lambda, x_2), \quad (39)$$

where the state vector is

$$\mathbf{x} = [\lambda \ x_2 \ M_1]^T, \quad (40)$$

$z_1(\lambda, x_2)$, $z_3(\lambda, x_2)$, $z_{40}(\lambda)$, and $z_5(\lambda)$ are nonlinear functions, and $\mathbf{d}(\lambda, x_2)$ is the disturbance input vector.

The approach to the SA-based optimal tuning of the parameters of input m.f.s of Takagi–Sugeno fuzzy models for ABS is expressed in terms of the following steps presented as follows and based on the modal equivalence principle (Galichet and Foulloy, 1995). This approach is different to the sector nonlinearity approach (Ohtake *et al.*, 2001), but the SA algorithm can be applied after the initial Takagi–Sugeno fuzzy models are obtained by the sector nonlinearity approach.

Step 1. Set the structure of the Takagi–Sugeno fuzzy model, that means the number of operating points which is equal to the number of rules n_R of the Takagi–Sugeno fuzzy

model, the number of input linguistic terms of the variables λ , x_2 and M_1 , the shapes of the m.f.s of the input linguistic terms, the operators in the inference engine, and the defuzzification method. The SUM and PROD operators in the inference engine, and the weighted average defuzzification method are applied for this process.

Step 2. Linearize the continuous-time state-space equations of the process, Equation (37) and the output equation which is the first equation in Equation (36) at n_R important operating points. This yields in n_R linearized continuous-time local process models. The local process models are placed in the rule consequents of the continuous-time Takagi–Sugeno fuzzy model, and they are related to the modal values of the input m.f.s which are exactly the coordinates of the operating points in terms of the modal equivalence principle (Galichet and Foulloy, 1995). The complete rule-base of the continuous-time Takagi–Sugeno fuzzy model is the following particular form of the continuous-time Takagi–Sugeno fuzzy model, Equation (14):

$$\begin{aligned} \text{Rule } i : & \text{ IF } \lambda(t) \text{ IS } T_{\lambda,l}^i \text{ AND } x_2(t) \text{ IS } T_{x_2,p}^i \text{ AND } M_1(t) \text{ IS } T_{M_1,q}^i \\ & \text{THEN } \begin{cases} \dot{\mathbf{x}}(t) = \mathbf{A}_i \mathbf{x}(t) + \mathbf{B}_i u(t) \\ y_m(t) = \mathbf{C}_i \mathbf{x}(t) \end{cases}, \quad i = 1 \dots n_R, \end{aligned} \quad (41)$$

where $T_{v,l}^i$, $T_{v,p}^i$, and $T_{v,q}^i$ are the linguistic terms $T_{\lambda,l}$, $T_{x_2,p}$, and $T_{M_1,q}$ of the input variables (scheduling variables) $\lambda(t)$, $x_2(t)$, and $M_1(t)$, respectively, with the recommended numbers of linguistic terms and of ranges of indices $l = 1 \dots 5$, $p \in \{1, 2\}$ and $q = 1, v \in \{\lambda, x_2, M_1\}$, and the matrices \mathbf{A}_i , \mathbf{B}_i , and \mathbf{C}_i in the rule consequents have appropriate dimensions.

Step 3. Set the sampling period T_s and discretize the n_R models in the rule consequents of the Takagi–Sugeno fuzzy model, Equation (41) accepting the zero-order hold. The complete rule-base of the discrete-time Takagi–Sugeno fuzzy model is

$$\begin{aligned} \text{Rule } i : & \text{ IF } \lambda(k) \text{ IS } T_{\lambda,l}^i \text{ AND } x_2(k) \text{ IS } T_{x_2,p}^i \text{ AND } M_1(k) \text{ IS } T_{M_1,q}^i \\ & \text{THEN } \begin{cases} \mathbf{x}(k+1) = \mathbf{A}_{d,i} \mathbf{x}(k) + \mathbf{B}_{d,i} u(k) \\ y_m(k) = \mathbf{C}_{d,i} \mathbf{x}(k) \end{cases}, \quad i = 1 \dots n_R, \end{aligned} \quad (42)$$

where $y_m(k)$ is the discrete-time Takagi–Sugeno fuzzy model output, and the matrices $\mathbf{A}_{d,i}$, $\mathbf{B}_{d,i}$, and $\mathbf{C}_{d,i}$ in the rule consequents have appropriate dimensions exemplified as follows.

The recommended value of the sampling period is $T_s = 0.01$ s which accounts for the process dynamics (Precup *et al.*, 2012c). The disturbance input vector is neglected in Equations (41) and (42) for the sake of simplicity and for dealing with models which are close to real-world applications.

Step 4. Define the optimization problem that leads to optimal Takagi–Sugeno fuzzy models

$$\rho^* = \arg \min_{\rho \in D} J(\rho),$$

$$J(\rho) = \sqrt{\frac{1}{N} \sum_{k=1}^N (y(k, \rho) - y_m(k, \rho))^2} = \sqrt{\frac{1}{N} \sum_{k=1}^N (e_m(k, \rho))^2}, \quad (43)$$

where ρ is the parameter vector of the Takagi–Sugeno fuzzy model and also the vector variable of the objective function $J(\rho)$, ρ^* is the optimal parameter vector of the Takagi–Sugeno fuzzy model and the solution to Equation (43), $y(k, \rho)$ is the process output at k th sampling interval, $y_m(k, \rho)$ is the fuzzy model output, the modeling error is

$$e_m(k, \rho) = y(k, \rho) - y_m(k, \rho), \quad (44)$$

D is the feasible domain of ρ and N is the length of the discrete time horizon. The elements of the vector ρ are a part of the parameters of the input m.f. parameters. $J(\rho)$ in Equation (43) is the root mean square error (RMSE) between the model output (the wheel slip) of the Takagi–Sugeno fuzzy model and of the real-world process.

Step 5. Apply the SA algorithm described in the next section to solve the optimization problem, Equation (43) that leads to the vector ρ^* . The elements of the vector ρ^* are the optimal input m.f. parameters, and they belong to the optimal Takagi–Sugeno fuzzy model.

20.4.1. Application of SA Algorithm

The operating mechanism of SA algorithms uses a probabilistic framework to accept the solution on the basis of the analogy with the temperature decrease in metallurgy (Kirkpatrick *et al.*, 1983; Geman and Geman, 1984). Considering the initial solution represented by the vector $\zeta \in \mathbf{R}^{q_s}$ (q_s is the dimension of the search space) with the fitness value $g(\zeta)$ of the fitness function (i.e., of the objective function) g , $g : \mathbf{R}^{q_s} \rightarrow \mathbf{R}$, the new probable solution is represented by the vector $\psi \in \mathbf{R}^{q_s}$ which is chosen such that to belong to the vicinity of ζ . In order to make the SA algorithm more computationally efficient, two additional iteration indices are defined (Precup *et al.*, 2013a), namely the success rate s_r and the rejection rate r_r . The success rate s_r aims the acceleration of the cooling process by forcing a jump in temperature when the minimum value of the fitness function changes for a preset number of times at the same temperature level. The rejection rate r_r is proposed as an alternative index for the convergence of the algorithm, and it is reset only when small values of the fitness function are found and not when the temperature cools.

The steps of the SA algorithm applied in the step 5 of the optimal tuning approach are:

Step A. Generate the initial solution, i.e., conduct the following operations: generate a random initial solution ζ and calculate its fitness value $g(\zeta)$, set the minimum temperature θ_{\min} , using the notation k_{iter} for the current iteration index, initialize the maximum allowed

number of iterations k_{\max} for each temperature step, the maximum accepted success rate s_r , the maximum accepted rejection rate r_r , and the minimum accepted value of the fitness function g_{\min} , set the initial temperature θ_0 , i.e., the temperature $\theta_{k_{iter}}$ for $k_{iter} = 0$, and set the initial rejection rate $r_r = 0$.

Step B. Set the initial value of the iteration index $k_{iter} = 0$ and the initial success rate $s_r = 0$.

Step C. Generate a new probable solution ψ in the vicinity of ζ by disturbing ζ , and calculate its fitness value $g(\psi)$.

Step D. Accept or not the new solution by means of the change of fitness expressed as the difference $\Delta g_{\psi\zeta}$

$$\Delta g_{\psi\zeta} = g(\psi) - g(\zeta). \quad (45)$$

If $\Delta g_{\psi\zeta} \leq 0$, accept $\zeta = \psi$ as the new vector solution. Otherwise, set the random parameter q_k , $0 \leq q_k \leq 1$, and calculate the probability p_ψ of ψ to be the next solution

$$p_\psi = \exp\left(\frac{-\Delta g_{\psi\zeta}}{\theta_{k_{iter}}}\right). \quad (46)$$

If $p_\psi > q_k$, $\zeta = \psi$ is the new solution.

Step E. If the new solution is accepted, then update the new solution and g , increment k_{iter} and reset $r_r = 0$. Otherwise, increment r_r . If r_r has reached its maximum value $r_{r\max}$, go to step I; otherwise, continue with the next step.

Step F. Increment s_r . If s_r has reached its maximum value $s_{r\max}$, continue with the next step; otherwise, increment k_{iter} . If k_{iter} has reached its maximum value k_{\max} , continue with the next step; otherwise, go to step C.

Step G. The temperature is decreased in terms of the temperature decrement rule, referred to also as the cooling schedule, which gives the next temperature $\theta_{k_{iter}+1}$

$$\theta_{k_{iter}+1} = \alpha_{cs} \theta_{k_{iter}}, \quad (47)$$

where $\alpha_{cs} = \text{const}$, $\alpha_{cs} < 1$.

Step H. If $\theta_{k_{iter}} > \theta_{\min}$ or $g(\zeta) > g_{\min}$, go to step C. Otherwise, continue with the next step.

Step I. Validate the obtained vector solution ζ by checking the following inequality-type constraint which guarantees that the Takagi–Sugeno fuzzy model with the obtained model parameters $\rho = \zeta$ ensures the convergence of the objective function:

$$|y(N, \rho) - y_m(N, \rho)| \leq \varepsilon_y |y(N, \rho) - y(1, \rho)|, \quad (48)$$

where the value $\varepsilon_y = 0.001$ is recommended for a 2% settling time, and it is taken from the optimal tuning of Takagi–Sugeno FCs (Precup *et al.*, 2013a, 2013b; David *et al.*, 2013).

Step J. The algorithm is stopped, and the last validated vector solution ζ is the solution to the optimization problem, Equation (43).

This SA algorithm is mapped onto the optimization problem, Equation (43) by means of the following relations between the parameter vectors ζ and ψ in the SA algorithm and the parameter vector ρ in the optimization problem:

$$\psi = \rho, \quad \zeta = \rho \quad (49)$$

and between the fitness function g in the SA algorithm and the objective function J in the optimization problem:

$$g(\psi) = J(\rho), \quad g(\zeta) = J(\rho). \quad (50)$$

In this context, the last new solution found will be the vector solution ρ^* to the optimization problem, Equation (43):

$$\rho^* = \zeta. \quad (51)$$

20.4.2. Example Concerning a Laboratory ABS

The optimal tuning approach has been applied to obtain optimal Takagi–Sugeno fuzzy models of the laboratory ABS. A part of the results and of the implementation details related to the Takagi–Sugeno fuzzy model characterized by $n_R = 3$ is presented as follows.

The largest domains of variation of the state variables in all ABS operating regimes are $0 \leq \lambda \leq 1.1$, $0 \leq x_2 \leq 180$, and $0 \leq M_1 \leq 11$. As pointed out in the step 2, the first input (scheduling) variable, λ , uses five linguistic terms, $T_{\lambda,l}$, $l = 1 \dots 5$. The triangular m.f.s of these linguistic terms are

$$\mu_{LT_{\lambda,l}} : [0, 1.1] \rightarrow [0, 1],$$

$$\mu_{LT_{\lambda,l}}(\lambda) = \begin{cases} 0, & \lambda < a_{\lambda,l}, \\ 1 + \frac{(\lambda - b_{\lambda,l})}{(b_{\lambda,l} - a_{\lambda,l})}, & a_{\lambda,l} \leq \lambda < b_{\lambda,l}, \\ 1 - \frac{(\lambda - b_{\lambda,l})}{(c_{\lambda,l} - b_{\lambda,l})}, & b_{\lambda,l} \leq \lambda < c_{\lambda,l}, \\ 0, & \lambda \geq c_{\lambda,l}, \end{cases} \quad l = 1 \dots 5. \quad (52)$$

The parameters $a_{\lambda,l}$, $l = 1 \dots 5$ and $c_{\lambda,l}$, $l = 1 \dots 5$, are the feet of the triangular m.f.s, they are variable and they belong to the vector variable ρ of the objective function, and the parameters $b_{\lambda,l}$, $l = 1 \dots 5$, which stand for the modal values of the m.f.s, are fixed and they take the values $b_{\lambda,1} = 0.1$, $b_{\lambda,2} = 0.2$, $b_{\lambda,3} = 0.4$, $b_{\lambda,4} = 0.8$, and $b_{\lambda,5} = 1$. The second input

variable, x_2 , uses two linguistic terms, $T_{x2,p}$, $p \in \{1, 2\}$, with the triangular m.f.s of type, Equation (52). The parameters $a_{x2,p}$, $p \in \{1, 2\}$, and $c_{x2,p}$, $p \in \{1, 2\}$, are the feet of the triangular m.f.s, they are variable and they belong to ρ , and the parameters $b_{x2,p}$, $p \in \{1, 2\}$, are fixed and they take the values $b_{x2,1} = 50$ and $b_{x2,2} = 150$. The third input variable, M_1 , uses one linguistic term, $T_{M_1,1}$, with the triangular m.f. of type, Equation (52). The parameters $a_{M_1,1}$ and $c_{M_1,1}$, which represent the feet of the triangular m.f.s, are variable and they belong to ρ , and the parameter $b_{M_1,1}$ is fixed, $b_{M_1,1} = 10$. Therefore, the parameter vector ρ of the Takagi–Sugeno fuzzy model that will be obtained by the application of the SA algorithm is

$$\rho = [a_{\lambda,1} \ c_{\lambda,1} \ a_{\lambda,2} \ c_{\lambda,2} \ a_{\lambda,3} \ c_{\lambda,3} \ a_{\lambda,4} \ c_{\lambda,4} \ a_{\lambda,5} \ c_{\lambda,5} \ a_{x_2,1} \ c_{x_2,1} \ a_{x_2,2} \ c_{x_2,2} \ a_{M_1,1} \ c_{M_1,1}] \quad (53)$$

and the feasible domain in Equation (43) is

$$\begin{aligned} D = & [-0.01, 0.01] \times [0.15, 0.25] \times [0.05, 0.15] \times [0.35, 0.45] \times [0.15, 0.25] \\ & \times [0.75, 0.85] \times [0.35, 0.45] \times [0.95, 1.05] \times [0.75, 0.85] \times [1.05, 1.15] \\ & \times [-0.1, 0.1] \times [130.9, 170.1] \times [40, 60] \times [170, 190] \times [-0.5, 0.5] \\ & \times [10.5, 11.5] \in \mathbf{R}^{16}. \end{aligned} \quad (54)$$

Setting the sampling period $T_s = 0.01$ s, the control signal u applied to the laboratory ABS has been generated as two weighted sums (one for training and one for validation) of pseudo-random binary signals. $N = 10000$ data points have been used in the training, and $N = 25000$ data points have been used for validation. The variation of u versus time is presented in Figure 20.8 that gives the input data for validation (testing). The state-space model matrices in the rule consequents of the rules 1 and 10 of the Takagi–Sugeno fuzzy model, Equation (42) are

$$\begin{aligned} \mathbf{A}_{d,1} &= \begin{bmatrix} 0.9441 & -0.0025 & 0.0194 \\ -1.0335 & 1.0012 & -0.0601 \\ 0 & 0 & 0.8157 \end{bmatrix}, \quad \mathbf{B}_{d,1} = \begin{bmatrix} 0.0021 \\ -0.0059 \\ 0.1843 \end{bmatrix}, \\ & \mathbf{C}_{d,1} = [1 \ 0 \ 0], \\ \mathbf{A}_{d,10} &= \begin{bmatrix} 1.0041 & -0.0003 & 0.0069 \\ -0.3192 & 1 & -0.0519 \\ 0 & 0 & 0.8157 \end{bmatrix}, \quad \mathbf{B}_{d,10} = \begin{bmatrix} 0.0007 \\ -0.0054 \\ 0.1843 \end{bmatrix}, \\ & \mathbf{C}_{d,10} = [1 \ 0 \ 0]. \end{aligned} \quad (55)$$

The first set of experimental results is presented in Figure 20.9 as the outputs of the real-world process (the ABS laboratory equipment) and of the initial akagi–Sugeno fuzzy model before the application of SA algorithm, i.e., with the parameter vector ρ

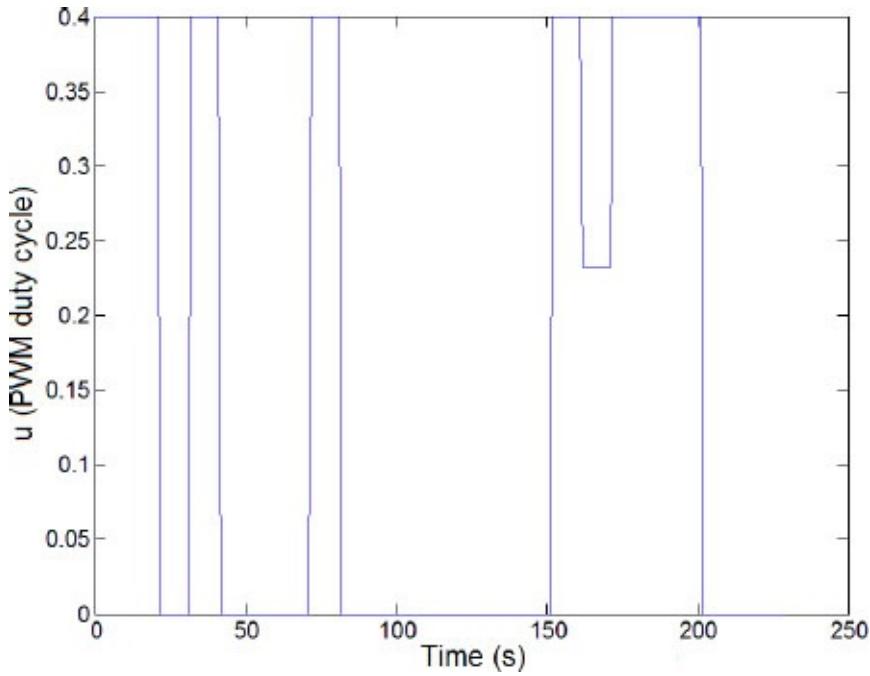


Figure 20.8: Control signal versus time used in the validation (testing) data.

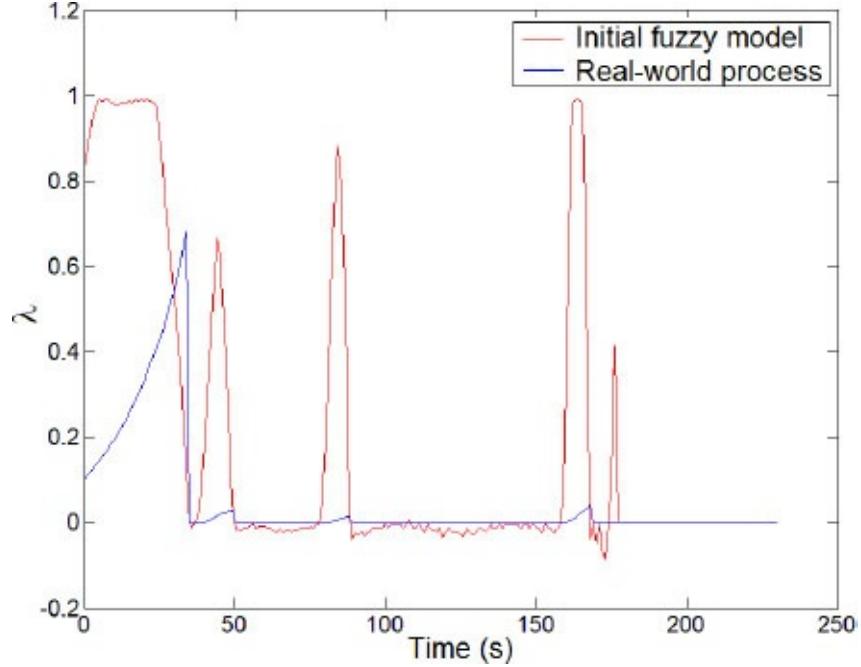


Figure 20.9: Real-time experimental results: Wheel slip λ versus time for the initial Takagi–Sugeno fuzzy model and for the real-world process considering the validation dataset.

$$\rho = [0 \ 0.2 \ 0.1 \ 0.4 \ 0.2 \ 0.8 \ 0.4 \ 1 \ 0.8 \ 1.1 \ 0 \ 150 \ 50 \ 180 \ 0 \ 11]^T. \quad (56)$$

The results presented in Figure 20.9 for the validation dataset illustrate the large difference between the Takagi–Sugeno fuzzy model and the real-world process, and they justify the need for the optimal tuning of the parameters of Takagi–Sugeno fuzzy models.

The maximum success and rejection rates in the SA algorithm have been set to $r_{r\max} = 100$ and to $s_{r\max} = 50$. A limit of $k_{\max} = 80$ iterations for each temperature step has been imposed. The SA algorithm has been stopped after reaching a final temperature of $\theta_{80} = 0.093246$. The initial temperature has been set to $\theta_0 = 1$, and the minimum temperature

has been set to $\theta_{\min} = 10^{-8}$. The value of the parameter in the temperature decrement rule has been set to $\alpha_{cs} = 0.8$. The minimum accepted value of the fitness function (i.e., of the objective function) has been set to $g_{\min} = 10^{-4}$.

The final solution obtained by the SA algorithm is and it corresponds to the final value $J(\rho^*) = 0.74989$ of the objective function that shows a decrease from the initial value $J(\rho) = 0.95373$.

$$\rho^* = [0.0086 \ 0.1614 \ 0.09901 \ 0.4067 \ 0.2295 \ 0.8471 \ 0.4019 \ 0.9578 \\ 0.847 \ 1.12 \ 0.02202 \ 169 \ 59.85 \ 180.5 \ -0.3227 \ 11.04]^T, \quad (57)$$

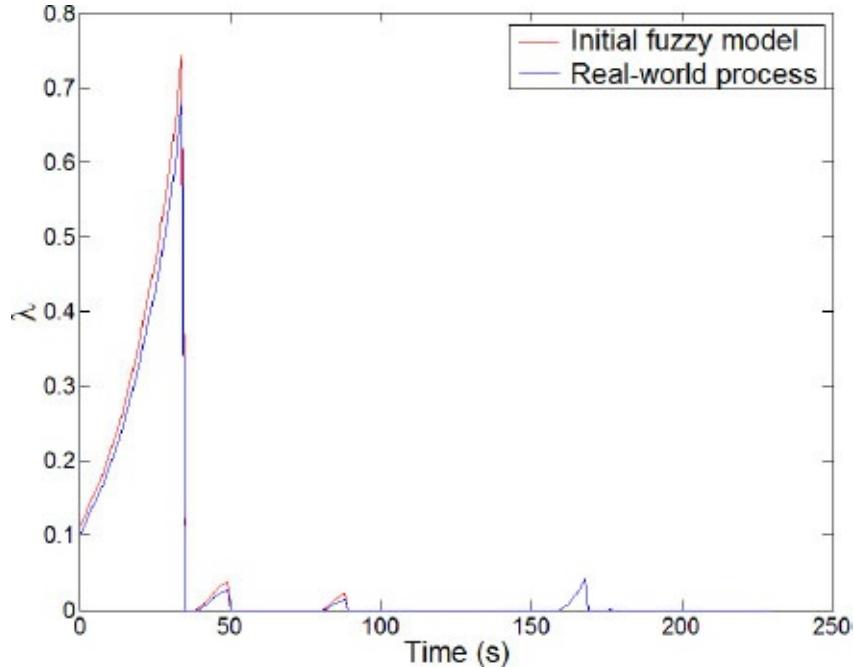


Figure 20.10: Real-time experimental results: Wheel slip λ versus time for the Takagi–Sugeno fuzzy model after optimization by the SA algorithm and for the real-world process considering the validation dataset.

The second set of experimental results is presented in [Figure 20.10](#) as the output of the real-world process and of the Takagi–Sugeno fuzzy model after the application of the SA algorithm [i.e., with the parameter vector ρ^* given in Equation (57)], for the validation dataset. [Figure 20.10](#) shows that the performance of the Takagi–Sugeno fuzzy model is improved by the application of the SA algorithm which optimizes the input m.f. parameters because reduced modeling errors are obtained.

The aspects concerning the comparisons of nature-inspired optimization algorithms presented in [Section 20.3](#) are applied to the comparison of the SA algorithm with other algorithms as well. Once again we recommend the presentation of the results of the comparisons of the performance indices as average values for several runs of the algorithms; this offers fair comparisons because the impact of the random parameters in the algorithms is alleviated.

20.5. Conclusions and Outlook

This chapter has presented aspects concerning the nature-inspired optimization of FCs and fuzzy models. The presentation has been focused on Takagi–Sugeno and Mamdani PI-FCs which are rather general as the presented design methodology can easily be applied to PD-FCs and extended to PID-FCs.

Once the optimization problems are defined, the nature-inspired optimization algorithms can be applied to get the optimal tuning of the parameters of FCs and fuzzy models. The chapter has given the exemplification of two such algorithms, the SA and the CSS. The results can be extended with no major difficulties to other algorithms. We advise the reduction of the number of parameters tuned by the nature-inspired algorithms in order to have a reasonable dimension of the search space which will offer an efficient search for the optimal solution.

The major shortcoming of nature-inspired algorithms in optimal design and tuning is represented by the large number of evaluations of the objective functions. This can be solved by the inclusion of gradient information that can be included from data-driven approaches.

Acknowledgments

This work was supported by a grant of the Romanian National Authority for Scientific Research, CNCS–UEFISCDI, Project No. PN-II-ID-PCE-2011-3-0109, and by a grant in the framework of the Partnerships in priority areas — PN II program of the Romanian National Authority for Scientific Research ANCS, CNDI–UEFISCDI, project number PN-II-PT-PCCA-2011-3.2-0732. The authors would like to thank M.Sc. Ramona-Bianca Grad and Dr. Mircea-Bogdan Rădac for their precious support in the experimental part.

References

- Almaraashi, M., John, R., Coupland, S. and Hopgood, A. (2010). Time series forecasting using a TSK fuzzy system tuned with simulated annealing. In *Proc. 2010 IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE 2010)*. Barcelona, Spain, pp. 1–6.
- Angelov, P. and Yager, R. (2012). A new type of simplified fuzzy rule-based systems. *Int. J. Gen. Syst.*, 41, pp. 163–185.
- Angelov, P. and Yager, R. (2013). Density-based averaging—A new operator for data fusion. *Inf. Syst.*, 222, pp. 163–174.
- Åström, K. J. and Hägglund, T. (1995). *PID Controllers Theory:Design and Tuning*. North Carolina, US: Instrument Society of America, Research Triangle Park.
- Baranyi, P., Tikk, D., Yam, Y. and Patton, R. J. (2003). From differential equations to PDC controller design via numerical transformation. *Comput. Ind.*, 51, pp. 281–297.
- Baranyi, P., Yam, Y. and Varlaki, P. (2013). *TP Model Transformation in Polytopic Model-Based Control*. Boca Raton, FL: Taylor & Francis.
- Batyryshin, I., Kaynak, O. and Rudas, I. J. (2002). Fuzzy modeling based on generalized conjunction operations. *IEEE Trans. Fuzzy Syst.*, 10, pp. 678–683.
- Bingül, Z. and Karahan, O. (2011). A fuzzy logic controller tuned with PSO for 2 DOF robot trajectory control. *Expert Syst. Appl.*, 38, pp. 1017–1031.
- Castillo, O. and Melin, P. (2012a). A review on the design and optimization of interval type-2 fuzzy controllers. *Appl. Soft Comput.*, 12, pp. 1267–1278.
- Castillo, O. and Melin, P. (2012b). Optimization of type-2 fuzzy systems based on bio-inspired methods: A concise review. *Inf. Sci.*, 205, pp. 1–19.
- Chang, Y.-H., Tao, C.-W., Lin, H.-W. and Taur, J.-S. (2012). Fuzzy sliding-mode control for ball and beam system with fuzzy ant colony optimization. *Expert Syst. Appl.*, 39, pp. 3624–3633.
- Chen, D., Wang, J., Zou, F., Zhang, H. and Hou, W. (2012). Linguistic fuzzy model identification based on PSO with different length of particles. *Appl. Soft Comput.*, 12, pp. 3390–3400.
- Das, S., Pan, I. and Das, S. (2013). Fractional order fuzzy control of nuclear reactor power with thermal-hydraulic effects in the presence of random network induced delay and sensor noise having long range dependence. *Energy Convers. Manage.*, 68, pp. 200–218.
- David, R.-C., Dragos, C.-A., Bulzan, R.-G., Precup, R.-E., Petriu, E. M. and Radac, M.-B. (2012). An approach to fuzzy modeling of magnetic levitation systems. *Int. J. Artif. Intell.*, 9, pp. 1–18.
- David, R.-C., Precup, R.-E., Petriu, E. M., Radac, M.-B. and Preitl, S. (2013). Gravitational search algorithm-based design of fuzzy control systems with a reduced parametric sensitivity. *Inf. Sci.*, 247, pp. 154–173.
- David, R.-C., Grad, R.-B., Precup, R.-E., Radac, M.-B., Dragos, C.-A. and Petriu, E. M. (2014). An approach to fuzzy modeling of anti-lock braking systems. In Snášel, V., Krömer, P., Köppen, M. and Schaefer, G. (eds.), *Advances in Intelligent Systems and Computing*, Vol. 223. Cham, Heidelberg, New York, Dordrecht, London: Springer-Verlag, pp. 83–93.
- Dragos, C.-A., Precup, R.-E., David, R.-C., Preitl, S., Stinean, A.-I. and Petriu, E. M. (2013). Simulated annealing-based optimization of fuzzy models for magnetic levitation systems. In *Proc. 2013 Joint IFSA World Congr. NAFIPS Annu. Meet*. Edmonton, AB, Canada, pp. 286–291.
- Feriyonika and Dewantoro, G. (2013). Fuzzy sliding mode control for enhancing injection velocity performance in injection molding machine. *Int. J. Artif. Intell.*, 10, pp. 75–87.
- Formentin, S., Karimi, A. and Savaresi, S. M. (2013a). Optimal input design for direct data-driven tuning of model-reference controllers. *Autom.*, 49, pp. 1874–1882.
- Formentin, S., van Heusden, K. and Karimi, A. (2013b). A comparison of model-based and data-driven controller tuning. *Int. J. Adapt. Control Signal Process.* doi: 10.1002/acs.2415.
- Galichet, S. and Foulloy, L. (1995). Fuzzy controllers: Synthesis and equivalences. *IEEE Trans. Fuzzy Syst.*, 3, pp. 140–148.

- Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distribution and the Bayesian restoration in images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6, pp. 721–741.
- Haber, R. E., Haber-Haber, R., Jiménez, A. and Galán, R. (2009). An optimal fuzzy control system in a network environment based on simulated annealing. An application to a drilling process. *Appl. Soft Comput.*, 9, pp. 889–895.
- Haber, R. E., del Toro, R. M. and Gajate, A. (2010). Optimal fuzzy control system using the cross-entropy method. A case study of a drilling process. *Inf. Sci.*, 180, pp. 2777–2792.
- Hjalmarsson, H., Gevers, M., Gunnarsson, S. and Lequin, O. (1998). Iterative feedback tuning: theory and applications. *IEEE Control Syst. Mag.*, 18, pp. 26–41.
- Ho, D. T. and Garibaldi, J. M. (2013). An improved optimisation framework for fuzzy time-series prediction. In *Proc. 2013 IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE 2013)*. Hyderabad, India, pp. 1–8.
- Hou, Z.-S. and Wang, Z. (2013). From model-based control to data-driven control: Survey, classification and perspective. *Inf. Sci.*, 235, pp. 3–35.
- Jain, R., Sivakumaran, N. and Radhakrishnan, T. K. (2011). Design of self tuning fuzzy controllers for nonlinear systems. *Expert Syst. Appl.*, 38, pp. 4466–4476.
- Jiang, H., Kwong, C. K., Chen, Z. and Ysim, Y. C. (2012). Chaos particle swarm optimization and T-S fuzzy modeling approaches to constrained predictive control. *Expert Syst. Appl.*, 39, pp. 194–201.
- Johanyák, Z. C. (2010). Student evaluation based on fuzzy rule interpolation. *Int. J. Artif. Intell.*, 5, pp. 37–55.
- Kaveh, A. and Talatahari, S. (2010a). A novel heuristic optimization method: charged system search. *Acta Mech.*, 213, pp. 267–289.
- Kaveh, A. and Talatahari, S. (2010b). Optimal design of truss structures via the charged system search algorithm. *Struct. Multidisp. Optim.*, 37, pp. 893–911.
- Kaveh, A. and Talatahari, S. (2010c). A charged system search with a fly to boundary method for discrete optimum design of truss structures. *Asian J. Civ. Eng. (Build. Housing)*, 11, pp. 277–293.
- Kayacan, E., Oniz, Y., Aras, A. C., Kaynak, O. and Abiyev, R. (2011). A servo system control with time-varying and nonlinear load conditions using type-2 TSK fuzzy neural system. *Appl. Soft Comput.*, 11, pp. 5735–5744.
- Kirkpatrick, S., Gelatt, C. D. Jr. and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220, pp. 671–680.
- Kóczy, L. T. (1996). Fuzzy IF-THEN rule models and their transformation into one another. *IEEE Trans. Syst., Man, Cybern., Part A: Syst. Humans*, 26, pp. 621–637.
- Kumaresan, N. (2011). Optimal control for stochastic linear quadratic singular periodic neuro Takagi–Sugeno (T-S) fuzzy system with singular cost using ant colony programming. *Appl. Math. Model.*, 35, pp. 3797–3808.
- Li, C., Zhou, J., Xiao, J. and Xiao, H. (2013). Hydraulic turbine governing system identification using T-S fuzzy model optimized by chaotic gravitational search algorithm. *Eng. Appl. Artif. Intell.*, 26, pp. 2073–2082.
- Linda, O. and Manic, M. (2011a). Interval type-2 fuzzy voter design for fault tolerant systems. *Inf. Sci.*, 181, pp. 2933–2950.
- Linda, O. and Manic, M. (2011b). Uncertainty-robust design of interval type-2 fuzzy logic controller for delta parallel robot. *IEEE Trans. Ind. Inf.*, 7, pp. 661–670.
- Liu, G. and Yang, W. (2000). Learning and tuning of fuzzy membership functions by simulated annealing algorithm. In *Proc. 2000 IEEE Asia-Pac. Conf. Circuits Syst. (APCCAS 2000)*. Tianjin, China, pp. 367–370.
- Lu, H.-C. and Liu, H.-K. (2013). Ant colony fuzzy neural network controller for cruising vessel on river. *Appl. Ocean Res.*, 42, pp. 43–54.
- Melin, P., Astudillo, L., Castillo, O., Valdez, F. and Garcia, M. (2013). Optimal design of type-2 and type-1 fuzzy tracking controllers for autonomous mobile robots under perturbed torques using a new chemical optimization paradigm. *Expert Syst. Appl.*, 40, pp. 3185–3195.
- Obradović, D., Konjović, Z., Pap, E. and Rudas, I. J. (2013). Linear fuzzy space-based road lane model and detection. *Knowl.-Based Syst.*, 38, pp. 37–47.

- Oh, S.-K., Jang, H.-J. and Pedrycz, W. (2011). A comparative experimental study of type-1/type-2 fuzzy cascade controller based on genetic algorithms and particle swarm optimization. *Expert Syst. Appl.*, 38, pp. 11217–11229.
- Ohtake, H., Tanaka, K. and Wang, H. O. (2001). Fuzzy modeling via sector nonlinearity concept. In *Proc. Joint 9th IFSA World Congr. 20th NAFIPS Int. Conf.* Vancouver, BC, Canada, 1, pp. 127–132.
- Onieva, E., Milanés, V., Villagrá, J., Pérez, J. and Godoy, J. (2012). Genetic optimization of a vehicle fuzzy decision system for intersections. *Expert Syst. Appl.*, 39, pp. 13148–13157.
- Owens, D. H., Freeman, C. T. and Thanh, V. D. (2013). Norm-optimal iterative learning control with intermediate point weighting: theory, algorithms, and experimental evaluation. *IEEE Trans. Control Syst. Technol.*, 21, pp. 999–1007.
- Pedrycz, W. and Song, M. (2012). A genetic reduction of feature space in the design of fuzzy models. *Appl. Soft Comput.*, 12, pp. 2801–2816.
- Pérez, J., Milanés, V., Godoy, J., Villagrá, J. and Onieva, E. (2013). Cooperative controllers for highways based on human experience. *Expert Syst. Appl.*, 40, pp. 1024–1033.
- Precup, R.-E., Preitl, S., Tar, J. K., Fodor, J., Ursache, I.-B. and Clep, P. A. (2008). Low-cost fuzzy logic approach to ship course control. In *Proc. 50th Int. Symp. ELMAR-2008*. Zadar, Croatia, 2, pp. 423–426.
- Precup, R.-E., Preitl, S., Petriu, E. M., Tar, J. K., Tomescu, M. L. and Pozna, C. (2009). Generic two-degree-of-freedom linear and fuzzy controllers for integral processes. *J. Franklin Inst.*, 346, pp. 980–1003.
- Precup, R.-E. and Hellendoorn, H. (2011). A survey on industrial applications of fuzzy control. *Comput. Ind.*, 62, pp. 213–226.
- Precup, R.-E., David, R.-C., Petriu, E. M. and Preitl, S. (2011). Optimal fuzzy controllers tuned by charged system search algorithms. In *Proc. 2011 Online Conf. Soft Comput. Ind. Appl. (WSC 16)*. pp. 1–10.
- Precup, R.-E., David, R.-C., Petriu, E. M., Preitl, S. and Radac, M.-B. (2012a). Fuzzy control systems with reduced parametric sensitivity based on simulated annealing. *IEEE Trans. Ind. Electron.*, 59, pp. 3049–3061.
- Precup, R.-E., David, R.-C., Petriu, E. M., Preitl, S. and Radac, M.-B. (2012b). Novel adaptive gravitational search algorithm for fuzzy controlled servo systems. *IEEE Trans. Ind. Inf.*, 8, pp. 791–800.
- Precup, R.-E., Spataru, S. V., Radac, M.-B., Petriu, E. M., Preitl, S., Dragos, C.-A. and David, R.-C. (2012c). Experimental results of model-based fuzzy control solutions for a laboratory antilock braking system. In Hippe, Z. S., Kulikowski, J. L. and Mroczek, T. (eds.), *Advances in Intelligent and Soft Computing*, Vol. 99. Berlin, Heidelberg: Springer-Verlag, pp. 223–234.
- Precup, R.-E., David, R.-C., Petriu, E. M., Radac, M.-B., Preitl, S. and Fodor, J. (2013a). Evolutionary optimization-based tuning of low-cost fuzzy controllers for servo systems. *Knowl.-Based Syst.*, 38, pp. 74–84.
- Precup, R.-E., David, R.-C., Petriu, E. M., Preitl, S. and Radac, M.-B. (2013b). Fuzzy logic-based adaptive gravitational search algorithm for optimal tuning of fuzzy controlled servo systems. *IET Control Theory Appl.*, 7, pp. 99–107.
- Precup, R.-E., Radac, M.-B., Petriu, E. M., Dragos, C.-A. and Preitl, S. (2013c). Simulated annealing approach to fuzzy modeling of servo systems. In *Proc. 2013 IEEE Int. Conf. Cybern. (CYBCONF 2013)*. Lausanne, Switzerland, pp. 267–272.
- Precup, R.-E., David, R.-C., Petriu, E. M., Preitl, S. and Radac, M.-B. (2014). Novel adaptive charged system search algorithm for optimal tuning of fuzzy controllers. *Expert Syst. Appl.*, 41, pp. 1168–1175.
- Preitl, S. and Precup, R.-E. (1996). On the algorithmic design of a class of control systems based on providing the symmetry of open-loop Bode plots. *Buletinul Stiintific al U.P.T. Trans. Autom. Control Comput. Sci.*, 41(55), pp. 47–55.
- Preitl, S. and Precup, R.-E. (1997). *Introducere in Conducerea Fuzzy a Proceselor*. Bucharest, Romania: Editura Tehnica.
- Preitl, S. and Precup, R.-E. (1999). An extension of tuning relations after symmetrical optimum method for PI and PID controllers. *Autom.*, 35, pp. 1731–1736.
- Preitl, S., Precup, R.-E. and Preitl, Z. (2010a). Aspects concerning the tuning of 2-DOF fuzzy controllers. In *Proc. 10th Triennial Int. SAUM Conf. Syst., Autom. Control Measurements (SAUM 2010)*. Niš, Serbia, pp. 210–219.
- Preitl, S., Precup, R.-E., Dragos, C.-A. and Radac, M.-B. (2010b). Tuning of 2-DOF fuzzy PI (D) controllers. Laboratory applications. In *Proc. 11th IEEE Int. Symp. Comput. Intell. Inf. (CINTI 2010)*. Budapest, Hungary, pp. 237–242.

- Radac, M.-B., Precup, R.-E., Preitl, S., Tar, J. K. and Burnham, K. J. (2009). Tire slip fuzzy control of a laboratory anti-lock braking system. In *Proc. Eur. Control Conf. 2009 (ECC 2009)*. Budapest, Hungary, pp. 940–945.
- Radac, M.-B., Precup, R.-E., Petriu, E. M. and Preitl, S. (2011). Application of IFT and SPSA to servo system control. *IEEE Trans. Neural Netw.*, 22, pp. 2363–2375.
- Radac, M.-B., Precup, R.-E., Petriu, E. M., Cervenak, B.-S., Dragos, C.-A. and Preitl, S. (2012). Stable iterative correlation-based tuning algorithm for servo systems. In *Proc. 38th Annu. Conf. IEEE Ind. Electron. Soc. (IECON 2012)*. Montreal, QC, Canada, pp. 2500–2505.
- Škrjanc, I., Blažič, S. and Matko, D. (2002). Direct fuzzy model-reference adaptive control. *Int. J. Intell. Syst.*, 17, pp. 943–963.
- Škrjanc, I. and Blažič, S. (2005). Predictive functional control based on fuzzy model: Design and stability study. *J. Intell. Robot. Syst.*, 43, pp. 283–299.
- Su, Z.-G., Wang, P.-H., Shen, J., Zhang, Y.-F. and Chen, L. (2012). Convenient T-S fuzzy model with enhanced performance using a novel swarm intelligent fuzzy clustering technique. *J. Process Control*, 22, pp. 108–124.
- Sugeno, M. (1999). On stability of fuzzy systems expressed by fuzzy rules with singleton consequents. *IEEE Trans. Fuzzy Syst.*, 7, pp. 201–224.
- Takagi, T. and Sugeno, M. (1985). Fuzzy identification of systems and its applications to modelling and control. *IEEE Trans. Syst., Man, Cybern.*, 15, pp. 116–132.
- Tikk, D., Johanyák, Z. C., Kovács, S. and Wong, K. W. (2011). Fuzzy rule interpolation and extrapolation techniques: Criteria and evaluation guidelines. *J. Adv. Comput. Intell. Intell. Inf.*, 15, pp. 254–263.
- Topalov, A. V., Oniz, Y., Kayacan, E. and Kaynak, O. (2011). Neuro-fuzzy control of antilock braking system using sliding mode incremental learning algorithm. *Neurocomputing*, 74, pp. 1883–1893.
- Vaščák, J. (2012). Adaptation of fuzzy cognitive maps by migration algorithms. *Kybernetes*, 41, pp. 429–443.
- Vaščák, J. and Pal'a, M. (2012). Adaptation of fuzzy cognitive maps for navigation purposes by migration algorithms. *Int. J. Artif. Intell.*, 8, pp. 20–37.
- Wang, H., Zhao, L., Du, W. and Qian, F. (2011). A hybrid method for identifying T-S fuzzy models. In *Proc. 2011 Eighth Int. Conf. Fuzzy Syst. Knowl. Discov. (FSKD 2011)*. Shanghai, China, 1, pp. 11–15.
- Wang, L., Yang, R., Pardalos, P. M., Qian, L. and Fei, M. (2013). An adaptive fuzzy controller based on harmony search and its application to power plant control. *Int. J. Electr. Power Energy Syst.*, 53, pp. 272–278.
- Yanara, T. A. and Akyürek, Z. (2011). Fuzzy model tuning using simulated annealing. *Expert Syst. Appl.*, 38, pp. 8159–8169.

Chapter 21

Genetic Optimization of Modular Neural Networks for Pattern Recognition with a Granular Approach

Patricia Melin

In this chapter, we propose a multi-objective hierarchical genetic algorithm (HGA) for modular neural network (MNN) optimization. A granular approach is used due to the fact that the dataset is divided into granules or sub modules. The main objective of this method is to know the optimal number of sub modules or granules, but also allow the optimization of the number of hidden layers, number of neurons per hidden layer, error goal and learning algorithms per module. The proposed approach is based on a Micro GA and was tested for a pattern recognition application. Simulation results show that the proposed MNN approach offers advantages over existing neural network (NN) models

21.1. Introduction

Hybrid intelligent systems are computational systems that integrate different intelligent techniques. Examples of these techniques are modular neural networks (MNN) and genetic algorithms (GAs). Hybrid intelligent systems are now being used to support complex problem solving and decision-making in a wide variety of tasks. Hybrid intelligent systems allow the representation and manipulation of different types and forms of data and knowledge, which may come from various sources. In this chapter, these techniques are combined using a granular approach. It was decided to apply the proposed method to pattern recognition to test the approach with complex problems.

There are many works that combine different techniques and they have demonstrated that the integration of different intelligent techniques provide good results, such as in Hidalgo *et al.* (2009); Melin *et al.* (2009, 2010, 2011, 2012); Mendoza *et al.* (2009a, 2009b); Sánchez and Melin (2010).

This chapter is organized as follows: [Section 21.2](#) contains the basic concepts used in this research work, [Section 21.3](#) contains the general architecture of the proposed method, [Section 21.4](#) presents experimental results and in [Section 21.5](#), the conclusions of this work are presented.

21.2. Basic Concepts

In this section, we present a brief overview of the basic concepts used in this research work.

21.2.1. Modular Neural Networks

Neural networks (NNs) can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques [24]. The MNNs are comprised of modules. The idea on which this kind of learning structure is based on the divide-and-conquer paradigm: The problem should be divided into smaller sub problems that are solved by experts (modules) and their partial solutions should be integrated to produce a final solution (Azamm, 2000; Khan *et al.*, 2009; Santos *et al.*, 2006). A module can be a sub-structure or a learning sub-procedure of the whole network (Auda and Kamel, 1999).

The results of the different applications involving MNNs lead to the general evidence that the use of MNNs implies a significant learning improvement comparatively to a single NN and especially to the back-propagation NN. Each NN works independently in its own domain. Each of the NNs is build and trained for a specific task (Melin and Castillo, 2005).

21.2.2. Multi-Objective HGA

A Genetic algorithm (GA) is an optimization and search technique based on the principles of genetics and natural selection (Haupt and Haupt, 2004; Mitchell, 1998; Segovia *et al.*, 2002). GAs are non-deterministic methods that employ crossover and mutation operators for deriving offspring. GAs work by maintaining a constantsized population of candidate solutions known as individuals (chromosomes) (Coley, 1999; Huang and Wechsler, 1999; Nawa *et al.*, 1999).

Introduced in Tang *et al.* (1998), a HGA is a type of GA is that for some complex systems, which cannot be easily represented, this type of GA can be a better choice. The complicated chromosomes may provide a good new way to solve the problem (Wang *et al.*, 2002; Worapradya and Pratishthananda, 2004).

Multi-objective optimization (MO) seeks to optimize the components of a vector-valued cost function. Unlike single objective optimization, the solution to this problem is not a single point, but a family of points known as the Pareto-optimal set. Each point in this surface is optimal in the sense that no improvement can be achieved in one cost vector component that does not lead to degradation in at least one of the remaining components (Fonseca and Fleming, 1993).

There are three general approaches to MO. The first is to combine the individual objective functions into a single composite function (Aggregating functions). The second

is to use Population-based approaches and the third is to use Pareto-based approaches. A Pareto optimal set is a set of solutions that are non-dominated with respect to each other. Pareto optimal sets can be of varied sizes, but the size of the Pareto set increases with the increase in the number of objectives (Abraham *et al.*, 2005).

In this work, the multi-objective GA is based on a Micro genetic algorithm, proposed in Coello Coello *et al.* (2007); Coello Coello and Toscano Pulido (2005). Two main characteristics of this kind of genetic algorithm are that it works with a small population and has a re-initialization process.

21.2.3. Granular Computing

Granular computing is often defined as an umbrella term to cover many theories, methodologies, techniques, and tools that make use of granules in complex problem solving. Granular computing is a new term for the problem solving paradigm and may be viewed more on a philosophical rather than technical level (Yao, 2000, 2004, 2005a, 2007).

Granular computing has begun to play important roles in bioinformatics, e-Business, security, machine learning, data mining, high-performance computing and wireless mobile computing in terms of efficiency, effectiveness, robustness and uncertainty (Bargiela and Pedrycz, 2006; Yao, 2005b; Yu and Pedrycz, 2009).

A granule may be interpreted as one of the numerous small particles forming a larger unit. The philosophy of thinking in terms of levels of granularity, and its implementation in more concrete models, would result in disciplined procedures that help to avoid errors and to save time for solving a wide range of complex problems. At least three basic properties of granules are needed: internal properties reflecting the interaction of elements inside a granule, external properties revealing its interaction with other granules and, contextual properties showing the relative existence of a granule in a particular environment (Yao, 2001).

21.3. General Architecture of the Proposed Method

The proposed method combines MNNs and fuzzy logic as response integrators. In particular, it can be used for pattern recognition. This proposed method is able to use some datasets, for example to use “N” biometric measures to identify someone and the data of each biometric measure would be divided into different numbers of sub modules. The general architecture of the proposed method is shown in [Figure 21.1](#). For joining the different responses of each biometric measure fuzzy integration is used. The proposed method also performs the optimization of the MNNs (as number of layers, goal error, number of neurons, etc.) and the different parameters of the fuzzy integrator.

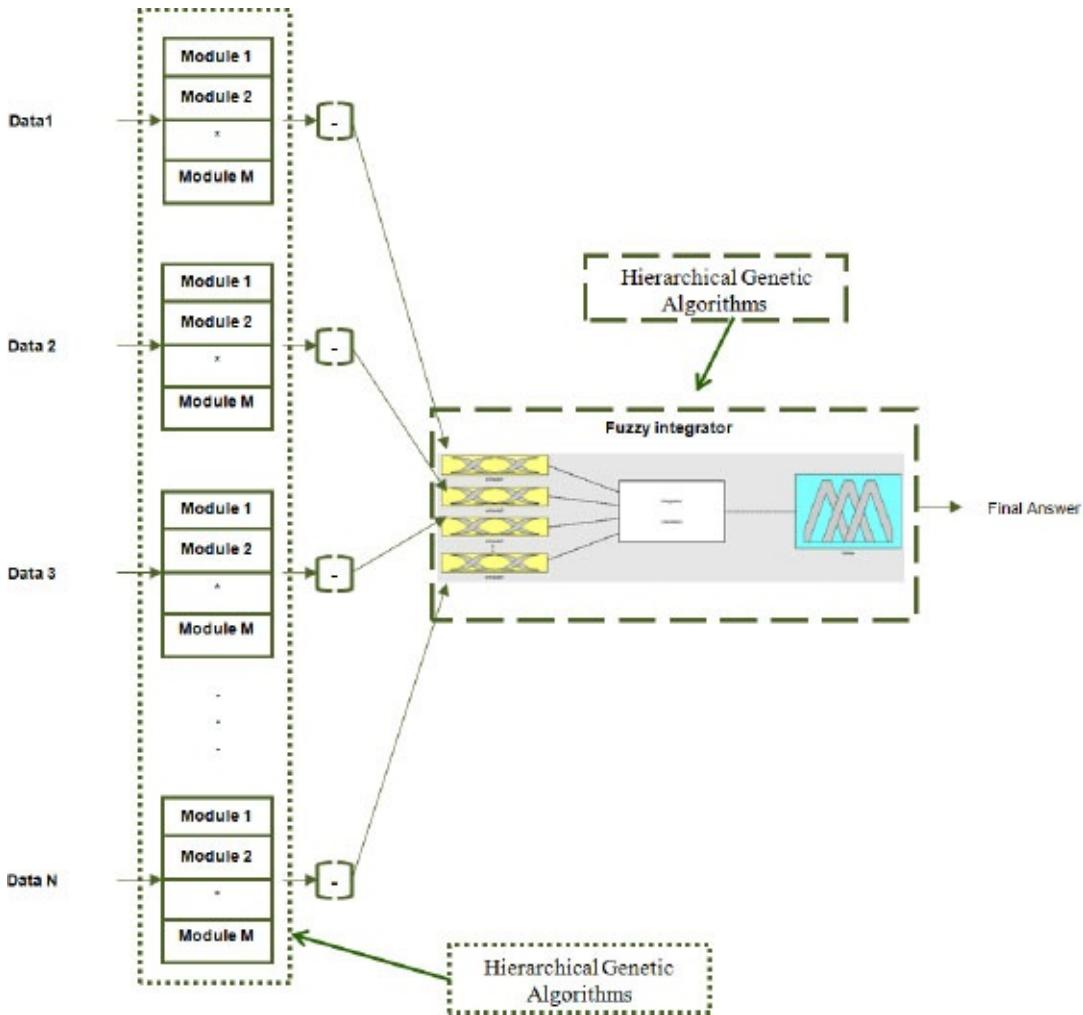


Figure 21.1: The general architecture of the proposed method.

21.3.1. Architecture of the Proposed Method for the MNN

The proposed method for MNN consists in changing the number of modules and the data per module, for example in the case of human recognition; it means that there will be different number of persons in each sub module. The number of sub modules can be established by a genetic algorithm, but at this moment the number is established randomly. The architecture of the proposed method for the modular neural network is shown in [Figure 21.2](#).

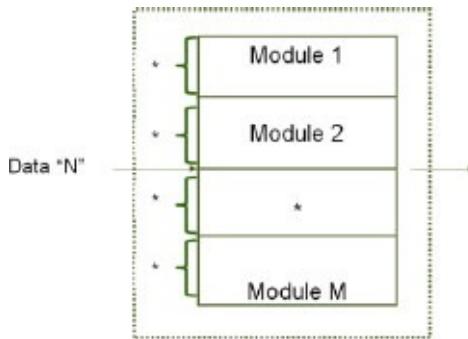


Figure 21.2: The architecture of proposed method for the MNN.

This method also chooses randomly which images will be used for training, but first the percentage of images for training is established (at this moment that percentage is defined randomly).

21.3.2. Description of the Multi-Objective HGA for MNN Optimization

With the purpose of knowing the optimal number of modules and the percentage of data for training, it is proposed the use of a genetic algorithm that allows the optimization of these parameters and others as the number of hidden layers, number of neurons per hidden layer, error goal and learning algorithms per module.

[Figure 21.3](#) shows the chromosome, which was proposed for optimization of the neural networks.

The way in which the multi-objective HGA works is illustrated and described in more detail below.

First, a random population is generated. This random population is divided in two parts: a non-replaceable and replaceable portion. The non-replaceable portion never changes during the evolution, this helps to provide diversity. The replaceable portion experiences changes after certain condition is satisfied, this condition is called nominal convergence.

The working population at the beginning is taken (with a certain probability) from both portions of the main population. During each cycle, the MOHGA uses conventional genetic operators.

The external memory is initially empty, in each cycle the non-dominated vectors found are saved in that memory, logically a comparison is performed between the new vectors found and vectors already stored.

The MOHGA has two kinds of convergence. The first is the usually used (for example when it has the maximum number of cycle or generations, or when the value desired of one objective function is obtained). The second is called Nominal Convergence, in this case is established each five generations, here two non-dominated vectors are taken of the external memory and these are compared with two vectors of the Replaceable portion, if the two vectors taken of the replaceable portion are dominated by the others, those vector

are replaceable for the two vectors of the external memory, then the working population is re-initialized.

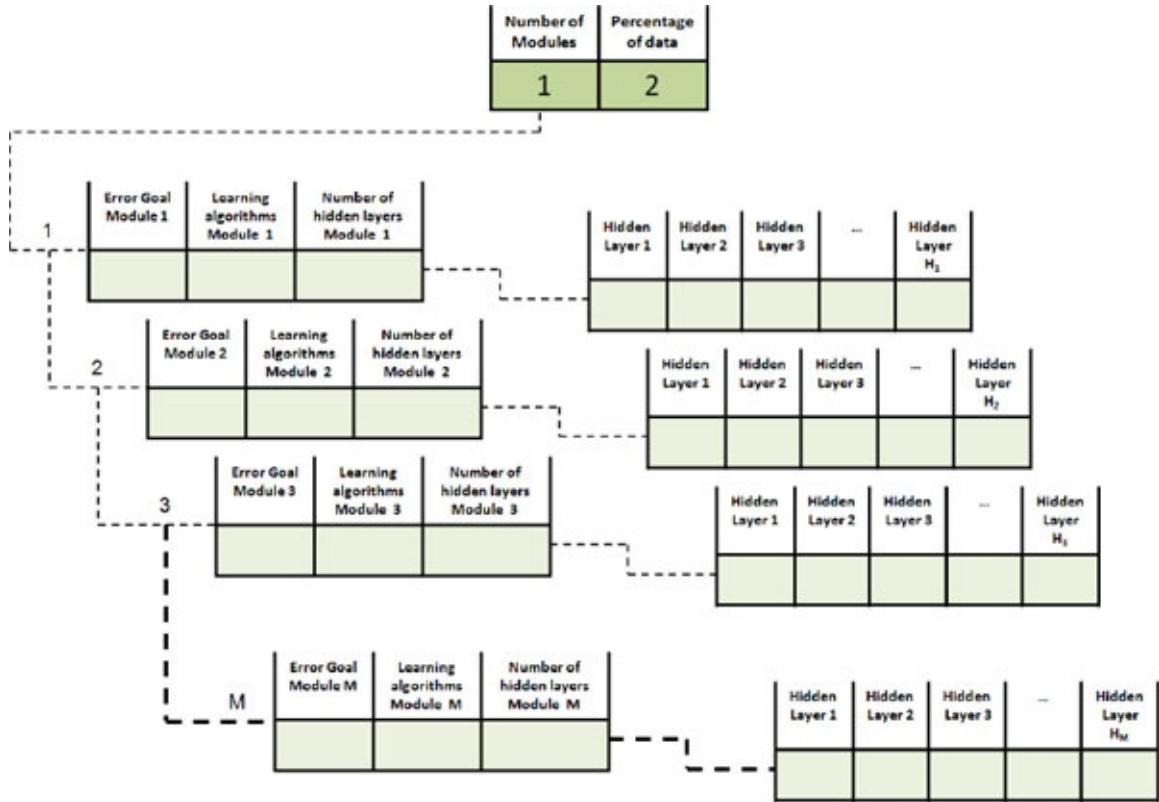


Figure 21.3: The chromosome of the multi-objective HGA for the MNN.

21.3.3. Objective Functions

In order to not only get the network that provides us with the lowest error of recognition another objective function is set, and so we not only obtain the best network with the lowest error of recognition, but also obtain a modular neural network that uses the lowest percentage of data for the training phase. The objective functions are defined below:

$$\text{Min } f_1 = \sum_{i=1}^m \left(\sum_{j=1}^{n_m} \frac{x_j}{n_m} \right), \quad (1)$$

$$\text{Min } f_2 = \text{percentage of data}. \quad (2)$$

21.3.4. Databases

The databases used in this work are described below in more detail.

21.3.4.1. Ear database

We used a database of the University of Science and Technology of Beijing (USTB, 2009). The database consists of 77 people that contain four images per person (one ear), the image dimensions are 300×400 pixels, and the format is BMP.

The persons are students and teachers from the department of Information Engineering. Two images with angle variation and one with illumination variation are used.

21.3.4.2. *Voice database*

In the case of voice, the database consists of 10 voice samples (of 77 persons), WAV format. The persons are students from the Tijuana Institute of Technology. The word that they said in Spanish was “ACCESAR”. To pre-process the voice the Mel Frequency Cepstral Coefficients were used.

21.4. Experimental Results

In this section, the results obtained in this work are presented. It was decided to use the database already described above. For the integration of responses the winner takes all method was used.

21.4.1. Non-Optimized Results

In this test the images percentage and the images, which would be used for training, were established randomly. The non-optimized results of the modular neural network are shown below.

21.4.1.1. Non-optimized results of ear

The best five results for the ear are shown in [Table 21.1](#). In this test, it can be noticed that when the number of data per module is varied the rate of recognition varies.

It can be noticed that in the training # 4, that when the images 2, 3, and 4 are used a rate of recognition of 100% is obtained.

21.4.1.2. Non-optimized results of voice

The best five results for the voice are shown in [Table 21.2](#). In this test, we can notice that when the number of data per module is varied the rate of recognition varies.

It can be noticed that in the training # 3, that when the voices 1, 3, 5, 7, 8 and 10 are used, a rate of recognition of 96.75% using eight sub modules is obtained.

21.4.2. Optimized Results

These tests make use of the multi-objective HGA, this MOHGA allows the optimization of parameters of the modular neural network, such as number of sub modules, percentage of data for training, number of hidden layers, number of neurons per hidden layer, error goal and learning algorithms per module.

Table 21.1: The best results for the ear (non-optimized).

Training	Images for training	Persons per module	Recognition rate
1	(1, 3, and 4)	Module # 1 (1 to 6) Module # 2 (7 to 14) Module # 3 (15 to 77)	67.53% (52/77)
2	(2 and 4)	Module # 1 (1 to 38) Module # 2 (39 to 70) Module # 3 (71 to 77)	77.92% (120/154)
3	(1 and 3)	Module # 1 (1 to 9) Module # 2 (10 to 44) Module # 3 (45 to 77)	83.11% (128/154)
4	(2, 3, and 4)	Module # 1 (1 to 40) Module # 2 (41 to 50) Module # 3 (51 to 77)	100% (77/77)
5	(2 and 3)	Module # 1 (1 to 23) Module # 2 (24 to 47) Module # 3 (48 to 77)	93.50% (144/154)

Table 21.2: The best results for voice (non-optimized).

Training	Images for training	Persons per module	Recognition rate
1	53% (1, 3, 6, 7, and 9)	Module # 1 (1 to 22) Module # 2 (23 to 57) Module # 3 (58 to 77)	278/385 72.20%
2	48% (1, 2, 5, 6, and 7)	Module # 1 (1 to 39) Module # 2 (40 to 68) Module # 3 (69 to 77)	260/385 67.53%
3	35% (2, 5, 8, and 9)	Module # 1 (1 to 36) Module # 2 (37 to 68) Module # 3 (69 to 77)	401/462 86.79%
4	46% (3, 5, 6, 7, and 10)	Module # 1 (1 to 40) Module # 2 (41 to 67) Module # 3 (68 to 77)	347/385 90.12%
5	59% (1, 3, 5, 7, 8, and 10)	Module # 1 (1 to 7) Module # 2 (8 to 39) Module # 3 (40 to 77)	298/308 96.75%

Table 21.3: Main parameters of the MOHGA.

Memory size	Non-replaceable memory	Replaceable memory	Working memory	Pareto optimal	Duration
50	25	25	5	7	12:48:08

21.4.2.1. Optimized results of the ear

The main parameters used in this evolution are shown in [Table 21.3](#) and the Pareto optimal set found for the ear are shown in [Figure 21.4](#).

The solutions found in the Pareto optimal set are shown in [Table 21.4](#), and the best architecture is shown in [Table 21.5](#).

The solutions that have a recognition rate greater than 97% are taken, and of the resulting set, the solution with lower percentage of data is the best for us.

21.4.2.2. Optimized results of the voice

The main parameters used in this evolution are shown in [Table 21.6](#).

The solutions found in the Pareto optimal set are shown in [Table 21.7](#), and the best architecture is shown in [Table 21.8](#).

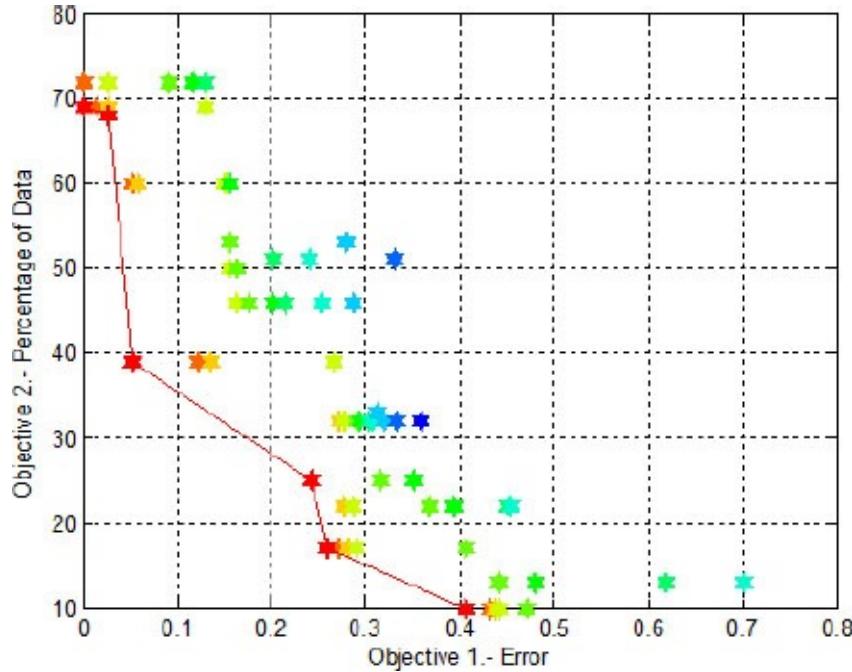


Figure 21.4: Pareto optimal set for the evolution of the ear.

Table 21.4: The best results for the ear (Pareto optimal set).

Solution	No. of modules	% of data	Total rec. (%)	Error
1	5	69	100	0
2	6	68	97.40	0.0260
3	5	39	94.80	0.0519
4	5	25	75.75	0.2424
5	9	17	74.02	0.2597
6	9	17	74.02	0.2597
7	5	10	59.30	0.4069

Table 21.5: The best result of the ear (optimized).

No. of mod.	% and images	No. hidden layers and num. of neurons	Persons per module	Rec. rate	Error
5	69% (2, 3, and 4)	3(173,135,44) 2(153,120) 4(72,184,96,116) 2(197,166) 3(164,22,94)	Module # 1 (1 to 6) Module # 2 (7 to 13) Module # 3 (14 to 27) Module # 4 (28 to 53) Module # 5(54 to 77)	77/77	0

Table 21.6: Main parameters of the MOHGA.

Memory size	Non-replaceable memory	Replaceable memory	Working memory	Pareto optimal	Duration
50	25	25	5	9	01:51:12

Table 21.7: The best results for voice (Pareto optimal).

Solution	No. of modules	% of data	Total rec. (%)	Error
1	5	79	98.05	0.0195
2	9	49	97.40	0.0260
3	9	44	96.96	0.0303
4	5	38	95.88	0.0411
5	10	19	89.44	0.1055
6	10	19	89.44	0.1055
7	7	17	83.76	0.1623
8	7	7	74.45	0.2554
9	6	4	73.73	0.2626

Table 21.8: The best result of the voice (optimized).

No. of mod.	% and voices	No. hidden layers and No. of neurons	Persons per module	Rec. rate	Error
9	49% (1, 3, 8, 9 and 10)	4 (57,144,128,83) 4 (156,189,158,193) 5(123,105,169,110,105) 1(89) 3(78,143,62) 2(101,38) 4(22,60,91,173) 4(81,128,139,118) 4(145,28,187,32)	Module # 1(1 to 14) Module # 2(15 to 35) Module # 3(36 to 46) Module # 4(47 to 50) Module # 5(51 to 53) Module # 6(54 to 55) Module # 7(56 to 64) Module # 8(65 to 72) Module # 9(73 to 77)	375/385 97.40%	0.0260

The solutions that have a recognition rate greater than 97% are taken, and of the resulting set, the solution with lower percentage of data is the best for us.

21.5. Conclusions

A new method for designing modular neural networks with a granular approach was proposed. The main goal of this work was providing the modular neural networks with the following characteristics: allow changing the number of modules, data per module, and percentage of data for training, all of that with the goal of obtaining a better rate of recognition.

A multi-objective HGA was developed for optimization of some parameters of the model of modular neural networks, these parameters are the number of modules, percentage of data for training, goal error per module, number of hidden layers per module and their respective neurons. This MOHGA is able to obtain the best modular neural network with the lowest error of recognition and that uses the lowest percentage of data for the training phase.

In this work when the tests with the ear are compared, a significant difference does not exist, because the database has few images per person, but if we compare the non-optimized versus the optimized test in the case of the voice, we can notice that with less data a good recognition rate is obtained.

References

- Abraham, A., Jain, L. and Goldberg R. (2005). *Evolutionary Multiobjective Optimization, First Edition*. Heidelberg, Germany: Springer.
- Auda, G. and Kamel, M. S. (1999). Modular neural networks, a survey. *Int. J. Neural Syst.*, 9(2), pp. 129–151.
- Azamm, F. (2000). *Biologically Inspired Modular Neural Networks*. Ph.D. thesis. Blacksburg, Virginia: Virginia Polytechnic Institute and State University.
- Bargiela, A. and Pedrycz, W. (2006). The roots of granular computing. *IEEE Int. Conf. Granular Comput. (GrC)*, 2006, San Jose, California, USA, pp. 806–809.
- Coello Coello, C. A., Lamont, G. B. and van Veldhuizen, D. A. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems, Second Edition*. Heidelberg, Germany: Springer.
- Coello Coello, C. A. and Toscano Pulido, G. (2001). A micro-genetic algorithm for multi-objective optimization. In Coello, C. A. and Corne, D. W. (eds.), *Evolutionary Multi-objective Optimization 2001. Lecture Notes in Computer Science*, 1993, pp. 126–140.
- Coley, A. (1999). *An Introduction to Genetic Algorithms for Scientists and Engineers, Har/Dskt Edition*. Singapore: WSPC.
- Database Ear Recognition Laboratory from the University of Science & Technology Beijing (USTB). Found on the Web page: <http://www.ustb.edu.cn/resb/en/index.htm> asp (Accessed 21 September 2009).
- Fonseca, C. M. and Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. *Int. Conf. Genetic Algorithms (ICGA 1993)*, San Mateo, USA, pp. 416–423.
- Haupt, R. and Haupt, S. (2004). *Practical Genetic Algorithms, Second Edition*. New York: Wiley-Interscience, pp. 42–43.
- Hidalgo, D., Castillo, O. and Melin, P. (2009). Type-1 and type-2 fuzzy inference systems as integration methods in modular neural networks for multimodal biometry and its optimization with genetic algorithms. *Inf. Sci.*, 179(13), pp. 2123–2145.
- Huang, J. and Wechsler, H. (1999). Eye location using genetic algorithm. *Second Int. Conf. Audio Video-Based Biometric Person Authentic.*, 1999, Corfet, Greece, pp. 130–135.
- Khan, A., Bandopadhyaya, T. and Sharma, S. (2009). Classification of stocks using self-organizing map. *Int. J. Soft Comput. Appl.*, 4, pp. 19–24.
- Melin, P. and Castillo, O. (2005). *Hybrid Intelligent Systems for Pattern Recognition Using Soft Computing: An Evolutionary Approach for Neural Networks and Fuzzy Systems, First Edition*. Heidelberg, Germany: Springer, pp. 119–122.
- Melin, P., Kacprzyk, J. and Pedrycz, W. (2009). *Bio-Inspired Hybrid Intelligent Systems for Image Analysis and Pattern Recognition*. Heidelberg, Germany: Springer.
- Melin, P., Mendoza, O. and Castillo, O. (2010). An improved method for edge detection based on interval type-2 fuzzy logic. *Expert Syst. Appl.*, 37(12), pp. 8527–8535.
- Melin, P., Mendoza, O. and Castillo, O. (2011). Face recognition with an improved interval type-2 fuzzy logic Sugeno integral and modular neural networks. *IEEE Trans. Syst., Man, Cybern., Part A*, 41(5), pp. 1001–1012.
- Melin, P., Sánchez, D. and Castillo, O. (2012). Genetic optimization of modular neural networks with fuzzy response integration for human recognition. *Inf. Sci.*, 197, pp. 1–19.
- Mendoza, O., Melin, P. and Castillo, O. (2009a). Interval type-2 fuzzy logic and modular neural networks for face recognition applications. *Appl. Soft Comput.*, 9(4), pp. 1377–1387.
- Mendoza, O., Melin, P. and Licea, G. (2009b). A hybrid approach for image recognition combining type-2 fuzzy logic, modular neural networks and the Sugeno integral. *Inf. Sci.*, 179(13), pp. 2078–2101.
- Mitchell, M. (1998). *An Introduction to Genetic Algorithms, Third Edition*. Colorado: Bradford.
- Nawa, N., Takeshi, F., Hashiyama, T. and Uchikawa, Y. (1999). A study on the discovery of relevant fuzzy rules using pseudobacterial genetic algorithm. *IEEE Trans. Ind. Electron.*, 46(6), pp. 1080–1089.

- Sánchez, D. and Melin, P. (2010). Modular neural network with fuzzy integration and its optimization using genetic algorithms for human recognition based on iris, ear and voice biometrics. In Patricia, M., Janusz, K. and Witold, P. (eds.), *Soft Computing for Recognition Based on Biometrics Studies in Computational Intelligence, First Edition*. Heidelberg, Germany: Springer, pp. 85–102.
- Santos, J. M., Alexandre, L. A. and Marques de Sá, J. (2006). Modular neural network task decomposition via entropic clustering. *Sixth Int. Conf. Intell. Sys. Design Appl., ISDA* (1) 2006, Jinan, China, pp. 1067–1072.
- Segovia, J., Szczepaniak, P. S. and Niedzwiedzinski, M. (2002). *E-Commerce and Intelligent Methods, First Edition*. Heidelberg, Germany: Physica-Verlag, p. 181.
- Tang, K. S., Man, K. F., Kwong, S. and Liu, Z. F. (1998). Minimal fuzzy memberships and rule using hierarchical genetic algorithms. *IEEE Trans. Ind. Electron.*, 45(1), pp. 162–169.
- Wang, C., Soh, Y. C., Wang, H. and Wang, H. (2002). A hierarchical genetic algorithm for path planning in a static environment with obstacles. *Can. Conf. Electr. Comput. Eng. 2002, IEEE CCECE 2002*, 3, Niagara Falls, Canada, pp. 1652–1657.
- Worapradya, K. and Pratisthananda, S. (2004). Fuzzy supervisory PI controller using hierarchical genetic algorithms. *Fifth Asian Control Conf., 2004*, 3, Taipei, Taiwan, pp. 1523–1528.
- Yao, J. T. (2005a). Information granulation and granular relationships. *Proc. 2005 IEEE Conf. Granular Comput. (GrC)*, San Jose, California, USA, pp. 326–329.
- Yao, J. T. (2007). A ten-year review of granular computing. *Proc. Third IEEE Int. Conf. Granular Comput. (GrC)*, 2007, San Jose, California, USA, pp. 734–739.
- Yao, Y. Y. (2000). Granular computing: basic issues and possible solutions. *Proc. Fifth Joint Conf. Inf. Sci.*, 2000, pp. 186–189.
- Yao, Y. Y. (2001). On modeling data mining with granular computing. *25th Int. Comput. Softw. Appl. Conf. (COMPSAC 2001)*, Taipei, Taiwan, pp. 638–649.
- Yao, Y. Y. (2004). A partition model of granular computing. *Lect. Notes Comput. Sci.*, 3100, Heidelberg, Germany, pp. 232–253.
- Yao, Y. Y. (2005b). Perspectives of granular computing. *IEEE Int. Conf. Granular Comput. (GrC) 2005*, San Jose, California, USA, pp. 85–90.
- Yu, F. and Pedrycz, W. (2009). The design of fuzzy information granules: Tradeoffs between specificity and experimental evidence. *Appl. Soft Comput.*, 9(1), pp. 264–273.

Chapter 22

Hybrid Evolutionary-, Constructive- and Evolving Fuzzy Neural Networks

Michael J. Watts and Nikola Kasabov

The term “evolving” in its most general form means “changing through time”, “developing”, or “unfolding”. In this chapter, we will describe several ways in which this concept has been applied to artificial neural networks (ANNs), i.e., evolutionary, constructive, evolving, and their hybrid combinations. The chapter presents a review of various methods, systems and their applications developed in the last 20 or so years and also points to new directions for the development of evolving ANN (EANN). It makes a clear distinction between evolutionary ANN, where the whole ANN is optimized with the use of heuristic driven evolutionary algorithms (EA) that are population, generation based, and the constructive EANN which develop their structure and functionality using incremental learning methods, even though they may use EA for some parameter optimization if necessary. A type of constructive ANN, called evolving connectionist system (ECoS) is presented in more detail in another chapter of the handbook.

22.1. Introduction

There are several well-known problems associated with conventional artificial neural network (ANN, also called connectionist systems). Difficulties such as selecting the topology of a network, the training algorithm and parameters used, the amount of knowledge an ANN can store, and their limited adaptability over time, all place constraints on the real world performance of connectionist structures.

The topology of an ANN will have a distinct effect upon its quality. The input features used are of critical importance. Including redundant or irrelevant features of the input space will cause the network to learn very slowly, if at all, and will require a larger network in order to handle the unnecessarily complex input space. Conversely, omitting important features will make learning in the network almost impossible. Statistical analysis can help make these decisions, but has a problem handling unending (continuous) data streams.

The number of hidden nodes in a feed-forward network is of great concern to an ANN practitioner. Although it has been established that a feed-forward ANN with sufficient hidden neurons can approximate any continuous function (Kolmogorov, 1957; Cybenko, 1989), the number of neurons needed is unknown. Again, it is a balancing act between having too few, which will make learning impossible due to a lack of connections within which to store knowledge, and having too many, which makes over-learning almost inevitable. The final architectural problem with ANN is the connectivity of the neurons. Some connections after training will probably be less relevant than others: their presence may cause confusion and inefficiency in the network.

Selection of the training parameters is a multi-parameter optimization problem: too low a learning rate means that the network will train slowly, but with less chance of becoming stuck in a local minimum. Too high a learning rate will speed the training process, but increases the chances of it being stuck in a local minimum. An additional danger is that a high learning rate will lead to over-training, or over-fitting, on the training data. The momentum parameter of the algorithm must also be chosen with care: if the momentum is too low, then it will not serve its purpose of smoothing the network's trajectory across the error surface. If it is too high, then the network may overshoot the global minimum. Some work has also been done on dynamically calculating the training parameters as learning is underway (Moreira and Fiesler, 1995; Schiffmann *et al.*, 1994).

ANN are limited in dynamic adaptation. Catastrophic forgetting is the term used to describe what happens when an ANN previously trained on one dataset is further trained on another (McCloskey and Cohen, 1989). Although the ANN is usually able to learn the new dataset quite adequately, it is likely to forget the previous dataset. This causes problems when the ANN must be used in a so-called “online learning” application, which requires new data to be continuously accommodated: the new data will be handled, but the old data will be forgotten.

Although there are now many variants of back-propagation training in existence, they all suffer from a lack of training speed (Fahlman, 1988). Back-propagation requires the repeated presentation of the entire training set, usually hundreds or thousands of times before an acceptable result is obtained. Although some back-propagation derived algorithms are usually much faster (e.g., quickprop (Fahlman, 1988)), these still require the repetitive presentation of the training set.

Most of these problems are search problems: we must search for the near-optimal topology of an ANN; we must search for the near-optimal parameters when training the ANN. There are, therefore, three main approaches to optimizing the structure of ANN:

- (a) Using evolutionary algorithms (EAs) for the optimization of the whole ANN; we call these evolutionary ANN ([Section 22.2](#)).
- (b) Using a constructive algorithm. We call these constructive ANN ([Section 22.3](#)).
- (c) Using improved constructive algorithms for the ANN to self-organize and adapt its structure, functionality and to accumulate new knowledge across multiple datasets and problems. This is the case of evolving connectionist systems (ECoS, [Section 22.4](#)). As a partial case ECoS can use EA for the optimization of some of their parameters if necessary, but the main emphasis is on the learning and adaptation methods that are brain-like methods.

In this chapter, we assume that the reader is already familiar with the basic principles of ANN and gradient-descent based ANN training. We also assume that the reader is familiar with the principles of EA, especially genetic algorithms (GA), evolutionary programming (EP) and evolution strategies (ES).

22.2. Evolutionary ANN

EA are search algorithms inspired by natural selection and biological evolution. Attempts to solve the search problems in ANN using the search abilities of EA have brought the two areas of ANN and EA together with pioneering work by D. Fogel and later by X. Yao.

While methods of combining EA and ANN continue to be an active area of research, most of the work in this field has been directed towards evolving the architecture of the ANN, its parameters, and its connection weights.

The type of EA used varies according to the application. GA are very widely used, but this raises issues with representation of the ANN as a chromosome, and in how the chromosomes (genotype) are mapped to the final ANN (the phenotype). EP has also been used, especially when evolving the connection weights of the ANN. This avoids the problems with mapping from genotype to phenotype, as the mutation evolutionary operator in EP works directly on the phenotype. This does, however, require finding a suitable method of mutation for the problem at hand.

Whichever EA is used, the fitness of each individual ANN within a population must be evaluated. This is usually derived from some measure of the performance of the ANN, which is often measured as the accuracy over a particular dataset, or how well it solves a particular problem. Care must be taken when selecting the fitness measure, as the EA may become stuck in an evolutionary dead-end if the fitness measure is not carefully designed to avoid it. An incautiously designed fitness function can also cause the EA to over-optimize the ANN on one aspect of the problem at the expense of other aspects.

22.2.1. Evolving the Architecture of ANN

As discussed above, the topology of an ANN has an enormous effect on its performance. Many researchers have investigated using EA for solving this problem. There are two general ways in which ANN topology can be evolved: directly, where the structure and connectivity of the ANN are encoded directly into the EA; and indirectly, where the EA evolves parameters of the ANN structure, which are then processed to create the final ANN. The fitness of the individuals is taken as some measure of the performance of the ANN, usually some measure of the error over a training dataset.

The most straight-forward approach used is to evolve through an EA the number of neurons in each layer. In Arena *et al.* (1993), each chromosome in the genetic algorithm was treated as a two-dimensional matrix, with each cell row representing the presence or absence of one node of the network. During evaluation, the number of nodes present in each layer was counted and an appropriate multilayer perceptron (MLP) architecture created. If no nodes were present in a row, then that layer was not included in the phenotype. Crossover was implemented as the exchange of a sub matrix from each parent. This method was tested on the optimization of a complex nonlinear system and was shown

to be effective at determining a near optimal topology for the MLP. This was a direct encoding approach.

A similar two-dimensional chromosome approach can also be used, where the connectivity of the neurons is being evolved. The values of the alleles represented the presence or absence of a connection between neurons. Neurons that had no connections were removed from the ANN.

A different approach was taken in Schiffman *et al.* (1993). Here both of the previous concepts were combined, and the length of the chromosome determined the number of nodes present, as well as the connectivity of the nodes. This approach to ANN design was tested on a medical classification problem, that of identifying thyroid disorders, and provided networks that were both smaller and more accurate than manually designed ANNs. These examples were again direct encoding schemas.

A simpler, but indirect, approach was used in Watts and Kasabov (1998) to optimize the structure of a Fuzzy Neural Network (FuNN, Kasabov *et al.*, 1997). FuNN have fuzzy membership functions attached to each input and output neuron, as well as hidden neurons that represent fuzzy rules. Watts and Kasabov used a simple GA to evolve the number of fuzzy MF for each input and output, as well as the number of hidden neurons. They also directly selected which input variables were used, that is, the GA simultaneously evolved the optimal feature set as well as the optimal FuNN structure.

Note, however, that all the above methods optimize an ANN, after which the ANN becomes a fixed-topology ANN, which is a significant limit for the dynamic adaptation of the ANN. That is, the EA optimizes the ANN to the dataset used in the evaluation function, it does not allow it to adapt to any other datasets.

22.2.2. Evolving the Training Parameters of ANN

In addition to the selection of an ANN's topology, it is also possible to use EA to select the training parameters for the network. In these applications the EA evolves the learning parameters of the ANN, and the fitness criterion is the performance of the ANN after training.

In Choi and Bluff (1995) a GA was used to select the training parameters for the back-propagation training of an MLP. In this work, the chromosome encoded the learning rate, momentum, sigmoid parameter and number of training epochs to be used for the back-propagation training of the network. This technique was tested with several different datasets, including bottle classification data, where a glass bottle is classified as either being suitable for reuse or suitable for recycling and breast cancer data, which classifies tissue samples as malignant or benign. For each of the test datasets, the genetically derived network outperformed those networks whose control parameters were manually set, often by a significant margin. However, this approach still suffers from the inherent limitations

of back-propagation training.

22.2.3. *Evolving the Connection Weights of ANN*

While gradient-descent methods such as back-propagation are powerful and useful for many applications, there are certain situations in which they will fail. As error-driven learning algorithms, back-propagation requires an error function that is continuous and differentiable, yet there are many real-world applications in which this is not the case. Back-propagation is also a local learning algorithm, that is, the change in connection weights moves the network from one point of the problem space to another nearby point. This means that the ANN can easily become trapped in local minima and fail to find the global optimum. Thus, for some problems EA training may be the only way in which to guarantee convergence and it becomes more efficient to train the ANN via EA. There are also some esoteric architectures of ANN (see below) that can only be feasibly trained via a search algorithm such as EA.

When evolving the connection weights of ANN, each individual in the EA population represents one set of connection weights. During the evaluation phase of the EA each individual is decoded into an ANN and the performance of the ANN is evaluated. Exactly how the ANN is evaluated depends entirely on the application being attempted. The simplest approach is to propagate an input dataset through the candidate ANN, and compare its outputs with the target outputs, that is, compute an error value, where the fitness of the individual is inversely proportional to the error. In these applications the EA is serving as a supervised learning algorithm. This, however, requires a set of known output values which might not be available or even be appropriate for the problem. The advantages of avoiding local minima, however, remain. An example of this approach was breast cancer detection from mammograms (Fogel *et al.*, 1997). Here, EP was used to evolve the connection weights of a MLP whose inputs were certain features of a mammogram. The evaluation function was a simple count of the number of examples that were correctly classified as cancer or not-cancer. While backpropagation was not successful at solving this problem, EP was able to produce a network with greater than 98% accuracy. Other examples of this approach are detecting sea mines from sonar returns (Porto *et al.*, 2005) and screening compounds for their anti-HIV activity (Ma *et al.*, 2006). In each of these examples, the EP algorithm works directly on the MLP, as opposed to a string of values as with GA. Changes are evolved to the connection weights. The changes to the weights are normally distributed around a mean of zero and selection was via a tournament process. The winner of each tournament was the MLP that had the highest evaluation, which was usually the lowest mean-squared error.

An interesting application of EP to MLP was evaluating potential future moves on a checkers board (Chellapilla and Fogel, 2001). The input to the MP was an encoded representation of the positions of pieces on a checkers board, and the output was an

evaluation of the “goodness” of the pieces’ distributions, that is, an evaluation of how advantageous the positions were to the player. The evaluation function used for this application was quite interesting. Since it was not feasible to evaluate the mean-squared error for this problem (such a measure would anyway be quite meaningless for this application) the candidate MLP played several games of checkers against other MLP, and the software chose the move that gave the best future board configuration. The games themselves were run using a simple checkers engine which would search through possible moves and future board configurations based on the current positions of pieces. The candidate MLP assessed the value of each configuration. MLP which assessed the values of configurations better won more games in the tournaments and were used as the basis of the next generation. MLP which were not so good lost more games and were eliminated. This approach worked well enough that the system was able to earn a “Master” rating on an online checkers system. A similar approach was later adapted to playing chess (Fogel *et al.*, 2004) where the system again achieved a “Master” rating.

An early example of GA-based training of non-MLP ANN was de Garis (1990, 1992). The networks used in these experiments were fully self-connected, synchronous networks. These networks are basically impossible to train via gradient descent and cannot learn using any approach other than EA. These “GenNets” were used to attempt tasks that were time dependent, such as controlling the walking action of a pair of stick legs. With this problem the inputs to the network were the current angle and angular velocity of the “hip” and “knee” joints of each leg. The outputs were the future angular acceleration of each of the joints. Both the inputs and outputs for this problem were time dependent. Conventional training methods proved to be incapable of solving this problem, while GenNets solved it very easily. Other applications of GenNets involved creating “production rule” GenNets, which duplicated the function of a production rule system. These were then inserted into a simulated artificial insect and used to process inputs from sensor GenNets. The outputs of the production rule GenNets sent signals to other GenNets to execute various actions, i.e., eat, flee, mate etc.

A similarly structured recurrent network was used in Fukuda *et al.* (1997), to attempt a similar problem. The application area in this research was using the genetically trained network to control a physical biped robot. Not only was the robot able to walk along flat and sloped surfaces, it was able to generalize its behavior to deal with surfaces it had not encountered in training. Comparison of the results gained from the genetically trained network to those trained by other methods showed that not only did the genetically trained network train more efficiently than the others; it was also able to perform much better than the others.

22.2.4. Problems with Evolutionary ANN

There are several problems in training ANN with EA. Firstly, the way in which connection

weight values are encoded in GA chromosomes is an issue. Most GA use binary encoding, but to accurately represent connection weights to a high precision requires a large number of bits per connection weight. Too many bits per weight leads to very long chromosomes, which are inefficient, while too few bits means that there will not be sufficient precision to find the optimal weight values.

While an EA can solve the problem of selecting the training parameters of an ANN, the problem of selecting the parameters for the EA remain. While EA are generally more robust to parameter selection than ANN, incorrect selection of parameters such as population size can mean that the EA takes too long to run (for too many individuals) or does not have enough genetic diversity to solve the problem (for too few individuals).

A substantial problem with using EA to optimize ANN is that it is slow. No matter what is being optimized, every candidate ANN must be evaluated. This can involve running backpropagation training over a complete training dataset, or putting each ANN through a tournament with several other ANN. Since there are usually many thousands of ANN evaluated in a single run of an EA, this adds up to a substantial amount of time. Even today, with the fast processors available, the amount of time taken to optimize an ANN using an EA can be prohibitive for large datasets.

Another significant problem with evolving the connection weights of ANN using GA is the so-called permutation problem (Hancock, 1992). Assume we have an ANN with three hidden neurons, where each of the hidden neurons is fully connected to each of the input neurons, and each connection has a unique weight value. The order in which these hidden neurons appear in the ANN does not affect the performance of the network at all; in fact every permutation of the neurons yields exactly the same output values. Yet the order in which the connection weights appear in an artificial chromosome does matter, as they will be treated as completely unique individuals by the GA. This makes crossover operations very inefficient at finding new solutions.

Finally, while EA can solve problems with topology and parameter section, and can select connection weights, they cannot solve the problem of limited dynamic adaptation. This is when further data related to the problem appears and cannot be learned by the ANN without forgetting what it has previously learned. This is a fundamental limit of fixed-topology ANN. This limit is the motivation for constructive neural networks, which are described in the next section.

22.3. Constructive ANN

The problems with evolutionary ANN described thus far all spring from the fixed nature of traditional algorithms: once architecture has been selected for the ANN, it is not possible or not easy to alter it. This means that additional knowledge cannot be accommodated, as is needed in life-long learning applications, due to the constant number of connections within which the knowledge of the ANN is stored.

If the topology of the network can be determined automatically, then the problem of adaptation in the evolutionary ANN becomes less significant. This is the motivation behind so-called constructive ANN, where the network is constructed by the training algorithm, during training. Instead of starting with an ANN of fixed topology, a constructive algorithm starts with a minimal network and progressively adds neurons and connections as needed by the training algorithm. This is the other meaning of “evolving”: while constructive algorithms do not follow the principles of EA, they are evolving as the ANN changes and adapts through time. That is, the network “evolves” by altering its structure as data is presented to it.

This helps with the topology determination process, as most constructive algorithms will stop adding neurons when such additions no longer improve performance. The learning algorithms applied to constructive networks are called constructive learning algorithms.

Constructive algorithms began as an extension of the backpropagation of errors training algorithms operating on the well-known MLP. One of the earliest constructive algorithms proposed, was the Dynamic Node Creation algorithm (Ash, 1989). As constructive algorithms matured, they started to move away from the backpropagation algorithm in favor of other training methods, and started to modify the MLP architecture to a greater degree than just expanding its hidden layer. Two examples of this kind of algorithm are the Tiling algorithm (Mézard and Nadal, 1989) and the Upstart algorithm (Frean, 1990). The Cascade Correlation algorithm (Fahlman and Lebiere, 1990) abandoned the MLP structure completely, but still used a form of backpropagation to set the connection weights. The final group of constructive algorithms is described here and is quite different from their predecessors: These are the Resource Allocating Network (RAN; Platt, 1991); the evolutionary nearest-neighbor MLP (NN-MLP; Zhao and Higuchi, 1996); and the grow and learn (GAL) network (Alpaydin, 1994). They generally have little in common with the architectures of previous ANN algorithms, and use a method of training specific to their architecture.

22.3.1. *Resource Allocating Networks*

The RAN was proposed in Platt (1991) as a way of quickly loading knowledge into a network without over-training. It is a two neuron layer network, where the first layer represents input space regions and the second layer represents the output function being

modeled. The first layer neurons use Gaussian functions. The parameters of each Gaussian function represent a specific region of the input space: each neuron in that layer calculates its activation based on how close its region is to the presented input vector, and will thus respond most strongly to inputs from that region. The output layer performs a weighted sum over the outputs of the input layer neurons, and adds a constant vector that is independent of the input vector. [Figure 22.1](#) shows a RAN network.

A RAN learns via two different mechanisms: the first adds Gaussian units according to the novelty of the input examples presented. The second modifies the connection weights according to the error of the network. The network begins with no stored patterns. As training examples are presented, both the distance between the existing Gaussian units and the input vector, and the difference between the actual and desired output vectors, are measured. If they are above a certain threshold, a new node is added. The Gaussian parameters of the new node are set so that its maximum response is for the current, novel example, with its width being proportional to its distance from the nearest existing node. The connections to the output layer are set to the difference between the network output and the desired output. If a new unit is not added, then the existing connections to the output layer are modified via a simple gradient descent algorithm so that the error of the network is reduced. As training progresses, the threshold values decay towards a set value: this means that the knowledge stored in the network will become more and more finely represented.

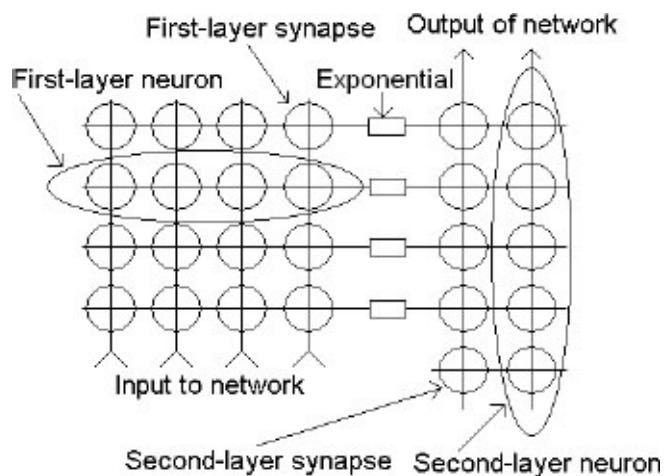


Figure 22.1: A RAN.

22.3.2. Nearest Neighbor MLP

The NN-MLP is a connectionist implementation of the nearest-neighbor classifier algorithm (Gates, 1972). In this architecture, each hidden layer neuron corresponds to a training example, with the incoming connection weight vector of the neuron being equal to the example vector it represents. Activation of the hidden neurons is calculated in the following manner: the distance between the incoming weight vector and the current example vector is calculated, and this measure is subtracted from a counter value embedded in each hidden neuron. When the value of this counter reaches zero, the neuron

fires, outputting a value of one. When a neuron fires, it inhibits all other neurons that belong to the same class from firing. An output neuron will fire when any of its inputs is one. The simplest way of constructing a NN-MLP is to create one hidden neuron for each example in the training set. This, however, will lead to a very large hidden layer, with all of the problems of efficiency and generalization accuracy that entails.

A means of constructing parsimonious NN-MLP was proposed in Zhao and Higuchi (1996). This algorithm is designed to find the minimum number of hidden neurons needed to accurately model the training dataset. The algorithm is called the *R4-Rule*, for the four separate procedures employed in it: Recognition, Remembrance, Reduction, and Review. Recognition is the process by which training examples are matched to hidden neurons. Of interest in this phase are two parameters associated with each neuron, the class label of the neuron, denoted as λ , and its importance, denoted as ρ . A hidden neuron n will fire if all three of the following conditions are true:

- It belongs to the same class as the training example (that is, $\lambda_n = \lambda_x$).
- The incoming connection weight vector of n is closer to x than any others.
- The ρ value is the largest. The values of λ_n and ρ_n are set automatically during training: whenever n is a winner, the value of ρ_n is increased. If it is a loser (it matches the first two conditions given above but has a lower ρ value) then ρ_n is decreased.

Remembrance involves evaluating the performance of the network across the dataset. If a large number of recognition errors occur, then new hidden neurons will be added to the network. As opposed to adding a single hidden neuron for each misclassified example, only one hidden neuron is added for each misclassified class. This reduces the number of neurons added to the network. If the recognition rate is high, then hidden neurons can be removed.

Reduction prunes neurons from the network when their values get very low. Only a single neuron is removed every time the reduction procedure is carried out. This is to stabilize the learning process. If the removal of the neuron causes classification errors then the reduction process is undone.

Review adjusts the connection weights of the network, and is carried out whenever a neuron is added or deleted. It is a supervised learning process, and adjusts the connection weights of the network to be closer to x .

These four steps and the relationships between them are shown in [Figure 22.2](#).

22.3.3. *GAL Networks*

GAL networks were proposed by Alpaydin (1994). They are three neuron layer binary classification networks that expand their structure as required to accommodate new training examples. The first neuron layer is the input layer: initially, this is the only layer

that exists in the network. The second layer of neurons is the exemplar layer, which represents the memorized training examples. The third layer is called the class layer. The neurons in this layer represent the classes that the network is learning to distinguish. A vector is propagated through the input layer to each neuron in the exemplar layer. There, the distance between the input vector and the incoming connection weight vector is calculated, with Euclidean distance being the most commonly used distance measure. The exemplar neuron with the least distance between the input vector and its incoming connection weight vector is the winner and is allowed to activate. The winner will emit a value of one, while all other exemplar neurons will stay quiescent. The exemplar neuron layer to class neuron layer connection weights identify which class neuron will activate for each exemplar neuron.

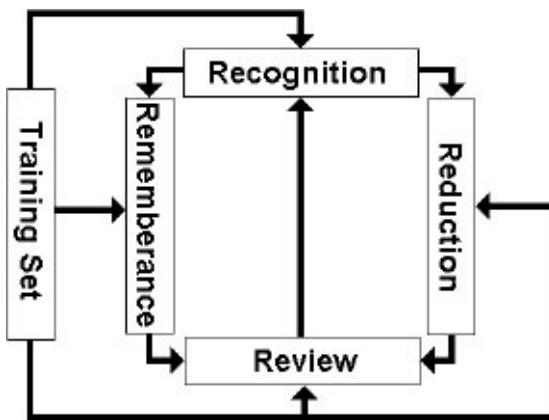


Figure 22.2: The R4 algorithm.

Training of a GAL network occurs one example at a time. If the current example is correctly classified (that is, the correct class neuron is activated) then no changes are made to the network. If the current example is misclassified, then a new exemplar neuron is added. When a new exemplar neuron is added, its incoming connection weights are set to the current input vector, so that it will activate the most strongly if that example is seen again. The outgoing connections of the new exemplar are set so that the desired class neuron activates. If the class is new (not represented by the network) then a new class neuron is added also.

During the course of training, some of the exemplar neurons may become redundant. This is because newer exemplars may sit closer to the boundaries of the classes being learned than the existing exemplars. To deal with these redundant neurons, sleep learning of GAL networks was developed. First, an exemplar neuron is randomly identified. Its incoming connection weight vector is extracted, and used as an input vector. The input vector is then propagated through the network to identify the desired output class for that neuron. Then, the vector is again propagated through the network with the current exemplar neuron disabled. If the network still correctly classifies the example, then the current exemplar is redundant and can be removed. This process can be repeated as many times as desired. Since this procedure can lead to an increase in error for the GAL network, it is often necessary to repeat the training/sleep learning process several times.

A GAL network will always have a winning exemplar neuron. This means that there will always be one class neuron that activates, whether it is the right class or not. To deal with this, the top two activated exemplars are found (the “best two” exemplars) and their corresponding classes identified. If the two classes are different, then no class neuron will activate. If they are the same, then that class will be the winner. This has the effect of giving the network the capacity to reject an example as unclassifiable.

The basic GAL learning algorithm does not modify the connection weights. The only modification to the network is via the addition and deletion of neurons. This has some disadvantages, such as sensitivity to noise and a tendency to memorize the training set at the expense of generalization. Fine-tuning of GAL can overcome this problem to some extent, by modifying the input to exemplar connection weights. This modification is done according to Equation (1).

$$W_{i,e}(t+1) = W_{i,e}(t) + \alpha(P_i - W_{i,e}(t)), \quad (1)$$

where

$W_{i,e}(t)$ is the connection weight from input i to exemplar e ,

α is a learning rate term, that is usually set to decay as learning progresses, and P_i is element i of the current training vector \mathbf{P} .

The intent of this modification is to move the winning exemplar neurons weight vector towards the statistical mean of that class. It is noted (Alpaydin, 1994, p. 407) that “a large number of iterations will be necessary” to tune the GAL network. GAL is limited to classification problems. However, it is able to continue to learn additional datasets after the completion of the initial training cycle. Also, the removal of redundant neurons through sleep learning will protect it from over-training. Although the training algorithm is iterative, the algorithm itself is quite simple. If the number of training iterations were kept to a minimum, then GAL could be expected to be very efficient.

22.3.4. Problems with Constructive Algorithms

This section has presented a sample of the constructive algorithms in existence. While there are many others, this sample covers the major principles employed and illustrates the drawbacks of these algorithms. The major problem that is suffered by most, especially the older, backpropagation trained MLP derived algorithms, is that they are limited to learning a single dataset, a problem that is shared by ANN evolved with EA. This is a major limitation, as one of the major advantages of constructive networks is that they can continue to adapt to new data throughout their lifetime. Another significant problem is that some constructive algorithms are only suitable for classification, for example GAL (Kwok and Yeung, 1999).

22.4. ECoS

ECoS networks have several advantages over other constructive algorithms. Firstly, they are not limited to a particular application domain; they can be applied to both classification and function approximation. Secondly, they do not require multiple presentations of the training dataset, as is the case with some of the constructive algorithms in existence, such as RAN and GAL. Finally, they are able to continue learning and are not restricted to learning a single training set as other constructive algorithms are.

ECoS is a class of constructive ANN architectures with a learning algorithm that modifies the structure of the network as training examples are presented. Although the seminal ECoS architecture was the Evolving Fuzzy Neural Network (EFuNN), several other architectures have been developed that utilize the ECoS algorithm. These include the minimalist Simple Evolving Connectionist System (SECoS) and the Dynamic Evolving Neural-Fuzzy Inference System (DENFIS).

ECoS was designed around the following principles (Kasabov, 1998):

- (1) The ECoS training algorithm is designed with the intention that all the algorithm can learn from the data is learned in the first training pass (one-pass learning). Additional exposure to the training data is not necessary.
- (2) ECoS are intended to be used in an online learning application. This means that new data will be constantly and continuously coming into the system, and that this data must be learned by the network without forgetting the old. The general ECoS architecture and learning algorithm allows an ECoS network to accommodate this new data without a catastrophic forgetting of the old.
- (3) The manner in which neurons are added to an ECoS means that some training examples are stored, initially, verbatim within the structure of the network. These examples are then either modified (refined) by exposure to further examples, or, depending upon the training parameters used, remain the same and can be later retrieved.

The advantages of ECoS are that they avoid the problems associated with traditional connectionist structures such as MLP (Kasabov, 1998; Kasabov *et al.*, 2003): They are hard to over-train, due to the constructive nature of their learning algorithm; they learn quickly, as the learning algorithm is a one-pass algorithm, that is, it requires only a single presentation of the dataset; and they are far more resistant to catastrophic forgetting than most other models, as new training data is represented by adding new neurons, rather than accommodating the additional data in the existing neurons. ECoS networks also have several advantages over other constructive algorithms. Firstly, they are not limited to a particular application domain; they can be applied to both classification and function approximation. Secondly, they do not require multiple presentations of the training dataset, as is the case with some of the constructive algorithms in existence, such as RAN

(Platt, 1991) and GAL (Alpaydin, 1994). Finally, they are able to continue learning and are not restricted to learning a single training set as are other constructive algorithms.

22.4.1. ECoS Architecture and Learning

The first ECoS network was EFuNN (Kasabov, 1998), from which a generalized constructive ANN architecture and training algorithm was derived. An ECoS network is a multiple neuron layer, constructive ANN. An ECoS network will always have at least one “evolving” neuron layer. This is the constructive layer, the layer that will grow and adapt itself to the incoming data, and is the layer with which the learning algorithm is most concerned. The meaning of the connections leading into this layer, the activation of this layer’s neurons and the forward propagation algorithms of the evolving layer all differ from those of classical connectionist systems such as MLP. For the purposes of this chapter, the term “input layer” refers to the neuron layer immediately preceding the evolving layer, while the term “output layer” means the neuron layer immediately following the evolving layer. This is irrespective of whether or not these layers are the actual input or output layers of the network proper. For example, in [Figure 22.3](#), which shows a generic ECoS structure, the “input layer” could be the input layer proper of the network (as with SECoS networks) or it could be a neuron layer that processes the actual input values for presentation to the evolving layer (as with EFuNN networks). By the same token, the “output layer” could be the actual output layer of the network (as it is with SECoS) or a neuron layer that further processes the outputs of the evolving layer (as with EFuNN). The connection layers from the input neuron layer to the evolving neuron layer and from the evolving layer to the output neuron layer are fully connected.

The activation A_n of an evolving layer neuron n is determined by Equation (2).

$$A_n = 1 - D_n, \quad (2)$$

where D_n is the distance between the input vector and the incoming weight vector for that neuron. Since ECoS evolving layers are fully connected, it is possible to measure the distance between the current input vector and the incoming weight vector of each evolving-layer neuron. Although the distance can be measured in any way that is appropriate for the inputs, this distance function must return a value in the range of zero to unity. For this reason, most ECoS algorithms assume that the input data will be normalized, as it is far easier to formulate a distance function that produces output in the desired range if it is normalized to the range zero to unity.

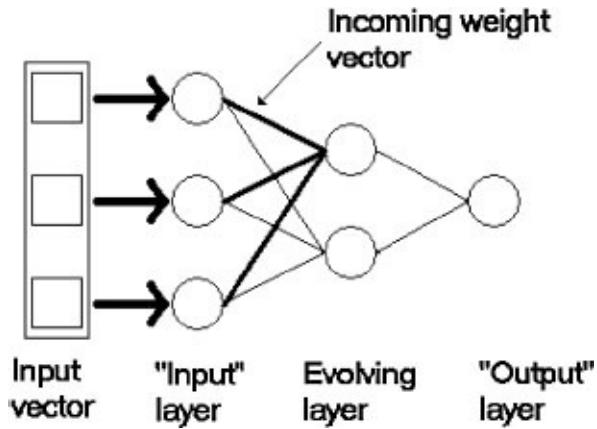


Figure 22.3: General ECoS architecture.

Whereas most ANN propagate the activation of each neuron from one layer to the next, ECoS evolving layers propagate their activation by one of two alternative strategies. The first of these strategies, entitled OneOfN propagation, involves only propagating the activation of the most highly activated (“winning”) neuron. The second strategy, Many of N propagates the activation values of those neurons with an activation value greater than the activation threshold A_{thr} .

The ECoS learning algorithm is based on accommodating new training examples within the evolving layer, by either modifying the weight values of the connections attached to the evolving layer neurons, or by adding a new neuron to that layer. The algorithm employed is described below.

- For each input vector \mathbf{I} and its associated desired output vector \mathbf{O}_d
 - Propagate \mathbf{I} through the network,
 - Find the most activated evolving layer neurons j and its activation A ,
 - If $A_j < S_{\text{thr}}$ then,
 - Add a neuron,
 - Else
 - Find the errors E between \mathbf{O}_d and the output activations A_o ,
 - If $E > E_{\text{thr}}$ then,
 - Add a neuron,
 - Else
 - Update the connections to the winning evolving layer neuron j .

The addition of neurons to the evolving layer is driven by the novelty of the current training example: if the current example is particularly novel (it is not adequately represented by the existing neurons) then a new neuron will be added. Four parameters are involved in this algorithm: the sensitivity threshold S_{thr} , the error threshold E_{thr} , and the

two learning rates η_1 and η_2 . The sensitivity threshold and error threshold both control the addition of neurons and when a neuron is added, its incoming connection weight vector is set to the input vector \mathbf{I} , and its outgoing weight vector is set to the desired output vector $\mathbf{O_d}$. The sensitivity and error thresholds are measures of the novelty of the current example. If the current example causes a low activation (that is, it is novel with respect to the existing neurons) then the sensitivity threshold will cause a neuron to be added that represents that example. If the example does not trigger the addition of a neuron via the sensitivity threshold, but the output generated by that example results in an output error that is greater than the error threshold (that is, it had a novel output), then a neuron will be added.

22.4.2. Evolving Fuzzy Neural Network

EFuNN were proposed by Kasabov (1998c). They are fuzzy neural networks developed within the ECoS framework, that is, they are FuNNs (Kasabov *et al.*, 1997) that have been modified to take into account the principles of ECoS. The distance measure used in EFuNN is the normalized fuzzy distance, as defined in Equation (3).

$$D_n = \frac{\frac{1}{2} \left(\sum_{i=1}^c |\mathbf{I}_i - \mathbf{W}_{i,n}| \right)}{\sum_{i=1}^c \mathbf{W}_{i,n}}, \quad (3)$$

where

c is the number of condition neurons,

\mathbf{I} is the input vector, and

\mathbf{W} is the input to condition layer weight matrix.

The learning algorithm is only slightly different from the general ECoS learning algorithm, in that the weight values for the incoming and outgoing connections of a new neuron are the fuzzified input and output values: there must therefore be two additional steps performed the fuzzification of the current input and desired output vectors, before insertion of the new neuron.

22.4.3. Simple Evolving Connectionist System

The SECoS was proposed (Watts and Kasabov, 2000) as a minimalist implementation of the ECoS algorithm. SECoS were first introduced in Watts (1999b) and more fully developed in Watts and Kasabov (2000). Lacking the fuzzification and defuzzification mechanisms of EFuNN, the SECoS model was created for several reasons. Firstly, they are a simpler alternative to EFuNN. Since they lack the fuzzification and defuzzification structures of EFuNN, SECoS are much simpler to implement: with fewer connection matrices and a smaller number of neurons, there is much less processing involved in simulating a SECoS network. They are also much easier to understand and analyze: while

EFuNN expands the dimensionality of the input and output spaces with its fuzzy logic elements, SECoS deals with the input and output space “as is”. Therefore, rather than dealing with a fuzzy problem space, SECoS deals with the problem space directly. Each neuron that is added to the network during training represents a point in the problem space, rather than a point in the expanded fuzzy problem space. Secondly, for some situations, fuzzified inputs are not only unnecessary but harmful to performance, as they lead to an increase in the number of evolving layer neurons. Binary datasets are particularly vulnerable to this, as fuzzification does nothing but increase the dimensionality of the input data. By removing the fuzzification and defuzzification capabilities, it is proposed that the adaptation advantages of EFuNN are retained while eliminating the disadvantages of fuzzification, specifically by eliminating the need to select the number and parameters of the input and output membership functions.

A SECoS has only three layers of neurons, the input layer, with linear transfer functions, an evolving layer with distance-based neuron activation functions, and an output layer with a simple saturated linear activation function. The distance function used in the evolving layer is the normalized Manhattan distance, as shown in Equation (4).

$$D_n = \frac{\sum_{i=1}^c |\mathbf{I}_c - \mathbf{W}_c|}{\sum_{i=1}^x |\mathbf{I}_c + \mathbf{W}_c|}, \quad (4)$$

where

c is the number of input neurons in the SECoS,

\mathbf{I} is the input vector,

\mathbf{W} is the input to evolving layer weight matrix.

The learning algorithm is the same as that used by EFuNN. However, in SECoS the input vector is the actual crisp input vector, while the desired outputs vector is the unmodified desired output vector.

22.4.4. Dynamic Evolving Fuzzy Inference System (DENFIS)

DENFIS was developed by Kasabov and Song (2002). While EFuNN and SECoS apply supervised clustering of the input–output data, the co-ordinates of the cluster centers being represented as connection weights (input and output respectively) of the evolving (hidden) neurons, DENFIS evolves the hidden nodes through unsupervised clustering of the input examples and then evolves a function to approximate the data in each cluster as a local output model. This function can be a linear regression, a Takagi–Sugeno function of any order, a nonlinear model such as another ANN, etc.

DENFIS is a time-series prediction model that can work successfully for online, real time prediction problems (Kasabov *et al.*, 2003; Kasabov, 2007).

22.4.5. Applications of ECoS

ECoS have been applied to many different application areas. The first of these is automatic speech recognition. In Kasabov (1999), EFuNNs were applied to spoken phoneme recognition. EFuNNs were again applied to this problem in Watts (1999a), and were shown to be both accurate and highly adaptable. The efficiency of SECoS networks in this application area was demonstrated in Watts and Kasabov (2000), where a comparison with MLP showed that they were faster training, more adaptable and more accurate than MLP. Speech recognition via whole word recognition has also been done using ECoS networks. EFuNNs were used in Ghobakhloou *et al.* (2000), while SECoS were adopted in Ghobakhloou and Seesink (2001), for speech-control of a robot, and in Ghobakhloou and Kasabov (2003), for control of a home-automation system.

Bioinformatics is another area where ECoS have been applied. Prediction of RNA initiation sites, that is, sites on RNA strands where protein transcription begins, was accomplished in Futschik *et al.* (1999). Comparisons between EFuNN and MLP were carried out, and EFuNN was found to be able to detect unique RNA patterns that other methods could not. EFuNN was used to model gene regulatory networks in Kasabov *et al.* (2000). The data for the model were gene expression levels captured from micro-arrays. Useful rules describing the development of the cells through time were discovered. Using EFuNN, the work in Futschik *et al.* (2003) identified cancer tissues from gene expression in micro arrays and used rule extraction to identify cancer genes. No comparison with other algorithms was done, but this was not an issue because knowledge discovery via rule extraction was the goal of the work. Useful rules were extracted from the trained EFuNN. A review of bioinformatics and neuroinformatics application of ECoS can be found in Kasabov (2008a; 2008b; 2007; 2014).

Clinical and micro array genetic data was combined using EFuNN in Futschik *et al.* (2003; 2003b) to generate patient prognoses, where the outcomes were either that the patient would die, or the patient would recover. A multi-modular approach was used, where clinical data was modeled by one module and genetic data by another. The reported accuracies of EFuNN were higher than those reported by earlier researchers (87.5% as opposed to 77.6%).

Another bio-medical application of EFuNN was presented in Leichter *et al.* (2001). In this paper, an EFuNN was used to classify the stimulus received by a subject based on the electrical activity of the brain, as read by an EEG. Although the purpose of the paper was to demonstrate an independent component analysis (ICA) based approach to de-noising the EEG data, EFuNN was found to be accurate.

ECoS networks have been shown to be well-suited to image and video processing. Classifying motion vector patterns, or the changes from one frame to the next in an MPEG video stream was the application in Koprinska and Kasabov (1999). The task was to classify a frame from a compressed video stream as one of six classes: static, panning,

zooming, object motion, tracking, and dissolve. This work compared the performance of EFuNN against the LVQ algorithm (Kohonen, 1997), as well as analyzing the effect of varying the number of membership functions on performance.

The performance of MLP and EFuNN were compared in Sorwar *et al.* (2001), where the application was the classification of images of textures. EFuNN was found to be more accurate and to have a much lower computational load than MLP. The lower computational load presumably came about from EFuNNs faster training algorithm.

Horticultural applications of image processing were the subject of Elder (2000); Woodford (2001, 2008a); Woodford *et al.* (2004). The problem dealt with here was classification of pest damage on apple tree leaves. This involved presenting full color images to an EFuNN that then had to identify which of three insect pests—Codling Moth, Leafroller, and Appleleaf Curling Midge—had caused the damage in the image. EFuNN were found to be more accurate at classifying images than k-means, MLP and SVM.

The ability to extract fuzzy rules was also considered to be an advantage in this application area. EFuNN were used to classify images of handwritten digits in Ng *et al.* (2004), where an extensive exploration of the parameters of the EFuNN and the problem itself was carried out. Accuracies ranged from 94–99%, showing how effective EFuNN was for this problem.

In the area of computer network security, ECoS have been shown to be very useful. In Abraham and Jain (2005); Abraham *et al.* (2005) EFuNN were used for network intrusion detection, where they achieved a detection accuracy of 100%. Incoming network packets were analyzed by EFuNN in Chavan *et al.* (2004) to determine whether or not the packets represented an attack. A comparison with MLP showed that the rapid adaptation capability of EFuNN made it the better choice for this application.

The flexibility of ECoS networks have led to them being applied to many other application areas. In Kasabov and Fedrizzi (1999) the task was to predict the New Zealand stock market index the SE40. Comparison of the results of EFuNN with a FuNN showed that EFuNN was both more accurate and more adaptable than FuNN. EFuNN were also applied to predicting the SE40 in Wang (2001), although no comparison with other models was presented. The NASDAQ-100 stock index was modeled in Abraham *et al.* (2001a), where an EFuNN was used to predict whether or not stocks were going to go up or down. The results were described in the paper as “promising”, although no comparison with other predictive methods was reported.

EFuNN was used in Abraham *et al.* (2001b; 2001c) to predict rainfall in the southern Indian state of Kerala, and a similar application, modeling rainfall in Switzerland, was presented in Woodford (2005; 2008b), where the goal was to learn more about the rainfall phenomenon by extracting fuzzy rules from EFuNN.

EFuNN was again used in de Moraes and dos Santos Machado (2005) for online

evaluation of trainee performance in a virtual reality training task. A comparison with MLP showed that EFuNN was marginally more accurate (98.6% compared with 95.6%).

The problem of detecting abnormal behavior by the occupants of an intelligent environment was addressed in Rivera-Illingworth *et al.* (2005), where they used a SECoS network to integrate the data from 18 different sensors in a single room. While the accuracy of SECoS was equivalent to other methods that were investigated, the fast training speed and adaptability of SECoS made it more useful.

Time-delay neural networks and EFuNN were used to classify the outputs of an artificial nose in Zanchettin and Ludermir (2004), where it was shown that EFuNN was faster and more accurate.

Many applications of DENFIS were reported for superior time series prediction when compared to other ANN and AI methods (Kasabov, 2007).

In every case where ECoS networks were compared with conventional ANN such as MLP, the ECoS networks were shown to have better accuracy, to train faster, and to adapt better. ECoS networks have therefore been shown to be a useful family of algorithms with wide-ranging applications. Other authors have further developed, modified and improved these techniques for various applications (see Kasabov (2007, 2014)).

22.4.6. Evolutionary Optimization of ECoS

Although ECoS networks are able to avoid many of the problems associated with other constructive algorithm, there is still scope for optimization using EA. The goal when optimizing ECoS is to achieve a network that is accurate (whether over training or testing datasets) and is parsimonious, that is, that has the minimum number of evolving-layer neurons necessary to achieve the first criterion. Firstly, the parameters must be optimized to the problem at hand; Secondly for datasets with a large number of features, feature selection is helpful in achieving a smaller ECoS; Thirdly, as online pattern-based learning algorithms, the order in which the initial training vectors are presented will also influence the performance of the ECoS.

There are two different modes in which an ECoS network may be optimized, online and offline. Online optimization is carried out while the ECoS is training, that is, while neurons are being added and connection weights modified. Conversely, offline optimization is carried out outside of the training process.

ES were used to optimize the parameters online in both Watts and Kasabov (2001) and Chan and Kasabov (2004). While Watts and Kasabov (2001) used a simple (1+1) ES to optimize all four basic training parameters for each training example, Chan and Kasabov (2004) used a (μ, λ) ES to optimize the learning rates only, with respect to a sliding window of data where the window was moved though the training data stream. A similar windowing-of-data approach was used in Kasabov *et al.* (2003), although in this case a

GA was used instead of an ES. The GA optimized all learning parameters and the fitness was evaluated as a function of the error over the training data window.

GA were used for offline optimization of both training parameters and the order of training examples in (Watts and Kasabov, 2002; Watts, 2004). The fitness function in this case was based on minimizing both the error over the provided dataset, and the overall change in the size of the evolving layer of the ECoS. A similar approach was taken in Minku and Ludermir (2005), although the order of the training examples was not optimized. A GA was also used for offline optimization of the training parameters in Kasabov and Song (2002), where the fitness function was based entirely on the error over the training dataset. A co-evolutionary GA was used to evolve ensembles of EFuNN in Minku and Ludermir (2006), where data was first clustered, and each cluster split into a training and a testing dataset. The GA optimized an individual EFuNN over each training and testing set, where the optimization was with respect to the accuracy and size of the network. The overall result was that the ensemble performed better than a single EFuNN trained over the entire dataset.

An unusual approach to offline optimization was reported in Arnott (2000). In this work, a GA was used to optimize the fuzzy rules and membership functions that were extracted from a trained EFuNN. The modified rules and MF were then re-inserted into the EFuNN for testing. Thus, the GA was really optimizing a fuzzy system, although the close-coupling between EFuNN and their associated rules and MF made it a useful approach.

22.5. Conclusions and Further Directions

This chapter presented an historic review of some of the methods, systems and applications of hybrid evolving ANN, including: evolutionary ANN, constructive ANN, and some ECoS methods and applications. More methods and applications of ECoS are presented in the next chapter of this volume. ECoS is an area of computational intelligence that is further developing with new methods being proposed every year, many of them based on more brain-inspired and nature inspired techniques. These are the evolving spiking neural networks (eSNN), the evolving neurogenetic systems and the quantum inspired ECoS.

References

- Abraham, A. and Jain, R. (2005). Soft computing models for network intrusion detection systems. In *Classification and Clustering for Knowledge Discovery*. Berlin, Heidelberg: Springer, pp. 191–207.
- Abraham, A., Jain, R., Sanyal, S. and Han, S. Y. (2005). SCIDS: A soft computing intrusion detection system. In *Distributed Computing — IWDC 2004*. Berlin, Heidelberg: Springer, pp. 252–257.
- Abraham, A., Nath, B. and Mahanti, P. K. (2001a). Hybrid intelligent systems for stock market analysis. In *Computational Science — ICCS 2001*. Berlin, Heidelberg: Springer, pp. 337–345.
- Abraham, A., Philip, N. S. and Joseph, K. B. (2001b). Will we have a wet summer? Soft computing models for long term rainfall forecasting. In *15th European Simulation Multi conference (ESM 2001), Modelling and Simulation*, pp. 1044–1048.
- Abraham, A., Steinberg, D. and Philip, N. S. (2001c). Rainfall forecasting using soft computing models and multivariate adaptive regression splines. *IEEE SMC Trans., Special Issue on Fusion of Soft Computing and Hard Computing in Industrial Applications*.
- Alpaydin, E. (1994). GAL: Networks that grow when they learn and shrink when they forget. *Int. J. Pattern Recogn.*, 8(1), pp. 391–414.
- Arena, P., Caponetto, R., Fortuna, L. and Xibilia, M. G. (1993). M.L.P. optimal topology via genetic algorithms. In *Artificial Neural Nets and Genetic Algorithms*, New York: Springer-Verlag Wien, pp. 670–674.
- Arnott, R. (2000). Application of a genetic algorithm to optimise EFuNN rule sets. *Technical report*, Department of Information Science, University of Otago, New Zealand.
- Ash, T. (1989). Dynamic node creation in backpropagation networks. *Connect. Sci.*, 1(4), pp. 365–375.
- Chavan, S., Shah, K., Dave, N., Mukherjee, S., Abraham, A. and Sanyal, S. (2004). Adaptive neuro-fuzzy intrusion detection systems. In *Proc. Int. Conf. Info. Tech.: Coding Comp. (ITCC04)*, pp. 70–75.
- Chan, Z. S. and Kasabov, N. (2004). Evolutionary computation for online and offline parameter tuning of evolving fuzzy neural networks. *Int. J. Comput. Intelligence Ap.*, 4(03), pp. 309–319.
- Chellapilla, K. and Fogel, D. B. (2001). Evolving an expert checkers playing program without using human expertise. *IEEE T Evolut. Comput.*, 5(4), pp. 422–428.
- Choi, B. and Bluff, K. (1995). Genetic optimisation of control parameters of a neural network. In Kasabov, N. K. and Coghill, G. (eds.), *Artificial Neural Networks and Expert Systems*. IEEE Computer Society Press, pp. 174–177.
- Cybenko, G. (1989). Approximation by super positions of sigmoidal function. *Math. Control Signal*, 2, pp. 303–314.
- de Garis, H. (1990). Genetic programming: Evolution of a time dependent neural network module which teaches a pair of stick legs to walk. In *Proc. Ninth European Conf. Artificial Intelligence*. Stockholm, Sweden.
- de Garis, H. (1992). Artificial nervous systems: The genetic programming of production-rule-GenNet circuits. In *Proc. Int. Joint Conf. Neural Networks*. Beijing, China.
- de Moraes, R. M. and dos Santos Machado, L. (2005). Evaluation system based on EFuNN for online training evaluation in virtual reality. In *CIARP 2005, LNCS 3773*.
- Fahlman, S. E. (1988). An empirical study of learning speed in back-propagation networks. *Technical Report CMU-CS-88-162*, Department of Computer Science, Carnegie-Mellon University.
- Fahlman, S. E. and Lebiere, C. (1990). The cascade-correlation learning architecture. In Touretzky, D. S. (ed.), *Advances in Neural Information Processing Systems 2*, Morgan Kaufman Publishers, pp. 524–532.
- Fogel, D. B., Wasson, E. C., Boughton, E. M. and Porto, V. W. (1997). A step toward computer-assisted mammography using evolutionary programming and neural networks. *Cancer Lett.*, 119(1), pp. 93–97.
- Fogel, D. B., Hays, T. J., Hahn, S. H. and Quon, J. (2004). A self-learning evolutionary chess program. *P. IEEE*, 92(12), pp. 1947–1954.
- Fontanari, J. and Meir, R. (1991). Evolving a learning algorithm for the binary perceptron. *Networks*, 2, pp. 353–359.
- Frean, M. (1990). The upstart algorithm: A method for constructing and training feedforward neural networks. *Neural Comput.*, 2(2), pp. 198–209.

- Futschik, M., Schreiber, M., Brown, M. and Kasabov, N. (1999). Comparative studies of neural network models for mRNA analysis. In *ISMB 1999*, Heidelberg, p. 43.
- Futschik, M. E., Reeve, A. and Kasabov, N. (2003). Evolving connectionist systems for knowledge discovery from gene expression data of cancer tissue. *Artif. Intell. Med.*, 28, pp. 165–189.
- Futschik, M. E., Sullivan, M., Reeve, A. and Kasabov, N. (2003). Modular decision system and information integration for improved disease outcome prediction. In *Proc. ECCB'2003*, pp. 13–15.
- Futschik, M. E., Sullivan, M., Reeve, A. and Kasabov, N. (2003). Prediction of clinical behaviour and treatment for cancers. *OMJ Applied Bioinformatics*, 2(3), pp. S53–S58.
- Fukuda, T., Komata, Y. and Arakawa, T. (1997). Recurrent neural networks with self-adaptive GAs for biped locomotion robot. In *Proc. Int. Conf. Neural Networks ICNN' 97*. IEEE Press, Houston.
- Ghobakhloo, A. and Kasabov, N. (2003). A methodology for adaptive speech recognition systems and a development environment. In *Proc. Artif. Neural Networks Neural Information Processing ICANN/ICONIP 2003 Int. Conf.* Istanbul, Turkey, pp. 316–319.
- Ghobakhloo, A. and Seesink, R. (2001). An interactive multi modal system for mobile robotic control. In *Proc. Fifth Biannual Conf. Artif. Neural Networks and Expert Systems (ANNES2001)*, pp. 93–99.
- Ghobakhloo, A., Watts, M. J. and Kasabov, N. (2000). Online expansion of output space in evolving fuzzy neural networks. In *Proc. ICONIP 2000*, Taejon, Korea, Vol. 2, pp. 891–896.
- Hancock, P. J. (1992). Genetic algorithms and permutation problems: A comparison of recombination operators for neural net structure specification. In *International Workshop on Combinations of Genetic Algorithms and Neural Networks, COGANN-92*, IEEE, pp. 108–122.
- Kasabov, N., Kozma, R., Kilgour, R., Laws, M., Taylor, J., Watts, M. and Gray, A. (1997). A methodology for speech data analysis and a framework for adaptive speech recognition using fuzzy neural networks. In Kasabov, N., Kozma, R., Ko, K., O'Shea, R., Coghill, G. and Gedeon, T. (eds.), *Progress in Connectionist-Based Information Systems*. Singapore: Springer.
- Kasabov, N. (1998). The ECoS framework and the ECO learning method for evolving connectionist systems. *J Adv. Comput. Intell.*, 2(6), pp. 195–202.
- Kasabov, N. (1998). Evolving fuzzy neural networks — algorithms, applications and biological motivation. In Yamakawa, T. and Matsumoto, G. (eds.), *Methodologies for the Conception, Design and Application of Soft Computing*. World Scientific, Vol. 1, pp. 271–274.
- Kasabov, N. (1999). Evolving connectionist systems: A theory and a case study on adaptive speech recognition. In *International Joint Conference on Neural Networks, IJCNN '99*. Vol. 5, pp. 3002–3007.
- Kasabov, N., Erzegovezi, L., Fedrizzi, M., Beber, A. and Deng, D. (2000). Hybrid intelligent decision support systems and applications for risk analysis and prediction of evolving economic clusters in Europe. In N. Kasabov, (ed.), *Future Directions for Intelligent Information Systems and Information Sciences*. Springer Verlag, pp. 347–372.
- Kasabov, N. and Fedrizzi, M. (1999). Fuzzy neural networks and evolving connectionist systems for intelligent decision making. In *Proc. Eighth Int. Fuzzy Systems Association World Congress*, Taiwan, pp. 30–35.
- Kasabov, N., Kim, J., Watts, M. J. and Gray, A. R. (1997). FuNN/2 — a fuzzy neural network architecture for adaptive learning and knowledge acquisition. *Inform. Sciences*, 101(3), pp. 155–175.
- Kasabov, N. and Song, Q. (2000). Dynamic evolving neuro-fuzzy inference system (DENFIS): Online learning and application for time-series prediction. In *Proc. Sixth Int. Conf. Soft Computing. Fuzzy Logic Systems Institute*, Iizuka, Japan, pp. 696–702.
- Kasabov, N. and Song, Q. (2002). GA — parameter optimisation of evolving connectionist systems for classification and a case study from bioinformatics. In *Proc. Ninth Int. Conf. on Neural Information Processing, ICONIP'02*. IEEE, Vol. 2, pp. 602–605.
- Kasabov, N., Song, Q. and Nishikanawa, I. (2003). Evolutionary computation for dynamic parameter optimisation of evolving connectionist systems for online prediction of time series with changing dynamics. In *Proc. Int. Joint Conf. on Neural Networks*. IEEE, Vol. 1, pp. 438–443.
- Kasabov, N. (2001). Brain-like functions in evolving connectionist systems for on-line, knowledge-based learning. In

- Kitamura, T. (ed.), *What should be Computed to Understand and Model Brain Function, Volume 3 of FLSI Soft Computing Series*. World Scientific, pp. 77–113.
- Kasabov, N. (2003). *Evolving Connectionist Systems: Methods and Applications in Bioinformatics, Brain Study and Intelligent Machines*. Springer.
- Kasabov, N. (2007). *Evolving Connectionist Systems, Second Edition*. Springer.
- Kasabov, N. (2008). Evolving intelligence in humans and machines: integrative connectionist systems approach, feature article. *IEEE CIS Magazine*, August, 2008, Vol. 3, No. 3, pp. 23–37.
- Kasabov, N. (2008). Adaptive modelling and discovery in bioinformatics: The evolving connectionist approach. *Int. J. Intell. Syst.*, 23, pp. 545–555.
- Kasabov, N. (ed.) (2014). *The Springer Handbook of Bio-/Neuroinformatics*. Springer, p. 1230.
- Kermani, B. G., Schiffman, S. S. and Nagle, H. T. (1999). Using neural networks and genetic algorithms to enhance performance in an electronic nose. *IEEE T. Bio-med. Eng.*, 46(4), pp. 429–439.
- Kohonen, T. (1997). *Self-Organizing Maps, Second Edition*. Springer.
- Kolmogorov, A. (1957). On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. *Dokl. Akad. Nauk. USSR*, 114, pp. 953–956. (In Russian)
- Koprinska, I. and Kasabov, N. (1999). An application of evolving fuzzy neural network for compressed video parsing. In *ICONIP/ANZIIS/ANNES'99 Workshop*. Dunedin, New Zealand, pp. 96–102.
- Kwok, T.-Y. and Yeung, D.-Y. (1999). Constructive algorithms for structure learning in feedforward neural networks for regression problems. *IEEE T. Neural Netw.*
- Leichter, C. S., Cichocki, A. and Kasabov, N. (2001). Independent component analysis and evolving fuzzy neural networks for the classification of single trial EEG data. In *Proc. Fifth Biannual Conf. Artif. Neural Networks and Expert Systems (ANNES 2001)*, pp. 100–105.
- Ma, C. Y., Wong, S. W., Hecht, D. and Fogel, G. B. (2006). Evolved neural networks for high throughput anti-HIV ligand screening. In *IEEE Congress on Evolutionary Computation, CEC 2006*. IEEE, pp. 2734–2741.
- McCloskey, M. and Cohen, N. (1989). Catastrophic interference in connectionist networks: The sequential learning project. *Psychol. Learn. Motiv.*, 24, pp. 109–164.
- Mézard, M. and Nadal, J.-P. (1989). Learning in feedforward layered networks: The tiling algorithm. *J. Phys. A*, 22, pp. 2191–2203.
- Minku, F. L. and Ludermir, T. B. (2005). Evolutionary strategies and genetic algorithms for dynamic parameter optimization of evolving fuzzy neural networks. In *The 2005 IEEE Congress on Evolutionary Computation*, IEEE. Vol. 3, pp. 1951–1958.
- Minku, F. L. and Ludermir, T. B. (2006). EFuNNs ensembles construction using a clustering method and a coevolutionary genetic algorithm. In *IEEE Congress on Evolutionary Computation, CEC 2006*, IEEE, pp. 1399–1406.
- Moreira, M. and Fiesler, E. (1995). Neural networks with adaptive learning rate and momentum terms. *Technical Report 95-04*, Institut Dalle D’Intelligence Artificielle Perceptive.
- Ng, G. S., Murali, T., Shi, D. and Wahab, A. (2004). Application of EFuNN for the classification of handwritten digits. *Int. J. Comput. Systems Signals*, 5(2), pp. 27–35.
- Platt, J. (1991). A resource-allocating network for function interpolation. *Neural Comput.*, 3(2), pp. 213–225.
- Porto, V. W., Fogel, D. B., Fogel, L. J., Fogel, G. B., Johnson, N. and Cheung, M. (2005). Classifying sonar returns for the presence of mines: evolving neural networks and evolving rules. In *2005 IEEE Symp. Comput. Intell. Homeland Security and Personal Safety*. IEEE Press, Piscataway, NJ, pp. 123–130.
- Rivera-Illingworth, F., Callaghan, V. and Hagras, H. (2005). A neural network agent based approach to activity detection in AmI environments. In *IEEE International Workshop, Intelligent Environments 2005 (IE05)*. Colchester, UK.
- Schiffmann, W., Joost, M. and Werner, R. (1994). Optimization of backpropagation algorithm for training multiplayer perceptrons. *Technical Report*, Institute of Physics, University of Koblenz.
- Schiffmann, W., Joost, M. and Werner, R. (1993). Application of genetic algorithms to the construction of topologies for

- multilayer perceptrons. In Albrecht, R., Reeves, C. R. and Steele, N. (eds.), *Artificial Neural Nets and Genetic Algorithms*. New York: Springer-Verlag Wien, pp. 675–682.
- Sorwar, G., Abraham, A. and Dooley, L. S. (2001). Texture classification based on DCT and soft computing. In *The Tenth IEEE Int. Conf. on Fuzzy Systems*, Vol. 3, pp. 545–548.
- Wang, X. (2001). “On-line” time series prediction system — EFuNN-T. In *Proc. Fifth Biannual Conf. Artif. Neural Networks and Expert Systems (ANNES 2001)*. pp. 82–86.
- Watts, M. J. (1999a). An investigation of the properties of evolving fuzzy neural networks. In *Proc. ICONIP’99*. Perth, Australia, pp. 217–221.
- Watts, M. J. (1999b). Evolving connectionist systems for biochemical applications. In Kasabov, N. and Ko, K. (eds.), *Emerging Knowledge Engineering and Connectionist-based Systems* (Proc. ICONIP/ANZIIS/ANNES’99 Workshop “Future directions for intelligent systems and information sciences”, Dunedin, 22–23 November 1999). Dunedin, New Zealand: University of Otago Press, pp. 147–151.
- Watts, M. J. and Kasabov, N. (2000). Simple evolving connectionist systems and experiments on isolated phoneme recognition. In *Proc. First IEEE Conf. Evolutionary Computation and Neural Networks*. IEEE Press, San Antonio, pp. 232–239.
- Watts, M. and Kasabov, N. (2001). Dynamic optimisation of evolving connectionist system training parameters by pseudo-evolution strategy. In *Proc. 2001 Congress Evolutionary Computation*. IEEE. Vol. 2, pp. 1335–1342.
- Watts, M. and Kasabov, N. (2002). Evolutionary optimisation of evolving connectionist systems. In *Computational Intelligence, Proc. World on Congress on* (Vol. 1, pp. 606–610). IEEE.
- Watts, M. J. (2004). *Evolving Connectionist Systems: Characterisation, Simplification, Formalisation, Explanation and Optimisation*. Doctoral dissertation.
- Woodford, B. J. (2005). Rule extraction from spatial data using local learning techniques. In *Proc. SIRC 2005—The Seventeenth Annual Colloquium of the Spatial Information Research Centre*. University of Otago, Dunedin, New Zealand.
- Woodford, B. J. (2001). Comparative analysis of the EFuNN and the Support Vector Machine models for the classification of horticulture data. In *Proc. Fifth Biannual Conf. Artif. Neural Networks and Expert Systems (ANNES 2001)*.
- Woodford, B. J. (2008a). Evolving neurocomputing systems for horticulture applications. *J. Appl Soft Comput.*, 8, pp. 564–578.
- Woodford, B. J. (2008b). Rule extraction from spatial data using an entropy-based fuzzy neural network. In Whigham, P. A., Drecki, I. and Moore, A. (eds.), *Proc. Twentieth Annual Colloquium of the Spatial Information Research Centre and GEOCOMP (SIRC’ 2008)*. The University of Auckland, Auckland, New Zealand, pp. 55–66.
- Woodford, B. J., Deng, D. and Benwell, G. L. (2004). A wavelet-based neuro-fuzzy system for data mining small image sets. In Hogan, J., Montague, P., Purvis, M. and Steketee, C. (eds.), *Australasian Workshop on Data Mining and Web Intelligence (DMWI2004)*. University of Otago, Dunedin, New Zealand, pp. 139–144.
- Zanchettin, C. and Ludermir, T. B. (2004). Evolving fuzzy neural networks applied to odor recognition in an artificial nose. In *Proc. 2004 IEEE Int. Joint Conf. on Neural Networks*, Vol. 1, pp. 675–680.
- Zhao, Q. and Higuchi, T. (1996). Evolutionary learning of nearest-neighbour MLP. *IEEE T. Neural Netw.*, 7(3), pp. 762–767.

Part V

Applications

Chapter 23

Applications of Computational Intelligence to Decision-Making: Modeling Human Reasoning/Agreement

Simon Miller, Christian Wagner and Jonathan Garibaldi

Computational Intelligence provides valuable approaches and tools for the modeling of human reasoning and decision making. In this chapter, we look at how Computational Intelligence methods can be employed in decision-making problems where information from humans (e.g., survey data) is available. Specifically, we explore an example of modeling cyber security expert decision making using a combination of evolutionary algorithms and aggregation operators as well as generating type-1/type-2 fuzzy sets directly from data to enable comparisons between real world objects such as restaurants. For each of these cases, we show how CI techniques can provide powerful and practical tools for delivering decision support. Throughout, we highlight the capacity of the respective Computational Intelligence methods to capture and model key aspects of human reasoning such as the integration of multiple relevant aspects in challenging cyber security decisions and the often significant uncertainty associated with survey responses.

23.1. Introduction

In this chapter, we will look at how computational intelligence (CI) methods can be employed in decision-making problems where modeling human reasoning or agreement is required. To explore this subject, two examples will be used:

- (1) Evolving Ordered Weighted Average (OWA) operators that model the decision-making processes of cyber-security experts.
- (2) Similarity based applications that employ type-1 and General type-2 Fuzzy agreement models created from interval survey data.

In each of these examples we will show how CI techniques can provide powerful and practical tools for aiding human decision-making problems. The focus of the case studies included here is the use of CI techniques with survey data. A set of real-world surveys is used to demonstrate how CI techniques including evolutionary algorithms (EAs) and fuzzy logic can be utilized to capture the reasoning processes of human experts, and model the object human perception of concepts and words.

23.2. Case Study: Evolving OWA Operators to Model Cyber-Security Experts' Decision-Making

In this example, we will show how EAs can be employed to design OWA operators that model the decision-making processes of cyber-security experts.

23.2.1. *Background*

This section will provide an overview of the scenario, and describe the methods that have been employed.

Cyber-crime has become an increasing concern over recent years. This has been brought about by the constantly growing number of every day activities that are conducted online, for example, banking, shopping and e-government, and the availability of software and hardware that aids would-be attackers, reducing the need for technical knowledge. As a result, the field of cyber-security has rapidly grown in order to address concerns about the security of information systems, and the sensitive data that they hold. An important part of an organization's cyber-security provision is the assessment of security offered by proposed new information systems, while they are still in the design phase. This activity can inform systems designers about where there are weaknesses in their systems, and whether the level of security offered is appropriate for the consequences of a successful attack.

Primarily, cyber-security experts carry out security assessments of proposed systems, as often, this is the only way they can be achieved. However, assessing the security provided by a proposed system is a large and difficult task, and often there are a limited number of cyber-security professionals who are capable of carrying out the job.

In this example, we will consider how the load on cyber-security experts can be reduced, through creating models of parts of their expertise that can be used as decision aids during security assessments. This is achieved using a combination of EAs and OWAs to model some of the reasoning used by experts during an assessment.

23.2.2. *The Scenario*

The case study that we will be using is a decision-making exercise that was conducted at GCHQ—the UK Government's signals intelligence and information assurance agency. In the exercise a group of 39 cyber-security experts took part in a survey about the difficulty of technical attacks, and compromising or bypassing security components. The experts were drawn from a wide variety of backgrounds and included software architects, technical security consultants, penetration testers, vulnerability researchers, and specialist systems evaluators. All of the experts involved were highly experienced in their field, and accustomed to how UK government information systems are typically constructed and operated.

During the exercise, the experts were presented with a scenario created by GCHQ technical experts. The scenario was a proposal for a new government system that included details of the architecture of the system, some potential attacks on the system and the (security) components it was made up of. In reality, assessing such systems is an extremely in-depth task that requires a great deal of information. To address this, the experts were told to assume that the software and hardware used, and the frequency of software updates were typical of the government systems that they came across in their day-to-day work.

[Figure 23.1](#) shows an example of the technical attacks that experts were shown. The diagram shows details of the system, the components of the system, and describes the path that an attacker would need to follow in order to successfully complete the attack. This attack involves an email with a malicious attachment containing an exploit, enabling the attacker to compromise the recipient's PC, and launch an ongoing attack on the system.

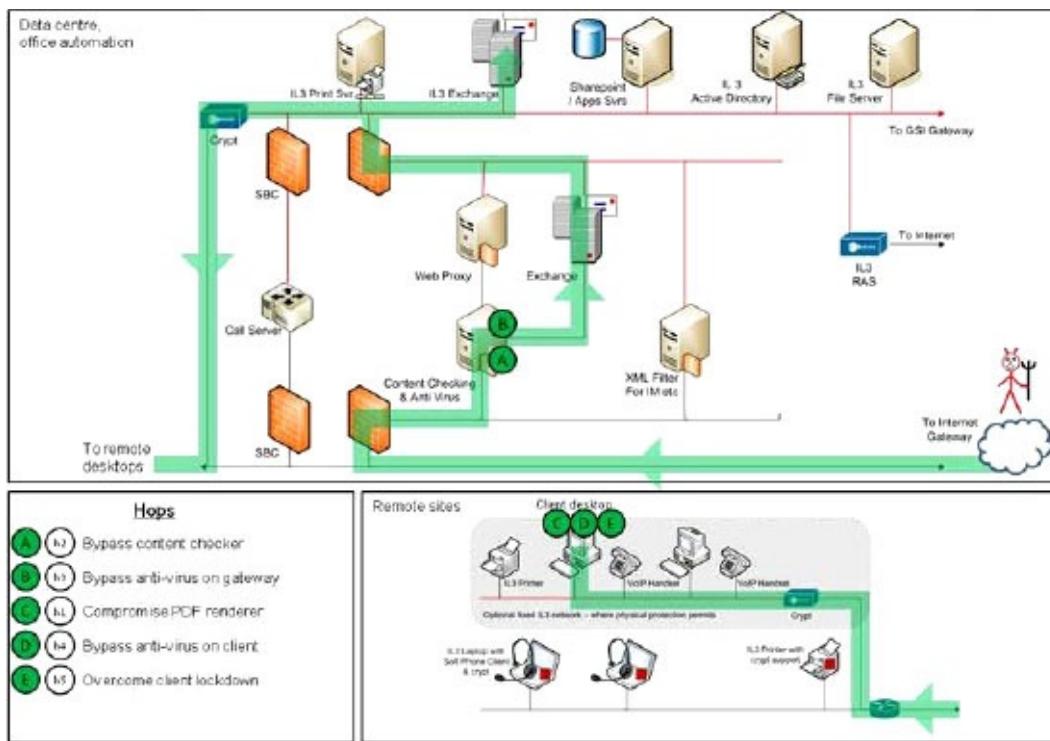


Figure 23.1: Example technical attack from scenario.

In order to complete the example attack shown in [Figure 23.1](#), five components need to be either compromised or bypassed:

- (1) Gateway Content Checker.
- (2) Gateway Anti-virus.
- (3) PDF renderer.
- (4) Client Anti-virus.
- (5) Client Access Controls.

23.2.3. The Survey

In total, the experts were shown 10 attacks on the proposed system. A survey was then conducted entailing two distinct types of activity:

- (1) Ranking the 10 technical attacks in order of how difficult they are to successfully complete without being detected.
- (2) Rating the difficulty of compromising or bypassing the 26 security components that make up the 10 attacks.

Ranking the attacks was simply a case of placing them in order from easiest to hardest. This method of eliciting difficulty allows experts to perform a series of pairwise comparisons in order to establish an order of difficulty, e.g., “How does attack *a* compare with attack *b*?”, is it easier or harder? Making distinctions in this way can facilitate the process of identifying salient attacks, as giving an abstract rating can sometimes prove difficult. A different type of survey was used to elicit the difficulties of compromising/bypassing each of the 26 security components. Experts were asked, “Overall, how difficult would it be for an attacker to successfully compromise/bypass this component without being noticed?”, and they gave a rating between 0 (very easy) and 100 (very hard). This type of questioning allows experts to give an indication of the absolute difference in difficulty between security components, unlike ranking, which provides an intuitive way to obtain an ordering but elicits no information about the degree of difference between attacks.

Once this data has been collected from the experts, we have two levels of information:

- (1) Rankings of attacks.
- (2) Ratings of the overall difficulty of compromising/bypassing specific security components that make up attacks.

In this example, we will look at how the overall difficulty of specific components contributes to the difficulty of attacks they form part of, and which components are most important when ranking attacks. This is achieved through the use of OWAs to derive attack rankings from overall difficulty ratings.

Table 23.1: Attacks with constituent components.

Attack	Components
1	2,3,1,4,5
2	6,7,6,8,4
3	9
4	10,11,4,5
5	12,13,2,3,14,15,4,5
6	16,16,17,4,5
7	6,18,4,5

8	19,20,21
9	22,23,24
10	25,26,1,4,5

Each of the 26 components belongs to one or more of the 10 attacks, Table 23.1 shows which components belong to each attack. Note that some components are duplicated in a single attack, this is because a particular attack may involve bypassing/compromising the same type of component more than once, for example, an attacker may have to compromise multiple firewalls to reach a target.

In practice, a system is only as secure as its weakest element, i.e., the easiest way in. Identifying which are the weakest aspects of a system, i.e., the easiest ways of attacking it, is thus a highly relevant component of system security assessment, though obviously it does not provide all of the answers. Using this data we demonstrate how technical attack rankings can be derived from the component ratings through aggregation, and compare them with the actual rankings provided by the experts. This is particularly useful in real world assessment, as the method provides the potential for highlighting the most salient attacks for a new system using a database of pre-rated components, reducing the burden on expert resources.

23.2.4. Ordered Weighted Averages

The concept of exploring the relationships between the overall difficulty of components, and the difficulty of attacks came after discussion with technical experts from GCHQ in which a hypothesis was raised. When we asked about how they decide how difficult an attack is, we were told that each component of the attack is assessed, estimating how difficult it would be to bypass/compromise. Then, the difficulty of an attack is based chiefly on the most difficult component, with the remaining components carrying less weight in order of difficulty.

In order to explore this, and simulate the process of creating attack ratings and rankings by aggregating assessments of security components, we use OWA operators (Yager, 1988). An OWA is an operator that allows us to assign weights to objects based upon their position in an ordering (e.g., a difficulty ranking), as opposed to a traditional weighted average in which weights are assigned to specific objects. Equation (1) describes the process, where n is the number of weights/objects. The OWA consists of a set of weights (w) that add up to 1, and a set of objects that we wish to apply the operator to. In this case study the objects are the security components, or more specifically, the experts' assessments of the security components. To perform the operation we first sort the objects into order, for our security components this means sorting them by difficulty in descending order. Once the objects have been placed into an ordering (o), we apply each of the weights to each of the objects in turn, the first weight is multiplied by the first object and so on. This allows us to specify a weight to apply to the most difficult

component to compromise, the second most difficult, and so on. Once all of the weights have been applied to the objects, the sum of the results is calculated. In practice, if the first weight is close to 1, the resulting operator will close to a maximum operator, if the final weight is close to 1, the operator will be close to a minimum operator, between these extremes lie an almost infinite number of operators.

$$OWA_w(a) = \sum_{i=1}^n w_i o_i. \quad (1)$$

[Table 23.2](#) shows the values provided by one expert for each of the components of attack 1 (2,3,1,4,5).

The OWA can then be applied to this set of values to produce an overall score for attack 1. For example, if we wanted to choose an OWA operator that gives more weight to the most difficult hop, our weights could be $w = 0.33, 0.27, 0.20, 0.13, 0.07$. To apply these to the expert's ratings $a = 25.00, 40.00, 20.50, 40.00, 70.00$, we first sort them by difficulty rating producing the following ordering: $o = 70.00, 40.00, 40.00, 25.00, 20.50$. We can then compute the OWA by multiplying each weight by the corresponding element in the ordering as shown in Equation (1), where n is equal to the number of weights/components. In this case a value of 46.59 is produced; the mean value is 39.10, showing that this OWA operator has given more weight to higher values of difficulty.

Table 23.2: Expert's 'overall' interval means for attack 1.

Component	Interval Mean
1	25.00
2	40.00
3	20.50
4	40.00
5	70.00

This procedure can be repeated for each of the attacks, using a given OWA operator. The result is a set of difficulty values for each of the attacks, that can be used to rank them from most to least difficult. Finally, the derived ranking can be compared with the actual ranking given by an expert to validate the OWA operator in question. In this case study we will use an EA to find suitable OWA weights for creating attack ratings.

23.2.5. Evolutionary Algorithms

EAs are a popular method of optimization for problems with very large solution spaces (see Chapter 14). The most widely used EA method, the genetic algorithm (GA) is a heuristic search technique inspired by evolutionary biology. Selection, crossover, and mutation are applied to a population of individuals representing solutions in order to find a near-optimal solution. The GA is able to find suitable solutions to a problem, while

evaluating a small fraction of the solution space. For problems with an extremely large solution space, this is essential.

Here, we will employ an EA to search a large number of possible OWA operators for the purpose of aggregating security component ratings to create ratings/rankings of specific technical attacks. The purpose of this is two-fold:

- (1) To discover suitable OWA operators for creating ratings/rankings of attacks using assessments of security components.
- (2) As the EA is unrestricted, the resulting weighting will also confirm/contradict our hypothesis that the most difficult component is the most important when ranking attacks, and will give a guide to how much importance can be assigned to each component.

For this case study the EA is implemented as follows:

Solutions are represented using a vector of weights. The maximum number of components in an attack is eight, therefore eight weights are used in solutions. For those attacks that do not have eight components, their component ratings are padded out with zeros so that there are eight. This avoids the potential for attacks that have fewer components being judged as being more difficult, purely because their weightings are concentrated on fewer components. An attack with more components should be more difficult than an equivalent attack with fewer components.

The *initial solution* is created by generating seven random points on a line from 0 to 1. The weights are the distances between zero, each point and 1. This ensures that the weights will add up to 1.

Fitness is determined by comparing each individual's actual attack ranking with the attack ranking derived from their component ratings using an OWA. Spearman's Rho is used to measure correlation between the rankings, producing a value between -1 (perfect negative correlation), 0 (no correlation), and 1 (perfect positive correlation). Error is calculated by subtracting each Spearman's Rho value from 1 , so a perfect positive correlation will result in an error of zero. The mean squared error (MSE) over all individuals provides a metric for assessing the performance of a particular set of weights.

In *parent selection*, individuals are sorted by fitness and random individuals are chosen from the population using absolute values over a normal distribution forming a complementary cumulative distribution. That is to say, it is more likely that a lower numbered (and therefore fitter) individual will be chosen, though it is possible for any individual to be chosen.

In *Elitism*, the best individuals from each generation are copied into the next generation. This ensures that the best individual in a generation cannot be worse than the best in the preceding generation.

Mutation is achieved by selecting two weights from a solution, increasing one by a small amount and decreasing the other by the same amount. If either weighting becomes invalid (>1 or <0) another two elements are selected. This method ensures that the weights still add up to 1 after perturbation.

A single point *crossover* is used that takes two parents and creates a child consisting of the first half of the first parent, and the second half of the second parent. To make sure that the weights add up to 1, they are normalized. This allows us to preserve the characteristics of each parent, while maintaining a valid set of weights.

23.2.6. Evolving OWAs

The next step is to conduct a series of tests with the EA to determine appropriate proportions for the genetic operators and population/generations. Configurations have been chosen to result in a similar number of evaluations.

Experiments have been conducted on three groups of individuals: “odd” (i.e., the 1st, 3rd, 5th,... individuals), “even” (i.e., the 2nd, 4th, 6th,... individuals) and “all” individuals. The reason for this is that after tuning the weights with 50% of the overall group, we can then test these weights to see how they perform on unseen individuals in the remaining 50% of the group. The group containing all individuals will give an indication of the best that can be found using the selected search algorithms for the entire group.

23.2.7. Results

[Table 23.3](#) shows the configurations tested, [Table 23.4](#) provides the best results for each group and [Table 23.5](#) describes the OWAs discovered. For each of these tests a population of 250 individuals is used over 250 generations, with 1% elitism.

Table 23.3: EA: Test set 1.

Test	Copy	Cross.	Mut.
1	0.00	0.20	0.79
2	0.20	0.20	0.59
3	0.40	0.20	0.39
4	0.60	0.20	0.19
5	0.79	0.20	0.00
6	0.50	0.00	0.49
7	0.30	0.40	0.29
8	0.20	0.60	0.19
9	0.10	0.80	0.09
10	0.00	0.99	0.00

Table 23.4: EA: Test set 1—best results.

Test	Mean Sp.	MSE
Even		
3	0.6866	0.1425
Odd		

1	0.5713	0.2543
All		
1	0.6159	0.2069
2	0.6159	0.2069

Table 23.5: EA: Test set 1—best OWAs.

Test	Best Weights							
	1	2	3	4	5	6	7	8
Even								
3	0.8985	0.0059	0.0311	0.0102	0.0065	0.0258	0.0037	0.0184
Odd								
1	0.7872	0.0185	0.1242	0.0505	0.0111	0.0032	0.0029	0.0024
All								
1	0.9582	0.0028	0.0242	0.0003	0.0057	0.0029	0.0002	0.0056
2	0.9621	0.0017	0.0221	0.0003	0.0045	0.0059	0.0009	0.0025

For all groups, the earlier tests result in the best solutions, when there is a small amount of crossover and large proportion of mutation. The copy operator does not appear to have much influence on the results, though this may be because the best individuals from each population are automatically copied to the next generation through elitism.

23.2.7.1. *Test set 2—population and generations*

The best configuration from the previous set of tests for each group was used with a variety of generations/population combinations. [Table 23.6](#) provides the configurations tested, [Table 23.7](#) provides the results and [Table 23.7](#) describes the best OWAs discovered. In the previous tests on the “all” group two configurations were tied for the lowest MSE. Both configurations were used in this set of tests, the (best) result shown in [Tables 23.4](#) and [23.7](#) was achieved using the setup from test 2 in the previous set of tests.

In these tests there is less variation in the results, as we might expect, altering the configuration has more of an effect on the results than changing the ratio of populations to generations. All of the solutions found with the EA have a large bias towards the most difficult component, with very little weight being attributed to the remaining components.

23.2.7.2. *Alternate group weightings*

The next stage examines how the best weights for one group perform on the other. [Table 23.9](#) shows the results of applying the best weights from the “odd” and “even” groups on the alternate group.

The results show that while the results are relatively good, suggesting that the OWAs found are robust, they are not as good as those seen in the previous tests with the same weights. It is reasonable to expect some reduction in performance, as the OWAs have been tuned to be specific to each group, that they still produce good aggregations indicates that

it is likely that OWAs tuned on a sample of data can be applied to unseen data with good results.

Table 23.6: EA: Test set 2.

Test	Gens.	Pop.
11	50	1250
12	100	625
13	200	315
14	300	210
15	400	155
16	500	125

Table 23.7: EA: Test set 2—best results.

Test	Mean Sp.	MSE
Even		
12	0.6885	0.1421
13	0.6866	0.1421
Odd		
13	0.5732	0.2519
16	0.5733	0.2519
All		
15	0.6165	0.2080

Table 23.8: EA: Test set 2—best OWAs.

Test	Best Weights							
	1	2	3	4	5	6	7	8
Even								
12	0.8907	0.0011	0.0323	0.0120	0.0070	0.0174	0.0035	0.0359
13	0.8899	0.0044	0.0297	0.0126	0.0078	0.0242	0.0030	0.0284
Odd								
13	0.7858	0.0223	0.1285	0.0471	0.0115	0.0031	0.0015	0.0002
16	0.7854	0.0246	0.1248	0.0479	0.0117	0.0033	0.0001	0.0021
All								
15	0.9456	0.0022	0.0296	0.0006	0.0068	0.0011	0.0086	0.0055

Table 23.9: EA: Application of best weights to alternate group.

Test	Mean Sp.	MSE
Best ‘even’ weights applied to ‘odd’		
12	0.5228	0.3013
13	0.5146	0.3118
Best ‘odd’ weights applied to ‘even’		
13	0.5915	0.2381
16	0.5947	0.2337

Table 23.10: EA: Extended tests.

Max Sp.	Min Sp.	Mean Sp.	Std. Dev. Sp.
0.6200	0.6104	0.6150	0.0022
Max MSE	Min MSE	Mean MSE	Std. Dev. MSE
0.2137	0.2057	0.2092	0.0020

23.2.7.3. Extended test

Finally, the best result from the “all” group is replicated 30 times with differing random seeds to give an idea of the variation in results. [Table 23.10](#) shows the results of using the best weights from the “all” group over 30 tests, including the mean Spearman’s Rho, mean MSE, and the standard deviation.

The results of the extended test show that the results are relatively stable. There is little difference between the maximum and minimum MSE (0.0080), and looking at the individual sets of weights it can be seen that they are all very similar. All solutions produced in this extended run of tests placed a weight greater than 0.92 on the first (most difficult) component.

23.2.8. Summary

In this section a series of EAs have been used to search for OWA operators, for the purpose of aggregating ratings of hops (security components) into ratings of AVs. It has been shown that EAs are able to find OWA operators that provide a robust method of deriving ratings and rankings of attacks from hop difficulty ratings.

What this tells us is that aggregating hop ratings is an appropriate method of rating and ranking specific attacks, and that given a database of ratings of generic hops (as was used in these experiments) unseen attacks can be rated. This provides the facility for systems designers to undertake what if? scenarios to estimate the effects of altering or adding/removing components to/from a proposed system.

Further to this we have also seen that the EAs, which are led solely by MSE, always find OWAs that place at least 92% of the weight on the most difficult hop. This lends weight to our hypothesis that the most difficult hop is the most important when rating the difficulty of an attack, and that the remaining hops carry little weight. The use of a combination of EAs and OWAs can be applied to any problem in which we want to find out the relative importance of sub-decisions (e.g., the level of security offered by specific components) in the process of making an overall decision (e.g., the difficulty of completing an entire attack).

23.3. Case Study: Using Interval Valued Survey Data in Similarity Based Applications of Type-1 and Type-2 Fuzzy Agreement Models

In this case study we will show how interval valued survey data can be modeled using type-1 (T1) and general type-2 fuzzy sets (GT2FSs) with an approach called the interval agreement approach (IAA) (Miller *et al.*, 2012), and how the resulting models can be employed in similarity based applications that aim to produce linguistic descriptions of objects (in this case, pubs and restaurants), and to compare objects with one another.

23.3.1. Background

Fuzzy logic sets (see Chapter 1) are a useful and popular method of modeling the opinions and perceptions of groups of people. A lot of the work in this area has concentrated on the creation of fuzzy set (FS)-based models for words and/or concepts (e.g., (Wagner *et al.*, 2015; Miller *et al.*, 2012; Mendel and Wu, 2010; Coupland *et al.*, 2010 and Liu and Mendel, 2007)). The aim is to accurately capture the meaning people associate with words/concepts using FSs in order to be able to perform computation and reasoning based on these modelled words/concepts. Type-2 fuzzy sets (T2FSs) have been highlighted as particularly suitable for this approach as their additional degree of freedom in comparison to type-1 fuzzy sets (T1FSs) allow them to more accurately capture the distribution of uncertainty within a given set. This in turn enables complex modelling of variation and uncertainty in the meaning of words.

Capturing data directly from individuals on an area of interest is an important, and often the only, means of eliciting information and knowledge which is commonly strongly subjective. Once the data have been collected, the combined opinions of a group (and sub-groups) can be represented in models allowing for analysis of, and reasoning with, the consolidated knowledge that has been acquired.

23.3.2. The Scenario

The case study we will be using for this section focuses on survey exercises in which participants were asked to give their opinions of aspects of pubs and restaurants, and how they perceive words in specific contexts. With the data collected during the surveys, fuzzy models are produced using an approach called the IAA (Miller *et al.*, 2012) and the resulting models are used in two example similarity-based applications.

23.3.3. The Survey

Data collected during two survey exercises are used in this case study. The first consisted of two questionnaires, one asked participants to rate a series of local pubs/bars according to a number of concepts such as *Ambience* and *Service*. For each pub (e.g., *Rose and Crown*), participants were asked a series of six questions (e.g., “*How would you rate the*

service in the Rose and Crown?”). In the other questionnaire, participants were asked to rate words in the context of specific concepts. For example, one of contexts presented to participants was *ambience in a pub*. They were then asked to rate the words *negative*, *neutral*, and *positive* in this context. The participants provided word-ratings for all 18 words (six concepts, with three words describing each). Words were chosen to elicit responses appropriate for each specific concept, e.g., *Delicious* for *Quality of Beer* or *Cheap* for *Cost of Beer*.

The second survey was conducted at the University of Nottingham, UK with a small group of six staff and students on two separate occasions, approximately eight weeks apart. On each occasion participants were asked to rate a series of concepts relating to the same 20 local restaurants. For example, questions included “*How polite are the staff*” and “*Overall, how would you rate this eating place?*”.

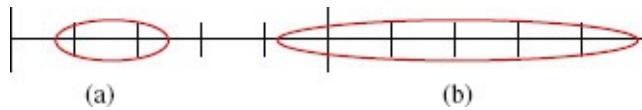


Figure 23.2: Interval responses, where “a” is a less uncertain response and “b” is a more uncertain response.

In both surveys a method of eliciting opinion was used that employed a design based on ellipses leading to intervals, i.e., where participants can express their uncertainty about a given response through specifying an ellipse/interval. The width of the interval denotes the participant’s certainty in their answer, a narrow interval is used when they are sure where on the scale the answer lies, and a wider one where they are less certain. [Figure 23.2](#) provides examples of more and less uncertain responses.

23.3.4. The Interval Agreement Approach

The IAA (Miller *et al.*, 2012; Wagner *et al.*, 2015) is a method of constructing T2 and GT2FSs using interval valued survey data like that used in this example. The IAA consists of two possible steps.

- (1) The combination of multiple interval valued data to create T1FSs.
- (2) The combination of T1FSs from step 1, to create GT2FSs.

For example, if we have multiple surveys of the same group (i.e., we have repeated a survey with the same participants) then we may use step 1 to produce a T1FS for each participant that models the opinions (over time) incorporating each of the responses an individual has given. The degree of membership to each of the T1FS models represents the intra-user agreement, that is, how much agreement there is over multiple surveys for the same person. Then, in the second step, we could combine the T1FS models to create a GT2FS. In this set, as before, the primary domain captures the agreement that the users have with themselves over repeated surveys. The secondary domain represents the inter-user agreement, that is, the level of agreement between multiple users. The GT2FSs capture both the intra- and inter-user agreement in two distinct domains.

Alternatively, if we only have one survey for each participant for example, we might use step 1 to model the inter-user agreement between a group of people for a specific survey. In this example, we will use both of these approaches for the two surveys we have conducted.

23.3.5. Similarity Measures

Similarity measures are methods that are used in (fuzzy) set theory to compare crisp, T1 and T2FSs.

23.3.6. Similarity Measures for Crisp and T1 Fuzzy Sets

In traditional (crisp) set theory, similarity measures have long been a useful tool allowing the comparison of sets in terms of their likeness. With the advent of FS theory, naturally, there was a desire to realize the important concept of similarity using FSs. A well-accepted and popular method used to measure similarity in both crisp and FS theory is the Jaccard similarity coefficient (Jaccard, 1908) which can be defined for crisp sets (CSs) as the cardinality of the intersection of two sets, divided by the cardinality of the union of the two sets as shown in Equation (2). For T1FSs, Equation (2) can be expressed as Equation (3). The result is a value $\in [0, 1]$ representing how similar A is to B , with 1 indicating that they are identical and 0 meaning that they are disjoint.

$$S_J^{CS}(A, B) = \frac{A \cap B}{A \cup B}, \quad (2)$$

$$S_J^{FS}(A, B) = \frac{\sum_{i=1}^N \min(\mu_A(x_i), \mu_B(x_i))}{\sum_{i=1}^N \max(\mu_A(x_i), \mu_B(x_i))}. \quad (3)$$

23.3.7. Similarity Measures for T2FSs

A number of methods have been proposed for measuring the similarity of interval T2FSs (IT2FSs) (e.g., Bustince, 2000; Gorzalczany, 1987; Wu and Mendel, 2008 and Zeng and Li, 2006) and GT2FSs (e.g., Mitchell, 2005 and Yang and Lin, 2009). In Nguyen and Kreinovich (2008) and Wu and Mendel (2009) the Jaccard similarity coefficient is extended for use with IT2FSs. Equation (4) shows how the extended Jaccard similarity coefficient is defined in Wu and Mendel (2009) where S_J^{IT2FS} is the extended Jaccard similarity function, and \tilde{A} and \tilde{B} are IT2FSs.

$$S_J^{IT2FS}(\tilde{A}, \tilde{B}) = \frac{\int_X^{} \min(\bar{\mu}_{\tilde{A}}(x), \bar{\mu}_{\tilde{B}}(x))dx + \int_X^{} \min(\underline{\mu}_{\tilde{A}}(x), \underline{\mu}_{\tilde{B}}(x))dx}{\int_X^{} \max(\bar{\mu}_{\tilde{A}}(x), \bar{\mu}_{\tilde{B}}(x))dx + \int_X^{} \max(\underline{\mu}_{\tilde{A}}(x), \underline{\mu}_{\tilde{B}}(x))dx}. \quad (4)$$

It can be seen that in this case, the upper and lower membership functions are used to

determine the similarity of two IT2FSs. As the IAA approach generates both T1 and GT2FSs, we employ the Jaccard similarity for consistency. For T1FSs we use the Jaccard method described in Equation (3), and for GT2FSs we employ an extended form of the IT2 Jaccard similarity measure described in Wu and Mendel (2009) which we introduced in McCulloch *et al.* (2013). Equation (5) shows how Equation (4) can be expressed for GT2FSs as a weighted average of the Jaccard similarity for IT2FSs.

$$S_J^{zGT2FS}(\tilde{A}, \tilde{B}) = \frac{\sum_{i=1}^I z_i S_J^{IT2FS}(\tilde{A}_i, \tilde{B}_i)}{\sum_{i=1}^I z_i}, \quad (5)$$

where \tilde{A} and \tilde{B} are GT2FSs, I is the number of zSlices Wagner *et al.*, 2010, i.e., the IT2FSs that make up the GT2FS, i indicates a particular zSlice and z_i is the secondary degree of membership at each zLevel. Using this method we are able to compare perceptions of concepts/words, which will enable us to create linguistic profiles of pubs/restaurants, and make comparisons between them.

23.3.8. Similarity Based Applications of Fuzzy Word and Concept Models

Using the interval valued data collected during the surveys, T1 and GT2 fuzzy models created using the IAA, and similarity measures, we now demonstrate two practical applications.

- (1) Creating linguistic descriptions for pubs based on the similarity of models of specific concepts (aspects) that relate to them to models of words that can be used to describe those concepts. For example, we can compute if a given pub's ambience is perceived as *negative*, *neutral*, or *positive* where both the labels and the pub-ratings are represented with fuzzy models.
- (2) Comparing restaurants. For example, an individual may say "*My favorite restaurant is x, which restaurants are most like x*", or, in an unfamiliar place, the system may be used to recommend restaurants that share characteristics with a user's preferred restaurants in her/his home town. This is achieved by comparing fuzzy models of the user's preferred restaurant(s) with models of other restaurants using the similarity measure described previously.

23.3.9. Comparing Concepts and Words

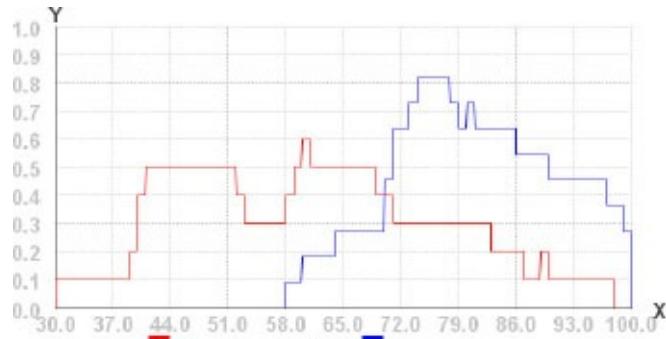
In this application two sets of fuzzy models are created using the IAA.

- (1) Word models describing a set of three adjectives for each of a set of six categories, namely: *Ambience*, *Service*, *Variety of Beer*, *Quality of Beer*, *Cost of Beer*, and *Overall rating*.

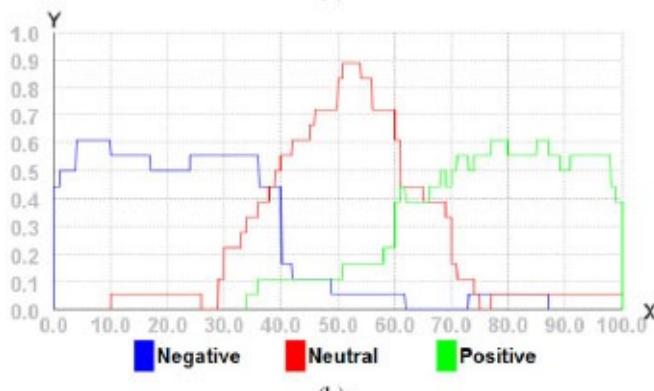
(2) Concept models of the six concepts for a series of 13 venues (pubs) in the center of Nottingham, UK.

Both models are created with the IAA using data collected during a single survey exercise with 18 local residents that employed two questionnaires, one for the rating of the words used to describe concepts of pubs in general and one for rating the actual aspects of individual pubs. The reason that we asked participants to rate the perception of words with regard to specific categories is that it is important that each word is evaluated separately in each context. Words may be perceived differently depending upon the context they are being used in.

Once the data had been collected, the IAA was used to create T1FSs representing each descriptive word and each concept for each pub. In this application the degree of membership represents the inter-user variation, that is, it shows the agreement between users over where the word lies on the scale. [Figure 23.3a](#) shows an example of the T1FS models of the *Ambience* concept for two pubs; *Trip to Jerusalem* and the *Joseph Else*. The figure shows that there concept is a significant difference in how the concept *Ambience* has been rated for each pub; the rating for the *Trip to Jerusalem* is superior to the rating for the *Joseph Else*. From a technical perspective, we can note that the fuzzy models resulting from the IAA are non-convex and non-normal. The non-convexity is a direct result of the IAA which does not remove outliers (by design). The non-normality is a result of the agreement operation, which results in wider and less *high* FSs for lower levels of agreement (overlap) of constituent intervals and higher FSs with a narrower support where there is greater overlap. Thus, the resulting FSs directly reflect the agreement and discord on the given concept in the given context across participants. While it may be tempting at first glance to normalize the sets, the fact that there is no perfect agreement across all participants means that the model for the word/concept should *not* be completely true (i.e., 1) anywhere.



(a)



(b)

Figure 23.3: T1 FSs produced with the IAA for (a) the concept of *Ambience* in *Trip to Jerusalem* and *Joseph Else* and (b) the word models associated with the concept *Ambience*.

Figure 23.3b shows an example of the T1FS word models for *Negative*, *Neutral*, and *Positive* in the context of *Ambience* in a pub. As with the concept models, the resulting FSs clearly indicate agreement and discord amongst participants.

In the context of the application, it is worth noting that the word models can easily change, for example, with a more demanding audience, different intervals may be elicited. Also, different words may provide (more) meaningful results, for example, one could build models for the concept of *music volume* with individual word models, e.g., *low*, *ok*, *loud*, *unbearable*. Again, different audiences are likely to provide very different ratings for the words resulting in different user-centric mappings (similarities) between concepts (pubs/venues) and the given words.

Next, we will look at the outputs of the described application.

23.3.10. Relating Concepts to Words—Results

Once a set of word models and a set of concept models has been produced, we can compare the ratings of the concepts (e.g., *Service* in pub x) to the contextual word models of adjectives used to describe the same concept (e.g., *poor*, *acceptable*, or *excellent* service). The results of this will show us which word model most closely matches the concept models for each pub, allowing us to ask is pub x's *Service* most similar to *poor*, *acceptable*, or *excellent* service? We will then use the similarity values to generate a linguistic description of the pub which is specific to the individual/group of individuals surveyed.

[Figure 23.4](#) shows three example profiles detailing the similarity between the type-1 fuzzy models of each concept for each pub and the T1 fuzzy word models representing the positive, neutral, and negative labels associated with each concept. In [Figure 23.4](#), we can see that the concept models produced for *Trip to Jerusalem* have a high degree of similarity to the word models that represent positive labels for each question. A more evenly balanced response is shown for the *Ropewalk*, and the *Joseph Else* has a set of concept models that most closely match the neutral word models associated with each question.

[Table 23.11](#) provides linguistic descriptions of each concept for each of the venues in [Figure 23.4](#). Words have been selected by taking the word model that has the maximum similarity to each concept for each venue. The linguistic descriptions are clearly representative of the profiles previously shown ([Figure 23.4](#)), however in some cases, taking the maximum may not provide as much information as one would like. For example, the concept *Cost* for *The Ropewalk* produces very close similarity values when compared with the word models *reasonable* and *cheap*. By taking the maximum, we discard the information that it has a (relatively) high similarity with *reasonable*. To address this, we could use qualifiers for each term (e.g., *quite* or *very*). Defining qualifiers could potentially involve a similar methodology to that used to produce the word models.

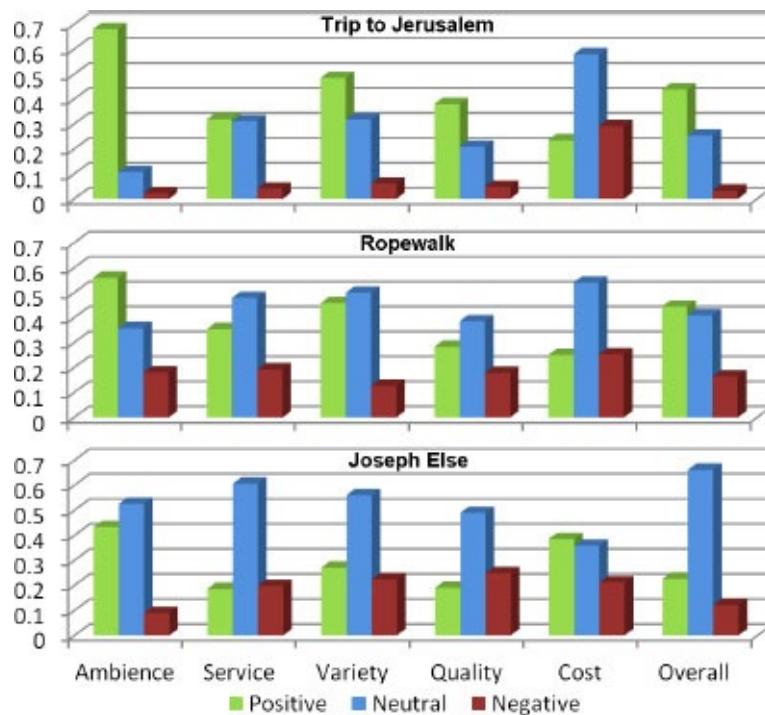


Figure 23.4: Similarity profiles for *Trip to Jerusalem*, *Ropewalk* and *Joseph Else*.

Table 23.11: Numeric example—Data.

Concept	Trip to Jerusalem	Ropewalk	Joseph Else
Ambience	Positive	Positive	Neutral
Service	Excellent	Acceptable	Acceptable
Variety of Beer	Wide	Wide	Reasonable
Quality of Beer	Delicious	Acceptable	Acceptable
Cost of Beer	Cheap	Cheap	Reasonable
Overall	Excellent	Acceptable	Acceptable

The column charts in [Figure 23.5](#) illustrate two examples of the similarity between the word models associated with an individual concept (e.g., *positive*, *neutral*, and *negative*) and the concept models (e.g., *Ambience* of the pubs) generated from the survey. The Ambience chart shows that for most venues the concept model produced for the *Ambience* question is most similar to the fuzzy word model for *positive* ambience. The results are slightly more varied for the *overall* concept which shows a stronger bias toward the *acceptable* label.

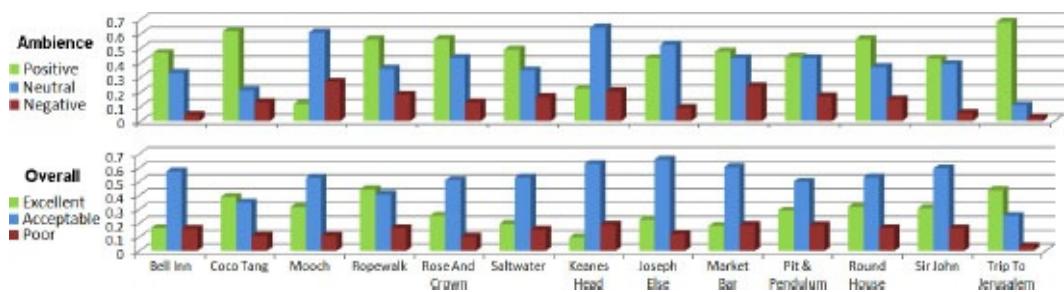


Figure 23.5: Similarity profiles for *ambience* and *overall* rating of pubs.

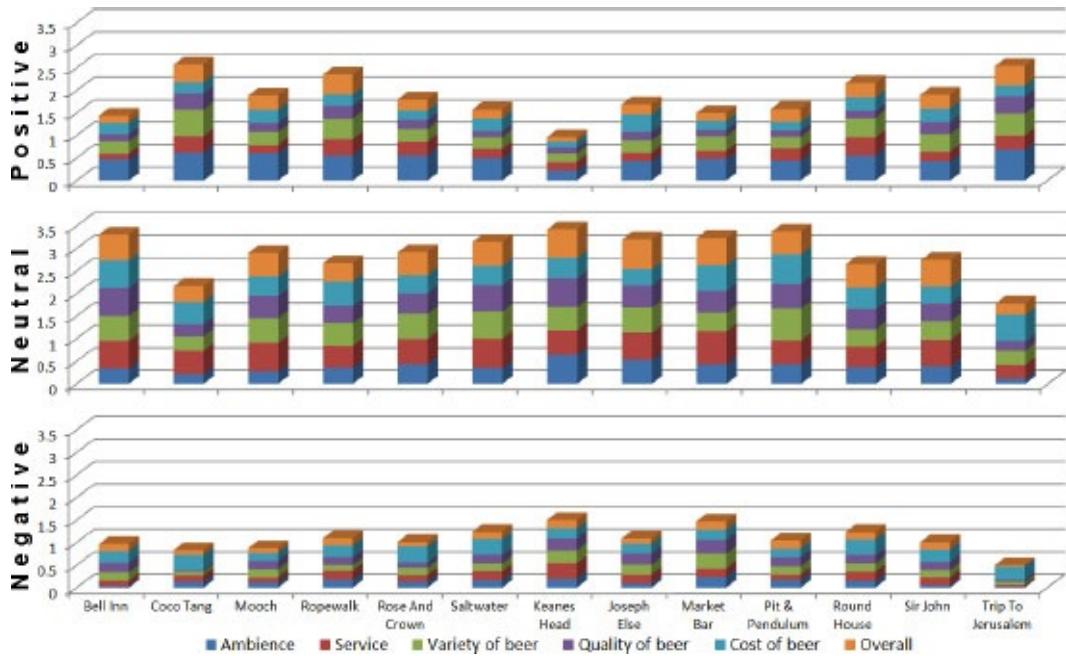


Figure 23.6: Total similarity of each pub with *Positive*, *Neutral*, and *Negative* word groups.

[Figure 23.6](#) contains three column charts, each showing the total similarity of the concept models for each venue to groupings of the *positive*, *neutral*, and *negative* word models associated with each concept. As all concepts were rated with three words, we grouped the words describing *positive* ratings, the words describing *neutral* ratings and the words describing *negative* ratings in the top, middle, and bottom rows respectively. In each chart the columns are divided, showing the contribution each concept has to the

overall total similarity for each pub/word group. It is immediately obvious that very few of the venues had greatest similarity with the *negative* word group models associated with each concept. For most venues the strongest similarities are with the *neutral* labels. Overall, this representation quickly summarizes the desirability of the different pubs (e.g., height of column in positive section) and visualizes which aspects (ambience, service, etc.) have contributed to this height/rating and how much.

23.3.11. Comparing Concept Models Over Multiple Survey Iterations

For the second application we use data collected in a survey conducted twice at the University of Nottingham, UK. Repeated surveying of the same group of participants allows us to derive not only models of the variation between people, but also models of the variation within an individual over time, i.e., the stability of their interpretation of a concept. Participants perceptions of concepts may change over time as their experience grows, or simply because their mood alters. Capturing the intra-user uncertainty is useful for extracting the *core* meaning of a concept. In this application we demonstrate how GT2FS models created with the IAA and interval valued survey data can be used to measure the similarity of (aspects of) restaurants.

The survey was conducted with a small group of six staff and students on two separate occasions approximately eight weeks apart. On each occasion participants were asked to rate a series of concepts relating to the same 20 local restaurants. For example, questions included “*How polite are the staff?*” and “*Overall, how would you rate this eating place?*”. Ratings were provided using intervals as detailed in [Section 23.3.3](#).

For the purposes of this case study, we focus on a subset of the data that includes data provided by all participants, over both surveys, to the question “*Overall, how would you rate this restaurant?*” for three of the restaurants. Though it uses only a portion of the available data, the example demonstrates how GT2FSs created using IAA can be used in conjunction with similarity measures to quantify the similarity of aspects of an item of interest.

As seen in [Section 23.3.4](#), the first step is to create T1FSs. In this example, we use the T1FSs to represent data for each survey separately, capturing the data for each concept, for each survey and for each restaurant. Like the previous application, the membership degree for the T1 sets represents the amount of inter-user agreement on the concept of *overall quality* of each restaurant. Once each restaurant has two T1 fuzzy sets, one generated from each survey, these are combined in step 2 to create a general type-2 fuzzy set for each restaurant. The GT2FS represents inter-user agreement/uncertainty in the primary degree of membership and intra-user agreement/uncertainty in the secondary degree of membership.

The three GT2FSs for this example representing the concept of *Overall Quality* can be seen in [Figure 23.7](#). The difference between the zSlices (the IT2FSs that make up the

GT2FS) indicates how the ratings changed from one survey to the next, i.e., all ratings that were present in the models of the first *and* second iteration are included in the zSlice protruding to the zLevel $z = 1$ (front (light gray) in Figure 23.7), while other ratings that were present in only one of the surveys are only captured in the zSlice protruding to the zLevel $z = 0.5$ (rear (dark gray) in Figure 23.7).

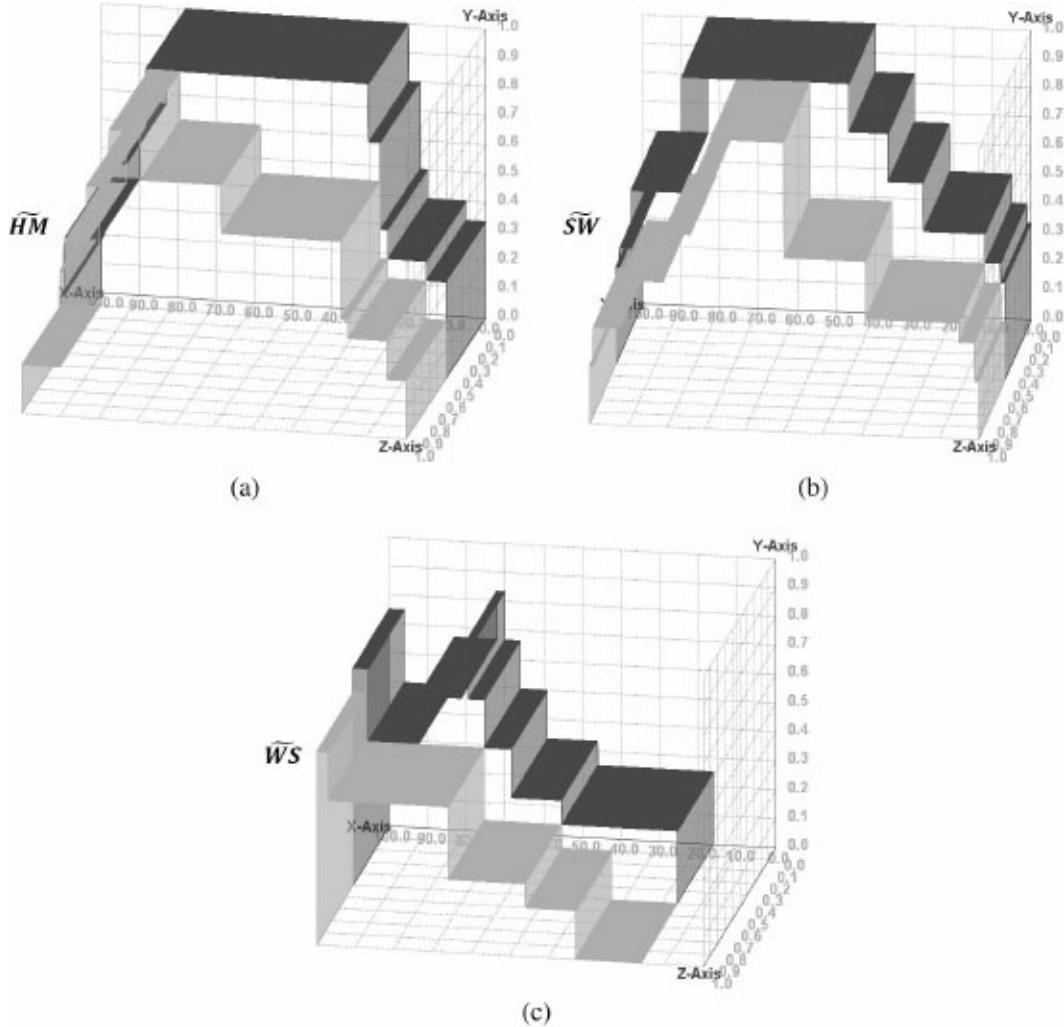


Figure 23.7: general type-2 fuzzy sets for (a) the concept of *overall quality* of (a) *Homemade* (\tilde{HM}) (b)*Saltwater* (\tilde{SW}) and (c) *World Service* (\tilde{WS}). Note that the zSlice Z_2 (front (light gray)) protrudes from $z = 0$ to $z = 1$ but is shown to protrude from $z = 0.5$ to $z = 1$ to improve visibility.

The sets of the restaurants *Homemade* (HM) and *Saltwater* (SW) show that there is some uncertainty in individual participants' perceptions over the two surveys and between participants. Both sets cover a large amount of the domain, and have zSlices that are significantly different from one another. This shows that some participants expressed a (relatively) high level of uncertainty in their answers, that there was a (relatively) low level of agreement (across all participants), and that their perceptions altered between the first and second surveys. Overall both sets are weighted towards the middle/high end of the quality scale. The set produced for *World Service* is different. In this set the area covered by higher membership degrees is much smaller, opinion is more consistently focused on the high end of the quality scale, and the zSlices are more alike. This indicates that participants were more certain of their answers, exhibited higher agreement with one another, and gave more consistent responses over the two surveys.

Similarity is computed using the extended Jaccard similarity measure shown in Equation (5). The resulting similarities between the restaurants are shown in [Table 23.12](#). Looking at the fuzzy sets in [Figure 23.7](#) it can be seen that the results of applying a similarity measure reinforce the impression that HM and SW are most alike, and that *World Service* is different in terms of the model of overall quality.

There are a number of extensions that could be made to our example application to make it more useful. Initially we stated that our aim was to allow the comparison of restaurants through questions like “*My favorite restaurant is x, which other restaurants are like x?*”. In the example we gave questions asking participants to rate restaurants on a scale from negative to positive, clearly most people would want to go to the restaurant that has the most positive response. For example, we can reasonably expect that nobody would say “*I’d like visit a restaurant where the food is rancid, which restaurants have food that is most similar to rancid?*”. They may however ask “*I like restaurant x, which restaurants are at least as good as x?*”. To compute this we would need to use a similarity measure that not only takes into account the proximity of the fuzzy representations of concepts, but also maintained information about in which direction (negative or positive) restaurant x is from restaurant y.

We could also create a different set of questions that relate to characteristics of an establishment rather than its quality, that do not rely on a positive/negative scale. For example, we could ask questions like “*Do you consider pub x to be traditional, or modern?*”, or “*Do you consider restaurant y to have a clientèle that is young or mature?*”. This kind of questioning will result in answers across the domain which describe a type of venue, rather than the perceived quality of a venue. Models of concepts like these are much more fitting for the purpose of recommending venues. For example, if a person likes a pub that is modern with a young clientèle, we can use similarity measures to find other pubs that are similar to the concept models associated with this pub, i.e., they are modern with a young clientèle. In addition to this we can also create profiles for individuals with questions such as “*Do you prefer pubs that are traditional or modern?*” and use a personal profile to match an individual with establishments that they may be interested in visiting. Part of this type of survey could be to use word models like *Modern* created using the approach seen in the first example, and then allow participants to use words to describe their preferences. These can then be used to identify venues whose concept models have a high degree of similarity with the relevant word models. Initial aspects of this work can be found in McCulloch *et al.* (2015).

Table 23.12: Comparison results.

Restaurants	Similarity
HM	SW
WS	0.3998
HM	WS
	0.3561

23.3.12. Summary

In this case study we have seen how both word and concept T1FS models can be created using interval valued survey data and the IAA, and how similarity measures enable us to measure the association between adjectives describing concepts specific to pubs and the perception of those concepts in the context of a particular pub. In other words, the approach enables the generation of user (group) specific ratings of venues or other things (cars, employers, etc.). While the final outputs can be used directly, for example, to generate a linguistic summary for a venue, the process itself generates a wealth of information. This includes detailed T1FS-based models which illustrate the interpretation of a given word/concept in a given context as perceived by a given group of users at the time of the data capture (here, at the time of the survey).

Following this, we saw how multi-iteration survey data can be employed to generate GT2FSs that allow us to compare restaurants. Again, the process itself generates interesting models which highlight the variation in the responses both on the intra-user as well as the inter-user level. We have shown how GT2FSs uniquely allow the capturing and modeling of the stability of perceptions (over repeated surveys) and how an extended Jaccard similarity measure (McCulloch *et al.*, 2013) can be employed to compare the resulting models.

We stress that *none* of the data used in this article are representative of the actual perception of any of the venues by the Nottingham city community (or any other community). The surveys conducted included only a relatively small, demographically non-balanced set of participants. The example's sole aim is the demonstration of how the selected methods can be applied.

23.4. Conclusions

In this chapter, we have demonstrated two approaches that employ CI techniques in human decision-making problems. We have seen how parts of expert decision-making processes can be modeled using a combination of EAs and OWAs, and how the resulting OWAs can be used to inform about the importance of specific sub-components of an overall decision. It has also been shown that the perceptions of survey participants can be modelled using T1 and GT2FSs through the IAA, and a similarity measure can be used with the resulting models can be used to create profiles of objects of interest (e.g., pubs) and make recommendations based upon a user's preferences.

Clearly, the scope for using CI techniques in decision-making problems is vast, here we have given just two specific (and related) examples. In reality, Fuzzy Logic, Evolutionary Computing, Artificial Neural Networks, and combinations of them are used in a multitude of projects that aim to aid the process of decision making.

References

- Bustince, H. (2000). Indicator of inclusion grade for interval-valued fuzzy sets. Application to approximate reasoning based on interval-valued fuzzy sets. *Int. J. of Approx. Reason.* 23(3), pp. 137–209.
- Coupland, S., Mendel, J. and Wu, D. (2010). Enhanced interval approach for encoding words into interval type-2 fuzzy sets and convergence of the word FOUs. In *2010 IEEE Int. Conf. on Fuzzy Systems (FUZZ)*, pp. 1–8.
- Gorzalczany, M. (1987). A method of inference in approximate reasoning based on interval-valued fuzzy sets. *Fuzzy Sets and Systems*, 21(1), pp. 1–17.
- Jaccard, P. (1908). Nouvelles recherches sur la distribution florale, *Bulletin de la Societe de Vaud de Sciences Naturelles*, 44, p. 223.
- Liu, F. and Mendel, J. (2007). An interval approach to fuzzistics for interval type-2 fuzzy sets. In *Fuzzy Systems Conference. FUZZ-IEEE 2007. IEEE International*, pp. 1–6.
- McCulloch, J., Wagner, C., Bachour, K. and Rodden, T. (2015) Give me what I want”—Enabling Complex Queries on Rich Multi-Attribute Data, to appear in Fuzzy Systems (FUZZ-IEEE), *IEEE Int. Conf.*, pp. 1–8.
- McCulloch, J., Wagner, C. and Aickelin, U. (2013). Extending similarity measures of interval type-2 fuzzy sets to general type-2 fuzzy sets. In *Submitted to the 2013 IEEE Int. Conf. Fuzzy Syst.* (Hyderabad, India.).
- Mendel, J. and Wu, D. (2010). *Perceptual Computing: Aiding people in making subjective judgments*, IEEE Press Series on Computational Intelligence (Wiley).
- Miller, S., Wagner, C. and Garibaldi, J. (2012). Constructing general type-2 fuzzy sets from interval-valued data. In *IEEE Int. Conf. Fuzzy* (Brisbane, Australia).
- Mitchell, H. (2005). Pattern recognition using type-ii fuzzy sets. *Inform. Sci.*, 170(2), pp. 409–418.
- Nguyen, H. and Kreinovich, V. (2008). Computing degrees of subsethood and similarity for interval-valued fuzzy sets: Fast algorithms. In *Proceedings of the 9th International Conference on Intelligent Technologies (InTech'08)*. Samui, Thailand, pp. 47–55.
- Wagner, C. and Hagras, H. (2010). Toward General Type-2 Fuzzy Logic Systems Based on zSlices. *Fuzzy Systems, IEEE Transactions*, 18(4), pp. 637–660.
- Wagner, C., Miller, S., Garibaldi, J. M., Anderson, D. T. and Havens, T. C. (2015). From interval-valued data to general type-2 Fuzzy Sets. *Fuzzy Systems, IEEE Transactions*, 23(2), pp. 248–269.
- Wu, D. and Mendel, J. (2008). A vector similarity measure for linguistic approximation: Interval type-2 and type-1 fuzzy sets. *Inform. Sci.*, 178(2), pp. 381–402.
- Wu, D. and Mendel, J. (2009). A comparative study of ranking methods, similarity measures and uncertainty measures for interval type-2 fuzzy sets. *Inform. Sci.*, 179(8), pp. 1169–1192.
- Yager, R. (1988). On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *T. Syst. Man Cyb., IEEE*, 18(1), pp. 183–190.
- Yang, M. and Lin, D. (2009). On similarity and inclusion measures between type-2 fuzzy sets with an application to clustering. *Comput. Math. Appl.*, 57(6), pp. 896–907.
- Zeng, W. and Li, H. (2006). Relationship between similarity measure and entropy of interval valued fuzzy sets. *Fuzzy Set. and Syst.*, 157(11), pp. 1477–1484.

Chapter 24

Applications of Computational Intelligence to Process Industry

Jose Macias Hernández

The process industry and, more specifically, chemical engineering processes have great opportunities to develop, test, and extensively implement tools with smart computing features. The complexity of the processes, the variety of interaction mechanisms that occur in these processes, and the infinite possibilities of design, in addition to failure testing, optimization, production planning, etc., can cause a myriad of computational intelligence (CI) applications to be possible. In this paper, we aim to review all these possibilities by listing some of the opportunities and developments, also to give some examples of problem solving. Potential opportunities for the use of CI can be identified throughout the design sequence of an industrial plant, from the conceptual design process to startup and troubleshooting. Similarly within an operating plant an identification of CI applications can be made if we analyze the structure of a facility by job function and from there select those areas or issues which also require a contribution to solving problems.

24.1. Introduction

Computational intelligence (CI) should first be defined. Terms such as “artificial intelligence”, “computer intelligence” and “soft computing” are often confused. Although it is a difficult problem and is subject to a great deal of debate, one of the best ways we can define this particular family of intelligent systems is by using a definition from Krishnakumar (2003) that clearly defines the term.

“From the perspective of computation, the intelligence of a system can be characterized by its flexibility, adaptability, memory, learning, temporal dynamics, reasoning, and the ability to manage uncertain and imprecise information”.

Some examples of tools that comply with this definition are neural nets, genetic algorithms, fuzzy systems, evolutionary programming, genetic algorithms, and evolutionary programming.

Early definition of process industrial intelligent applications is shown in Mc Avoy (2002) where partial least squares (PLS) and Expert Systems are used. Same approach can be seen in Macias (2001) and Macias *et al.* (2001). In this paper, we will focus on the evolving and fuzzy capabilities of CI.

Once the tool has been defined it is necessary to show the application. One such application is within the Process Industry. Here, we find very complex activities that rely strongly upon personal knowledge, processes, and technology. There is plenty of room to use these tools in the Process Industry.

The success of a process in the industry starts from the initial stage of development of the conceptual design and continues throughout the life of the plant until its retirement. It is therefore necessary to develop new products from raw materials that allow different processes to adapt to the needs of increasingly competitive markets, more stringent environmental restrictions, and tighter costs. This makes the use of process tools and simulation data in turn increasingly complex. Information processing must take into account the multiple interactions between data, and also the imprecise and variable behavior that the data has over time. That is why possibly intelligent tools such as fuzzy data processing are a possible solution to these increasingly complex problems.

An example is the Oil and Gas (O&G) sector which is one of the most complex process industries. In these processes, Intelligent Systems can play an important role to maintain profitability in this very economically competitive world. Process industries in general and O&G in particular require:

- Constant product supervision (high number of process variables).
- Cost reduction (optimization).
- Continuous Innovation for new products, processes (conceptual design tools).

To cope with these requirements the application of Intelligent Computing needs a huge amount of data. Nowadays, in process industries there is a large amount of data available.

There are several reviews in the literature that describe the uses of intelligent technologies in process industries. One of them, Kahraman (ed.) (2012) makes this analysis from the technological point of view. This is by grouping applications by computing intelligent tools.

In this approach, we are considering the problem itself and seeing which technology could be more suitable to solve it.

24.2. Information Systems

Every Process Plant has a supervisory system in place. Even the smallest plant has a “PLC” (Programmable Logic Controller) and a “SCADA” (Supervisory Control and Data Acquisition). These systems generate huge volumes of data that are processed and stored in sophisticated Information Systems.

The operator of a typical Distributed Control System (“DCS”), in a process industry, controls and monitors several process units and is responsible for more than 400 valves. The instrumentation database can have as many as 10,000– 12,000 continuous data points. Laboratory samples are routinely analyzed and more than 200–2,000 characteristics could be reported every day. The process operators continuously make decisions based on previous experience to drive the process towards targets. However, with “soft” sensors this process could be automated.

Data generated by “DCS” are normally of high-resolution (snap values) with a limited time frame. In the majority of cases there is low-resolution (average data values) with probably several years of data history.

This data feeds several corporate applications for data reconciliation (mass balance), logistics, and product movements, etc. As a result, it is possible to monitor the complete process flow and to analyze the plant performance and give full support to departments such as planning, process accounting, and operations.

There are, although, several problems associated with this kind of data, the most important are the high correlation and noise.

Because of high data correlation, the amount of useful information that is contained is much lower. In addition to that the problems and events associated with a running process plant make it very difficult to diagnose a problem in the operation or in the performance of the plant. Therefore, deep understanding of the process unit is challenging.

As a result, troubleshooting of a process unit when the problem has fully developed is more frequent than early diagnosis.

Also, the operation of the plant is normally set around process values and because of this short operational window the magnitude of the noise is almost as high as the magnitude of the process variations. Care should be taken to design models with high information content and robustness.

Definitely, it is necessary to have more tools to manipulate the data and convert it to information, so it can be used to help in diagnostics and process follow-ups.

If information can be successfully extracted from data, it can be made available to Operators, Process Engineers, Control Engineers, and Managers so they can better understand their respective plant processes and therefore make the necessary actions to optimize and lower cost. The “eTS+” (Angelov and Zhao, 2006) approach can probably be

used among others to effectively extract this valuable information.

To illustrate this concept the classical control pyramid with the relation between decision level and data volume can be used (see [Figure 24.1](#)). The greater the time frame, the greater the economic impact is or the more important a decision, the greater the application scope. Also, the volume of data gets reduced where it is converted to information.



Figure 24.1: Time frame and its relation to the decision level and application scope.

24.2.1. Data Reconciliation

A very important characteristic of process data, for subsequent analysis, is that it should be balanced; i.e., the material balances (mass components) and energy balances are met. The reconciliation data is commonly done using optimization software to reduce measurement error. The cumulative error (observed value and reconciled value) of all instrumentation is used as an objective function, along with some appropriate criteria. Recently, by using fuzzy logic, this process can be simplified significantly as Dr. Tan discusses in (Tan *et al.*, 2009). In Dr. Tan's paper, he writes how it is possible to obtain the value reconciled with a simple approximation of the observed value with the triangular or trapezoidal function and subsequent arithmetic fuzzy logic.

24.3. Raw Materials

Profitability starts when purchasing raw materials. This is most of the time true, because the price of raw materials is one of the most contributing factors to final product price. In the O&G industry, for instance, the price of raw material (crude oil) is around 90–95% of the final retail price. Therefore, a bad price of crude oil or purchasing by mistake could damage a Refinery's bottom line. Whereas, purchasing well-balanced crude oil at a good price can make more money than the refining activity itself. There is a vast variety of crude oil with different properties that will give different cuts and yields when processing. The price of crude oil depends on several factors, ranging from its intrinsic chemical properties to political stability in the country of supply, and short- and long-term market behavior. Supply chain, trading departments have to balance “spot market” contracts and “long term” contracts to maintain a stable crude oil supply to their refineries.

How do you go about selecting the right crude oil and how do you fix the break-even price on crude purchasing? The crude oil cuts should be balanced to account for the market share of the refinery's products, therefore only producing what you are able to sell. Any unbalance is normally undesirable. There are tools designed to help do this, but problems arise if their predictions are not correct. The real problem is to evaluate crude oil selection with actual plant performance. Again, there is a huge amount of data which is highly collinear.

24.4. Application Opportunities

A process plant is a very complex installation because of the number of interconnected process operations (pumping, distillation, reaction, heating, etc.). Also, it is because of work organization. If we make the effort to analyze the work flow in a typical installation we can identify different possible implementations of intelligent technologies and those will not be limited to instrument data mining.

The majority of these areas can be found in any complex industrial installation.

24.5. The Process of Designing a Design Process

The design process starts whenever a product demand or service is created.

24.5.1. *Process Integration*

Once you have created the need to produce a particular product from certain raw materials, the first step of the design is the conceptual design. This methodology consists of a series of decisions that are based on rules that define the design of the plant as defined by Douglas (1988) and Smith (2005). These rules are rigid and do not have any fuzzy treatment. These rules, on the whole, are the result of the combination of past experiences in processes, raw materials, and product properties. In some stages of conceptual design, such as the energy integration level, some interesting contributions have been made by the application of intelligent tools. But, in general, there are many opportunities to develop applications in this field. The last part of the design (energy integration) is when the number of design alternatives is complicated and it is necessary to use optimization rules and more complex evaluation. Recently there has been an initiative to incorporate fuzzy arithmetic to the analysis of heat exchange networks (Hussein *et al.*, 2013).

Similarly, the impact assessment of the life cycle (LCA) has references on the use of fuzzy tools for evaluation of alternatives (Tan, 2005). In this paper a model of decision support tools based on fuzzy is proposed. The subject is the selection of motor fuels.

24.5.2. *Process Engineering*

For the design, process engineers use simulation tools that use rigorous mathematical formulation of the basic operations of chemical engineering and thermodynamics. Besides the steady state, engineers use dynamic models that provide insight into the best control strategies and design virtual plants that allow the training of plant operators even before they have been built. Neural networks and neuro-fuzzy models have been used to speed up the computing capacity and more simple and robust models as seen in Macias *et al.* (2001).

24.6. Planning and Scheduling

The complexity and the opportunities of CI come from the very beginning with the purchase of raw materials. As shown before, for instance, in an Oil Refinery there is a tradeoff between quality, yields, spot and long-term deals, and international polices.

These problems are complex, nonlinear, and non-stationary and a model to develop should take into account all this information which is highly uncertain (ERTC, 2004; Goel and Prashant, 1999). Some companies reported commercial products that address these problems e.g., Aspentech, GenSym, but clearly there is an opportunity for more advanced solutions in this area.

Also, backcasting planning operations are necessary to correlate better prediction models. None of the commercial softwares available today can do this using an intelligent application that automatically identifies opportunities and evolves itself.

24.7. Operations

Most industrial plants are very complex in nature. From the processing of raw materials to finished products there are not only several, but hundreds of process operations. For every process operation, it is also necessary to consider raw material properties. Both, operations and process engineering departments have to cope with these problems. Clearly, the problems associated with it are difficult to analyze in real time. They have to deal with short-term process troubleshooting, security, and quality management. Process operators have to deal with problems associated with process security, stability, quality, and optimization targets of the plant. If we think of automatic data processing and diagnosis, there definitely are quite a few application opportunities of Evolving Intelligent System technologies in the field of process operation, like:

24.7.1. Operator Advisory Applications

Advisory Systems are required to help operators with real-time decision-making processes to maintain security, stability and, process optimization (Macías, 2001). Intelligent Systems can be very useful for tackling these problems.

24.7.2. Alarm Monitoring and Rationalization

The main goal here is to identify scenarios and analyze the sequence of events to determine the initial detection agent. Once the agent is identified, rationalization means to suppress any superfluous alarm that does not add new information but reduces the capability of the operator to react safely. Tools that can be useful and efficient are eClustering+ and eTS+. For instance, (Geng *et al.*, 2005) proposes a new system alarm optimization technique, based on a fuzzy clustering-ranking (“FCR”) algorithm, according to the correlativity among process-measured variables.

24.7.3. Emergency Manager

Emergency Manager is related to alarm monitoring and rationalization, but here, the goal is to help the operator to manage emergency situations. In Foong *et al.* (2010), the author proposes a Decision Support System for Alarm Rationalization using Risk Assessment Matrix (“ARRAM”). This method uses crisp rules although the author intends future work to be done on fuzzy data processing.

24.8. Maintenance

The cost of a mechanical failure or breakage should include the interruption of production, material outside of specifications, safety issues, etc. Therefore, early detection of potential failure or failure is a very important predictive maintenance program task. It is not always possible in a direct way to determine if a fault is occurring or can occur. On many occasions this detection can be inferred from the interpretation of the data (vibration analysis, mean time between failures (“MTBF”), process variables, indicators of corrosion, etc.). Therefore, detecting what, how and when it will happen is a very good opportunity to use smart computing tools. An early detection failure sensor is, in general, an open problem. There are successful applications where this is the case, like vibration analysis, but cross correlation between process operating data and equipment failure is not done in an automatic fashion. eClustering+ and eTS+ are very promising tools for this problem (Filev and Tseng, 2006).

24.9. Process Control

There are lots of references of using CI for Process Control. A review can be found in Precup and Hellendoorn (2010). Control Engineering has several general open issues:

24.9.1. Software Sensors

This is an area of very successful soft sensor applications (Barsamian and Macías, 1998; Fortuna *et al.*, 2007; Baht *et al.*, 2003, etc.). Most of them, however, are based on offline black-box type models. They are not able to re-calibrate, neither to provide an insight for the process (are not transparent), do not take into account environmental dynamics and internal changes in the plant (different operating modes, ageing, contamination), etc. As a result, the life cycle costs of conventional soft sensors are high.

A review of Bayesian methods can be found in Khatibisepehr *et al.* (2013) for use both in “steady state” and in “dynamic” mode.

24.9.2. Dynamic Model Identification

The classical control theory provides a solid theoretical basis for the linear algebra systems and a limited number of nonlinear systems. Data mining is suitable for offline cases, while dynamic model identification is a very promising area, but for the general nonlinear case there are practically no ready solutions apart from computing intelligence.

24.9.3. Detecting Drift and Shift in the Data Pattern in Streaming Data

An interesting article can be found in Angelov and Zhou (2008).

24.9.4. Controller Tuning

There are references using fuzzy logic to tune the controllers (Gaddam and Rajani, 2006). The main benefits of using Fuzzy Logic appear when process nonlinearities such as variable saturation are significant. A balance is obtained between both rise time and overshoot in response with this implementation. In this implementation a two degree parameter is sufficient to tune the controller. (Kratmüller, 2009), remarks that Fuzzy logic controllers are in general considered being applicable to plants that are mathematically poorly understood and where the experienced human operators are available. In indirect adaptive fuzzy control, the fuzzy logic systems are used to model the plant assuming that the fuzzy logic system represents the true plant.

The problem of detecting internal loop problems versus external influences can be addressed by eClustering+ and eClass.

24.9.5. Hybrid controllers (Multivariable Predictive Control, MPC + Fuzzy techniques)

MPC tools are mature; however, their utilization depends on a properly designed controller and the selection of variables. The first part is usually done in practice using Lyapunov theory, LMI, etc., by the design engineer. The variable selection is an important task which can be done automatically. However, a non-stationary system requires continuous model update. Fuzzy logic also arrived in these very popular industrial control systems. This paper (Sarimveis, 2003) introduces a new fuzzy control technique that modifies the general idea of linear MPC. The method is based on a dynamic fuzzy model of the process to be controlled, which is used for predicting the future behavior of the output variables. The method can be used with any type of fuzzy model and is particularly useful when a direct fuzzy controller cannot be designed due to the complexity of the process and the difficulty in developing fuzzy control rules. The method is illustrated via the application of a nonlinear single-input single-output reactor, where a Takagi–Sugeno model serves as a predictor of the processes future behavior.

Table 24.1: Potential application areas for the use of Intelligent Systems in process industry.

Departments/Areas	Application Type
Operations	Alarm monitoring, operator advisory, emergency management
Price prediction	Regression and time-series prediction models, e.g., eTS+
Process Engineering	Process diagnostics, e.g., eClustering+
Information Systems	Mining data streams, e.g., eTS+
Predictive Maintenance	Machine health monitoring and prognostics, e.g., eClustering+ and eTS+ (Filev and Tseng, 2006)
Control Engineering	eControl, Guided rule-base adaptation (Filev <i>et al.</i> , 2000)
Soft Sensors	eSensor

In the [Table 24.1](#), a summary of the application areas of Intelligent Systems is shown in relation to the Department or Areas that normally are of interest in an industry.

24.10. Process Industry, Oil Refinery Case

Once we have analyzed the different functions in a process facility (maintenance, engineering, process control, purchasing, etc.), it is also important to review a typical installation to identify opportunities for using CI in each process plant. An Oil Refinery can serve as an illustrative example of the use of smart technologies in process industries. A refinery is an industrial facility where crude oil is separated into its fraction and then, these fractions are treated to produce other intermediate products with a higher end value. Intermediate products will be blended to produce final products. These blends should be combined in the right proportions to comply with commercial specifications, producing as much final product with the lowest production cost possible (see [Figure 24.2](#)).

The sale of final products has to pay for the raw material and all the associated costs. Because of the high raw material cost, the petroleum refining process does not have a high economic return in terms of the percentage of the capital employed, but receives profit from the huge volume of crude oil processed.

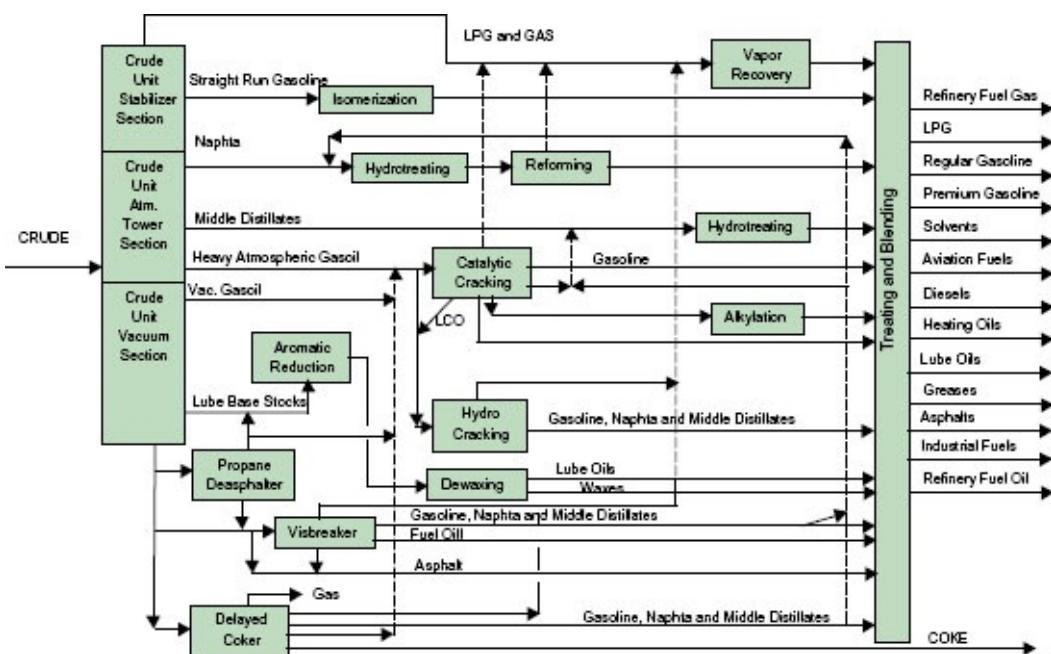


Figure 24.2: Typical oil refinery flow sheet.

Today, there is very little room to improve economic performance apart from reducing costs and improving the transformation processes.

To support these arguments, we should describe the refining process. Those areas in which the uses of smart technologies provide value will also be discussed.

24.11. Raw Material

The most interesting qualities of oil (other than product yield), which are present and which have the greatest impact on refinery economics are the sulfur content, density, parafinicity, and the residue carbon, salts, nitrogen and metals content. In analysis, crude oil qualities and their impact on production cost can explain part of the process of crude oil price construction.

The sulfur content impacts the economy of a refinery because the use of desulfurization units for most oil fractions is required. Acids or high-sulfur crude oil, where more sulfur has to be removed are cheaper than sweet or low-sulfur crude oil.

Low density crude oil correlate with lighter crude oil which produces much smaller amounts of residue, which is one of the fractions with the smallest value. Crude Oil is more expensive than the residue; therefore, this amount must be reduced to a minimum in any refinery.

The transformation of the naphtha into gasoline requires increasing the aromaticity of the oil fraction which uses energy and other manufacturing processes. Naphthenic naphtha converts to aromatics much more easily than its brother paraffinic naphtha, reducing the energy and other manufacturing processes needed. Consequently, the market cost of oil with a higher paraffinic content is cheaper.

The Carbon Residue is related to asphalt content, which is the cheapest fraction.

Salt produces corrosion in the process units. Therefore, the higher the salt content, the cheaper the crude oil.

Nitrogen and sulfur cause catalyst poisoning and must be removed and metals also produce catalyst poisoning and severe deactivation. Because of the high price of the catalyst employed in refinery process units, catalyst deactivation impacts very heavily on the economic performance of the refinery.

24.12. Process Units

The opportunities for the use of CI technology in process units can be analyzed mainly from the point of view of quality monitoring and control. Because there are a high number of processes, we select only the most typical ones. The same ideas can be extrapolated to others.

The principal operations required in a typical refinery are:

- Crude oil distillation.
- Reforming.
- Hydrotreating.
- FCC.
- Visbreaking.

24.13. Crude Oil Distillation

24.13.1. Process Description

It is the first unit in the refinery flow sheet (see [Figure 24.3](#)); its mission is to separate by distillation the different crude oil fractions by means of boiling the crude oil and then condensing it. Steam is added to improve the separation. The fractions extracted by side streams in the tower are:

- Naphtha.
- Kerosene.
- Gas oil.
- Fuel Oil.

Fuel Oil can be fed from the main distillation tower into a secondary “vacuum distillation tower”, if available, where asphalt or vacuum fuel oil is produced.

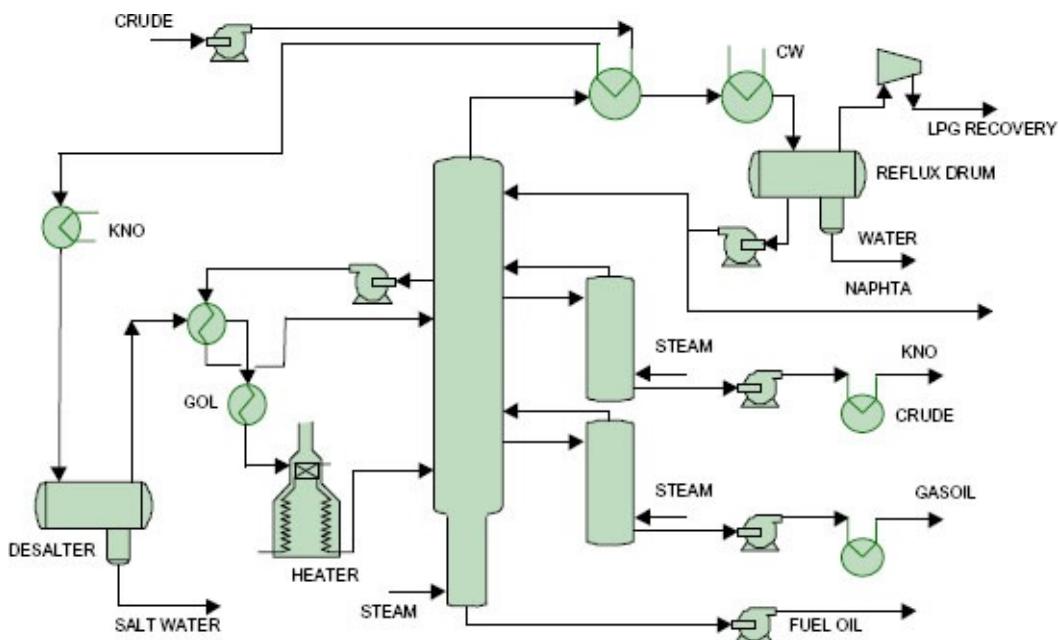


Figure 24.3: A reduced schematic of a crude oil distillation unit.

24.13.2. Process Key Variables

The most important key operating variables are:

- Furnace temperature.
- Top reflux, temperature, and pressure.
- Side tower heat removal (cooling).
- Side stream flows.

These variables produce different ranges of side stream qualities. These qualities are important and have to be correct and stable because there are no more opportunities to

correct them. If any product is left above or below the side stream it will be redirected to a different final product and this effect will impact the refinery economics.

24.13.3. Opportunities for CI

The opportunities of CI in quality monitoring are mainly through the use of “**soft sensors**”. For example, “soft” analyzers for side stream cut point calculations instead of real distillation analyzers that are expensive and highly maintenance-dependent. The currently used technologies are offline NN (neural networks), but there are successful applications using PLS and bias updates (Macías, 1998). This latter is especially important for close loop “soft” control due to its robustness.

Process units are pushed to their maximum capacity and in that region problems of “**flooding**” in the tower can arise. Therefore, there is a need to diagnose and maintain the distillation tower to its limits and “soft” fuzzy sensors will help for that tower loading calculations.

Another interesting application is the development of a fuzzy rule-based model for unit optimization to solve problems like:

(1) **Crude switch:** Any time there is a change in the crude oil supply to a distillation unit, the unit should be adjusted to maintain side stream qualities. The transient response of the unit is different to the one that happens in normal operation. Therefore, the transient should be detected and corrected in advance so the time that a product is off specification is reduced to a minimum. Detecting, evaluating and coping with this situation, represents an unsolved problem for refineries.

(2) **Feed properties and operation conditions:** The crude oil tower behaves differently from one crude oil type to another. The unit should be adjusted to the new optimum settings. This is typically accomplished by analyzing the process database to find the best operating conditions of the unit or using case-based reasoning. An alternative is to use CI for online updates in new conditions. A good example of a CI is the guided rule-based initialization applied to advanced manufacturing (Filev *et al.*, 2000).

24.14. Hydro Treating

24.14.1. Process Description

Hydro-treaters treat a feed (naphtha, kerosene, or gas oil) where hydrogen is added to convert the sulfur combined with the hydrocarbons into SH_2 (see [Figure 24.4](#)). The SH_2 gets partially dissolved in the liquid and will have to be removed in a stripper tower. This unit is used for treatment of naphtha, kerosene, and gas oil. The reaction takes place in presence of a catalyst at high temperature and pressure. The reactor pressure will depend on the hydrocarbon type and hydrogen purity, the higher the hydrocarbon boiling point, the higher the operating pressure. Metals must be removed before hand because they poison the catalyst. During the process, coke is formed due to cracking. Hydrogen is also used to minimize cracking in the reactor.

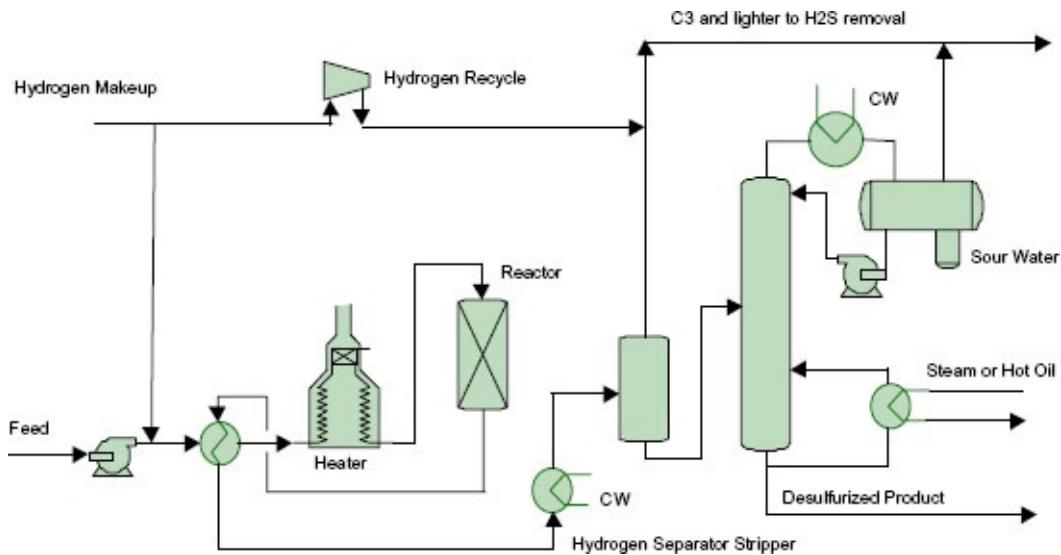


Figure 24.4: Hydro-treating unit.

24.14.2. Process Key Variables

The key variables that control the severity of the reaction are:

- Reactor temperature.
- Hydrogen partial pressure.
- Spatial velocity.

The partial pressure in the reactor is maintained with the separator pressure and the hydrogen recycling. Hydrogen benefits the RUN length or the time that the unit can be in service between having to regenerate the catalyst. The temperature or steam flow in stripper bottoms is controlled to remove the SH_2 dissolved in the reactor effluent.

24.14.3. Opportunities

Reactor models are only a mathematical approximation of the process and should be adjusted to process variables. Also, the **activity of the catalyst changes** and therefore a

long-term adjustment should be made. Modeling reactors have a calibration problem subject to **time variations**. This important adjustment problem calls for self-calibration models (evolving). The difficulty arises due to the fact that the feed is not perfectly characterized. This requires the use of fuzzy sets. eTS+ are thus, perfectly placed to be used as a backbone technology for the development of advanced evolving “soft” sensors (eSensors) to address this problem.

These evolving models can be of utility for integrated simulation models for monitoring and reaction control.

24.15. Catalytic Reforming Unit

24.15.1. Process Description

Reforming is a catalytic process designed to increase the aromatic and isoparafinic content of naphtha and therefore produces high-octane gasoline. The product is called reformate and because of its high aromatic content, it is also used to produce aromatics as a raw material for petrochemical facilities.

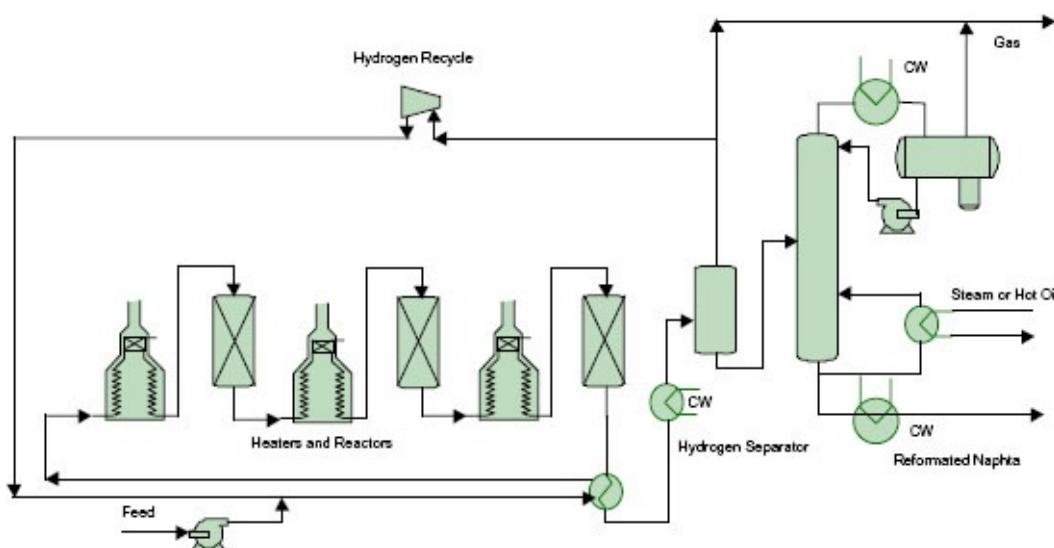


Figure 24.5: Catalytic reforming unit.

This chemical process takes place in several reactors in series using hydrogen to protect the catalyst preventing the poisoning from coke laydown. Hydrogen is produced also in the process unit due to dehydrogenation reactions. Some cracking reactions also take place producing liquefied petroleum gas (LPG) and refinery gas, (see [Figure 24.5](#)).

24.15.2. Process Key Variables

The key variables that control the severity of the reaction are, among others:

- Reactor temperature.
- Hydrogen partial pressure.
- Spatial velocity.
- Water/chlorine balance in the catalyst.

The temperature and spatial velocity are the most important short term key variables, while water/chlorine balance has long-term contribution. A post-fractionation unit separates the LPG and gas from reformatted naphtha. The operation variables in that tower are the **reflux ratio** and the **bottoms temperature**. Quality control variables are the octane number and the Reid vapor pressure of the reformatted naphtha. The higher the octane number, the higher aromatic content of reformate.

24.15.3. Opportunities

Like any process unit with catalysts, the activity of the unit will depend on the type of catalysts and also on the time that the unit works continuously. Therefore, CI models should cope with variations in catalyst activity to predict qualities like octane number and product yields. Such a model is much more complicated than the hydro-treating one due to the fact that the number of reactions is much higher.

Ideally, it should be able to re-calibrate and re-train automatically. It is not frequent to have octane analyzers in the bottom of the fractionation tower, so a CI-based soft sensor can be designed based upon operating conditions and type of feed. This latter characteristic should be derived upstream in the crude oil unit and naphtha hydro-treating units.

24.16. Catalytic Cracking

24.16.1. Process Description

Fluid catalytic cracking is a technology to break the vacuum gas oil into gasoline and lighter fractions. Due to the very high speed of catalyst deactivation and as the consequence of the severe operating conditions, the regeneration process takes place simultaneously in a second regenerator reactor. The catalyst moves from the reactor to the regenerator. The catalyst coke formed is burned in the regenerator using air (see [Figure 24.6](#)).

24.16.2. Process Key Variables

With a fixed type of feedstock, the major operating variables that affect the conversion and product distribution are;

- Cracking temperature.
- Catalyst to feed ratio.
- Space velocity.
- Type of catalyst.
- Recycle ratio, air.

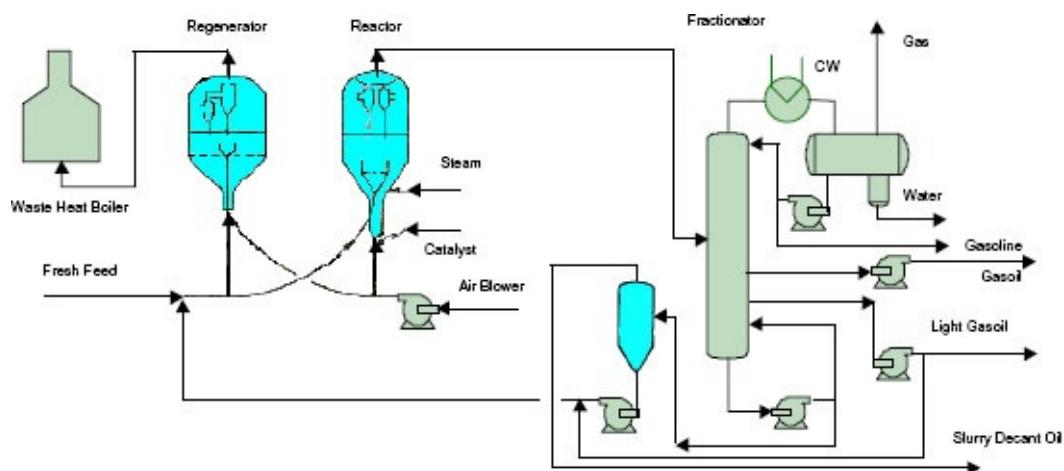


Figure 24.6: Fluid catalytic cracking unit.

To increase the conversion, we must increase reaction temperature, catalyst to feed ratio, catalyst activity, and reaction time. The limitations come from the limit on burning activity, limit in the air compressor, and temperature limitation in the regenerator. Sometimes, oxygen is added to increase the burning capability of the regenerator reactor.

24.16.3. Opportunities

Because of the high added-value of this process unit to a Refinery, it is very frequent to see models to optimize the unit performance. This is an important opportunity to use CI-based “soft” analyzers to calculate the octane number of the gasoline and product

distribution. There is also a case for **hybrid models** to simulate the reaction and regeneration sections in a mixed approach using rigorous simulation in the main fractionation and to balance catalyst activity with time and severity using e.g., eTS+.

A difficult and challenging task is to early detect and identify different types of feed that facilitates a faster move to unit limits. There is a large scope to address these using techniques such as eClustering+, for example. In recent articles, it is hard to see real examples of the use of CI. An example of the combination of process control (MPC approach) and process simulation can be found in Mihaela and Agachi (2010). Another paper, also using MPC and soft sensor can be seen in Yang *et al.* (1998). In none of these papers is there a fuzzy approximation for soft sensors either in steady state or in dynamic mode. This is definitely an open issue.

24.17. Thermal Cracking

Visbreaking of soaker type

24.17.1. Process Description

Visbreaking is a type of thermal cracking that reduces the viscosity of a heavier feed to a lower viscosity fuel by means of breaking higher molecular weight molecules into gasoline, kerosene, gas oil, and lower viscosity fuel. The reaction takes place by temperature and space velocity in the furnace (coil visbreakers) or a soaker drum (soaker visbreaker). The former achieves conversion by temperature, the latter by space velocity. As a result, soaker drum visbreakers get longer run times than coil visbreakers. The reactor effluent is separated in fractions in a fractionation and stabilizer towers. Run length is controlled by coke deposition in the furnace that limits skin temperature.

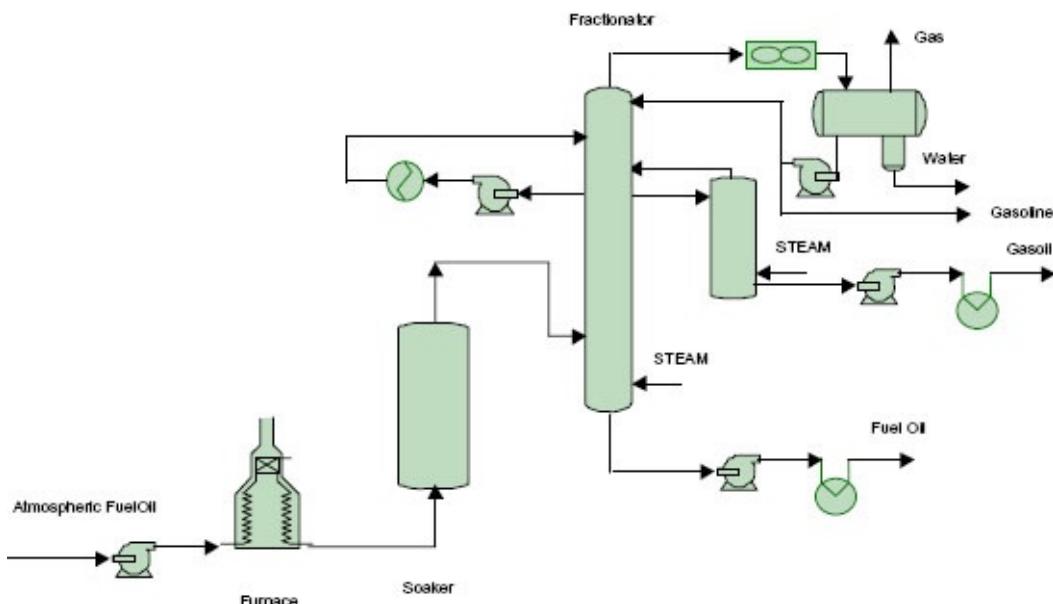


Figure 24.7: Visbreaking unit, soaker type.

24.17.2. Process Key Variables

The key process variables are the temperature and the spatial velocity. There are other operating variables in the fractionation tower like steam, reflux ratio, and heat distribution. The key variable is the stability of the fuel oil, which is drawn from the fractionation bottoms.

A big practical obstacle is the lack of analyzers for stability. Manipulating extraction side streams and stripping steam should control side streams cut point. Some types of feed are more sensitive to coke lay down than others. Prediction of coke lay down is important to optimize the run length.

24.17.3. CI Opportunities

There are great opportunities for CI-based “soft” analyzers:

- Predicting the stability of the fuel oil, profit key variable in this process unit.
- Soft Analyzers for cut point side stream control.
- Hybrid integrated models (vis-breaking reaction model and rigorous steady state tower simulation).
- There is a need for feed characterization for its crack ability and stability to predict coke laydown, run length, and maximum severity.
- Use this information to manipulate operating variables between feed switch.

24.18. Conclusion

Process Industry is normally very complex. There are a high number of process variables and lab samples and therefore a huge database. As a result there are a high number of opportunities for the implementation of Intelligent Systems.

CI can be applied to several fields ranging from operations or process engineering to maintenance, accounting, and supply chain. This increases the complexity of any application due to the fact that it has to account for the implication of different Departments of a process industry.

The information available in the oil refinery has several inherent characteristics: a high number of variables in a small range of variations, highly collinear and noisy data. Most of the time there is no clear relation between the data and type of feed. These factors increase uncertainty and noise.

The tools that are most commonly used include “soft” analyzer, model building, and time series predictors. The use of fuzzy rule-based systems is suitable for operator advisory systems in normal and emergency situations. Alarm management tools can be a special case of this application.

Control engineering is one of the activities with the highest number of opportunities for implementation of CI. It is very well suited for hybrid control, mixing between conventional control theory and fuzzy systems.

Even though there are a lot of tools already available in the market there are still so many unsolved problems that will make eTS+ and other CI interesting for the petrochemical and other process industries in the future.

References

- Angelov, P. and Zhou, X. (2006). Evolving fuzzy systems from data streams in real-time. In *Proc. 2006 International Symposium on Evolving Fuzzy Syst.* Ambleside, UK, pp. 29–35.
- Angelov, P. and Zhou, X. (2008). On line learning fuzzy rule-based system structure from data streams. In *Proc. World Congress on Computational Intelligence*. Hong Kong.
- Baht, S., Chatterjee, D. and Saraf, N. (2003). On line data processing and product properties prediction for crude distillation units. *AIChE 2003 Spring National Meeting*. New Orleans, Luisiana, USA.
- Barsamian, A. and Macías, J. (1998). Neural Network-based inferential property predictors. *Hydrocarb. Process.*, 77(10).
- Douglas, J. (1988), *Conceptual Design of Chemical Processes*. New York: McGraw-Hill.
- Filev D. and Tseng, T. (2006). Novelty detection-based machine health prognostics, In *Proc. 2006 International Symposium on Evolving Fuzzy Systems*, IEEE Press, pp. 193–199.
- Foong *et al.* (2010). Decision support system for alarm rationalization using risk assessment matrix. *Int. J. Comput. Appl.*, (0975–8887), 4(9).
- Fortuna L., Graziani, S., Rizzo, A, and Xibilia, M. G. (2007). Soft sensor for monitoring and control of industrial processes. In Grimble, M. J. and Johnson, M. A. (eds.), *Advances in Industrial Control Series*. Berlin, Germany: Springer-Verlag.
- Gaddam, M. and Rajani, A. (2006). Automatic tuning of pid controller using fuzzy logic, *International Conference on development and application system. Suceava, Romania*.
- Geng, Z. *et al.* (2005). A fuzzy clustering-ranking algorithm and its application for alarm operating optimization in chemical processing. *Process Saf. Prog.*, 24(1), pp. 66–75.
- Hussein, M.H. *et al.* (2013). Fuzzy analogical gates approach for heat exchangers networks. *Int. J. Comput. Appl.*, (0975–8887). 73(21).
- Kahraman, C. (ed.) (2012). *Computational Intelligence Systems in Industrial Engineering*, Atlantis Computational Intelligence Systems 6, Atlantis Press. doi: 10.2991/978-94-91216-77-0_1.
- Khatibisepehr, S. *et al.* (2013). Design of inferential sensors in the process industry: A review of Bayesian methods. *J. Process Contr.*, 23, pp. 1757–1596.
- Kratmüller, M. (2009). Adaptive fuzzy control design. *Acta Polytech. Hung.*, 6(4), pp. 29–49.
- Krishnakumar (2003). Intelligent systems for aerospace engineering—an overview, *NASA Technical Report*. Document ID: 30020105746.
- Macías H. J. (2001). Design of a production expert system to improve refinery process operations. *EUNITE Symp.*, Tenerife, Spain.
- Macías, H. and Feliu J. A. (2001). Dynamic study of inferential sensor (NN) in quality prediction of crude oil distillation tower side streams. 11th *Euro Symp. Comp. Aided Proc. Eng., ESCAPE 11*, Kolding, Denmark.
- Macias, J., Feliu, J. and Bosch, A. Simulation workbench between HYSYS rigorous model and its NN extension: Case study of a complex simulation with re-circulation, 4th *European Congress on Chemical Engineering, ECCE 4*.
- McAvoy, T. (2002). Intellingent control applications in the process industries, *Annu. Rev. Control.* 26, pp. 75–86.
- Mihaela, I. and Agachi, P. S. (2010). Optimal process control and operation of an industrial integrated fluid catalytic cracking plant using model predictive control. 20th *European Symposium on computer Aided Process Engineering —ESCAPE20*, Elsevier B.V.
- Precup, R. and Hellendoorn, H. (2011). A survey of industrial applications of fuzzy control, *Computers in Industry*. 62, pp. 213–226.
- Sarimveis, H. *et al.* (2003). Fuzzy model predictive control of nonlinear processes using genetic algorithms. *Fuzzy Set. Syst.*, 139, pp. 59–80.
- Smith, R. (2005). *Chemical Process: Design and Integration*. Wiley. Takagi, T. and Sugeno, M (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE T. Syst, Man Cyb.*, 15(1), pp. 116–132.

- Tan *et al.* (2005). A fuzzy decision support model for the selection of environment-friendly fuels for road vehicles. *Journal of the Eastern Asia Society for Transportation Studies*. 6, pp. 3264–3275.
- Tan *et al.* (2009). Fuzzy data reconciliation in life cycle inventory analysis. De la Salle University Library.
- Yang, S. H. *et al.* (1998). Soft sensor based predictive control of industrial fluid catalytic cracking processes, *Trans IChemE*, 76, Part A, Institution of chemical Engineers.

Chapter 25

Applications of Computational Intelligence to Robotics and Autonomous Systems

Adham Atyabi and Samia Nefti-Meziani

This chapter discusses the application of computational intelligence (CI) in the field of autonomous mobile robotics. The chapter is focused on basic concepts of computational intelligence in robotic domain with an emphasis on essential aspects of navigation such as localization, path planning, and obstacle avoidance both on single and swarm robots. The chapter reviews recently published studies on these aspects aiming to highlight the recent trends on the field. In addition, the applications of navigation and motion planning on robotic arms, haptic devices, humanoid robots, and robotic hands are discussed.

25.1. Introduction

25.1.1. What is Computational Intelligence (CI)

CI is defined as a set of nature-inspired computational approaches designed to handle complex problems to which classical and traditional approaches are ineffective (Włodzisław, 2007). This ineffectiveness is either due to not being possible to formulate the problem or the problem being NP-hard. Bezdek (1994) offered one of the early definitions for CI as: “*A system is called computationally intelligent if it deals only with numerical (low-level) data, has a pattern recognition component, and does not use knowledge in the AI sense; and additionally, when it (begins to) exhibit (i) computational adaptivity; (ii) computational fault tolerance; (iii) speed approaching human-like turnaround; and (iv) error rates that approximate human performance*”.

Włodzisław (2007) considered two different sets of cognitive functions namely low-level (“perception, object recognition, signal analysis, discovery of structures in data, simple association, and control”) and high-level (“systematic thinking, reasoning, complex representation of knowledge, episodic memory, planning, understanding of symbolic knowledge”). The low-level cognitive functions handle the problem via combinations of either supervised or unsupervised learning methods and adaptive systems. Examples of these methods are neural, fuzzy, evolutionary, probabilistic, and statistical (Bayesian networks or kernel) methods. The high level cognitive functions are commonly considered for addressing “nonalgorithmizable” problems through the use of Artificial Intelligence (AI)-based approaches (Włodzisław, 2007). Włodzisław considered Sequence learning, reinforcement learning, machine learning, and distributed multi-agent systems as areas in which the low and high level cognitive functions overlap with each other (Włodzisław, 2007). Poole *et al.* (1998) considered CI as the study of intelligent, agents (an agent is defined as an entity that acts in an environment and does something). An intelligent agent is considered as a system that acts intelligently, and it is flexible to environmental and goal changes, and capable of learning from experience. The authors considered the overall goal of CI as understanding the principles that results in intelligent behaviors (in either natural or artificial systems). However, Włodzisław (2007) disagreed with this view: “*Computational intelligence is certainly more than just the study of the design of intelligent agents, it includes also study of all non-algoritmizahle processes that humans (and sometimes animals) can solve with various degree of competence*”. Fukuda and Kubota (1997) explained the difference between CI and AI as: “*CI aims to construct intelligence from the viewpoints of biology, evolution and self-organization. CI tries to construct intelligence by internal description, while classical AI tries to construct intelligence by external (explicit) description. Therefore, information and knowledge of a system in CI should be learned or acquired by itself*”. There are several opinions regarding to the relationship between CI and AI and in here three of these opinions raised by some of the pioneers in the field are presented in order to better reflect the existing

viewpoints.

- Marks (1993) considered CI as an alternative to AI: “*Although seeking similar goals, CI has emerged as a sovereign field whose research community is virtually distinct from AI*”.
- Bezdek (1994) concluded that CI is a sub-set of AI after analyzing three levels of system complexity.
- Fogel (1995) emphasized on adaptation as an important feature of intelligence and declared that traditional symbolic AI systems are focused on artificial rather than intelligence since they do not adapt to the new problems in new ways. As a result, Fogel implied that only CI systems are intelligent.

25.1.2. Robotics Community and CI

The advanced robotic systems are often considered to replace human being in critical and dangerous tasks. Such systems collect information from the surrounding environment and react to the situations. CI (i.e., Neural Network, fuzzy systems, evolutionary approaches) and behavior-based AI methods are commonly used in order to realize intelligence in such robotic systems (Fukuda and Kubota, 1997). A review of application of CI in autonomous mobile robotics can be found in Wang (2002). Robot Motion Planning attracted the attention of academics in robotics community due to its complexity (being NP-complete) and its importance when dealing with various robotic applications. Bullinaria and Li (2007) considered the CI-based techniques such as neural competition, evolutionary computation, and fuzzy logic abilities in terms of providing powerful tools for realization of efficient control framework with adaptable parameters as one of the main reasons for their common use in robotic problems.

This chapter discusses the application of CI in the field of autonomous mobile robotics. The paper is focused on basic concepts of CI in robotic domain with an emphasis on essential aspects of navigation such as localization, path planning, and obstacle avoidance both on singular and swarm robots. The paper reviews recently published studies on these aspects aiming to highlight the recent trends on the field. In addition, the applications of navigation and motion planning on robotic arms, haptic devices, humanoid robots, human robot interactions, and robotic hands are presented.

The outline of this chapter is as follows: Section 2 introduces Navigation, localization, and Path Planning, discusses various classical and heuristic-based approaches proposed over the years to handle the motion-planning problem and also presents some of their recent applications. Section 3 presents Swarm Intelligence and its application on Swarm Robotics and Section 4 focuses on other application of motion planning and obstacle avoidance approaches including haptics, surgical devices, robotic hands and arms, and wearable robots. Conclusion is presented in Section 5.

25.2. Navigation and Motion Planning

Definition 1: Navigation: *The process or activity of accurately ascertaining one's position and planning and following a route.*¹

Navigation is the art of steering a course through a medium and consists of two essential components known as Localization and Path Planning. In mobile robots, localization is the ability of projecting one's real world position to a location inside a map. In other words, the term localization in robots refers to the ability of determining accurate positions in the search space according to the sensory information of the environment (Roy *et al.*, 1999). Planning in mobile robots is the ability to find a short, collision-free path from the starting position towards the predefined ending location.

25.2.1. Localization

Definition 2: Localization: *a determination of the place where something is.*²

As suggested by Pugh and Martinoli (2006, 2007); Pugh and Zhang (2005); Roy *et al.* (1999), some standard simulation-based localization methods have failed in real-world applications due to their inability to overcome the environmental conditions. These conditions and constraints can be categorized as follows:

- The results of small differences between training and testing phases in terms of initial locations (i.e., in real robots, a small change in the location of maps components or small differences between the training and testing phases can affect the overall performance of the entire process (Pugh and Martinoli, 2007)).
- Robots heterogeneity (i.e., in physical robots, even identical robots tend to have small differences which result in different environmental perceptions in the same states. Such an issue contributes to problems for homogeneous robots during knowledge sharing process. Consequently, the real-world robotic applications consists of robots with different types of abilities (Pugh and Martinoli, 2007; Pugh and Zhang, 2005)).
- As the literature suggested by Roy *et al.* (1999), in real-world applications, robots' environmental perceptions are unreliable due to the uncertain nature of the world.
- Due to the dynamic nature of real-world domains, the accuracy of the map and objects' locations would be uncertain (Roy *et al.*, 1999).

Localization is a challenging problem that can be found in various robotic scenarios such as multi-robot mapping, search and rescue, chemical leak detection and chemical source localization (Lu, 2013), underwater acoustic-based navigation (Webster *et al.*, 2014), or rendezvous search where two robots are meant to localize themselves in an unknown environment and then search for each other (Ozsoyeller *et al.*, 2014). Simultaneous localization and mapping (SLAM) is an approach utilized by robots in order to generate maps for unknown environments for which *no priori* knowledge exists or for

updating maps of the environments for which *priori* information are provided, while simultaneously tracking their current location in the map. Variations of SLAMs can be categorized based on their sensory system to laser-based, sonar-based, and vision-based systems. Other variations of sensorial sources includes compasses, infrared, and global positioning system (GPS). The used sensorial systems are often subject to measurement noise and range limitations. Probabilistic approaches such as Kalman Filter (KF), Extended Kalman Filter (EKF), Particle Filter, and Expectation Maximization (EM) are filtering approaches that are commonly used in SLAM studies to address the sensory unit limitations and its associated problems (Aulinas *et al.*, 2008). These filtering methods and their associated advantages and shortcomings are presented in [Table 25.1](#). In addition to sensor-based categorization, the SLAM methods can be categorized based on the working environment to ground-based (indoor and/or outdoor) (Davison and Murray, 2002; Estrada *et al.*, 2005; Kim and Sukkarieh, 2003; Newman and Leonard, 2003; Newman, 1999) airborne (Kim and Sukkarieh, 2003, 2004), and underwater (Eustice, 2005; Eustice *et al.*, 2006; Pizarro *et al.*, 2004; Sez *et al.*, 2006; Williams *et al.*, 2000).

Table 25.1: Commonly utilized Filtering approaches in SLAM studies and their advantages and shortcomings (Aulinas *et al.*, 2008).

Filtering approach	Pros and cons	References
Variations of KF	pros: Have high convergence and capable of handling uncertainty cons: Uses Gaussian assumption and it is slow in high dimensional maps	Davison and Murray (2002); Guivant and Nebot (2001); Jensfelt <i>et al.</i> (2006); Newman and Leonard (2003); Se and Lowe (2002)
Particle Filter	pros: Capable of addressing nonlinearities and non-Gaussian noise cons: Growth in complexity	Montemerlo <i>et al.</i> (2002, 2003)
EM	pros: Handles data association problem and is optimal for map building cons: Inefficient in high dimensional environments and inefficient cost growth	Burgard <i>et al.</i> (1999); Montemerlo and Thrun (2007)

Kim and Eustice (2013) used a visual SLAM algorithm for underwater vehicle scenarios. The algorithm considered two types of visual saliency measures called local and global saliences in order to tackle common challenges in underwater visual SLAMs such as limited field of view imagery and regions that are feature-poor. In Local Saliency normalized measure of intraimage feature diversity is considered, while the global saliency uses a normalized measure of interimage rarity. The proposed visual SLAM navigation approach is novel in the sense that it considers the imageries that are more informative and better contribute in construction of the map.

Mirkhani *et al.* (2013) provided comparison results among three approaches for

mobile robot localization. The considered algorithms included GA-based localization (GLASM), Harmony-Search-based Localization (HSL), and hybrid Harmony and Differential Evaluation-based localization (HIDE). The authors claimed that HSL is capable of addressing global localization problems due to its ability to deal with the existing arbitrary noise distribution in robotic environments. Various advantages for HSL can be considered over other optimization based localization approaches such as (1) having fewer mathematical requirements, (2) not being dependent to derivative information, and (3) involving all vectors when a new solution is being generated. The HSL method compensates for the noise through representing the robot's pose uncertainty using a set of estimates that are weighted by a fitness function. The proposed hybrid HS and DE approach has advantages such as faster convergence and exploring the potential solution space. The results indicated superiority of HSL over GLASM approach with ability to guarantee safe navigation in unknown environments with the drawback of having slow convergence. The hybrid HIDE approach showed promising results due to utilizing the fast convergence speed on DE with search quality of HS approaches.

Webster *et al.* (2014) provided a comparison study among various acoustic-based navigation approaches for underwater vehicles. The study considered a single-beacon cooperative acoustic navigation problem in which the ranges and state information gathered from a single reference source with an unknown location is employed to improve the localization and navigation in a UV. In this scenario, the acoustic broadcasts between a server and a client is used to assist localization and navigation of the UV. Decentralized extended information filter (DEIF) used in this study plays the role of a filter that is applied to acoustic broadcasts on both server and client sides in order to facilitate the low-bandwidth requirements in addition to allowing the client side to reconstruct the time of launch information from the server's acoustic broadcasts. The results indicate suitability of the DEIF approach in comparison to other decentralized acoustic cooperative localization methodologies suggested by literature for underwater vehicle navigation.

Drevelle and Bonnifait (2014) discussed an estimation approach capable of quantifying the localization confidence based on constraint propagation and interval analysis. The approach is capable of providing continuous estimates of vehicle position through fusion of GPS and 3D map data. The constraint propagation of a precise 3D map and the GPS measurements are used to generate tightly coupled position domains and the GPS wrong measurements are compensated using proprioceptive history data. The 3D map and the data history contains information regarding to the drivable roads. The use of road data in constraint propagation process allows elimination of dead-reckoning drifts and reduction of positioning ambiguities. The experimental results gathered from a navigation scenario in urban environment using up to 2 satellites indicated suitability of the proposed approach for real-time positioning.

Murillo *et al.* (2013) studied image-based localization and proposed an approach that takes advantage of a global gist descriptor for panoramas images. The proposed approach

provides adequate similarity measurement that results in a balance between discriminability and computational complexity in urban environments. The similarity measurement is facilitated through an algorithm that detects the loop-closure. In addition, a tree based approach is utilized that takes advantage of principle component analysis (PCA) for decomposing the panoramic gist descriptor. The results achieved from evaluating the approach against state-of-the-art loop-closure detectors based on local features indicated the potential of the technique with ability to provide favorable or comparable performance to other algorithms commonly used in the field of image-based localization.

Colle and Galerne (2013) presented a global localization method that is capable of fusing data from heterogeneous set of sensors (on-board sensors, environmental sensors, and sensory units of other robots). The proposed approach takes into account constraints such as (1) number of measurements, (2) generic goniometric measurements, (3) information regarding to the degree of inaccuracy presented as an interval deduced from sensor tolerances, and (4) associated complexities to the incorporation of sensory data originated from heterogeneous sources.

25.2.2. Path Planning

Definition 3: Motion planning: *is a term used in robotics for the process of breaking down a desired movement task into discrete motions that satisfy movement constraints and possibly optimize some aspect of the movement.*³

Path planning is one of the essential components in navigation of mobile robots and is known as the art of finding an optimized collision-free path from the starting location toward the predefined destination (Qin *et al.*, 2004). Qin *et al.* (2004) suggested two types of path planning in the robotic domain known as (1) global path planning and (2) local path planning. In global path planning, robots have complete information about the environment (i.e., they have access to a map which provides knowledge about the positions of located obstacles and goals in the environment), while in local path planning, robots have no information about the environment or the information is not complete (partial information) (Qin *et al.*, 2004). Although simplicity and computational efficiency are known as two major advantages of path planning techniques (in static, certain, and predictable environments), the overall performance in these techniques are highly dependent on the reliability of the information, especially the map. As Gu *et al.* (2003) has discussed, due to the dynamic, noisy and unpredictable nature of real-world robotic applications, it is quite difficult to implement a path planning technique based on a well-known predefined map. Although it is possible to use path planning techniques in such a scenario, it would be costly due to the huge amount of computations (i.e., in such a case, it would be necessary to compute huge amount of possibilities). In addition, it is almost impossible to predict all possible situations and potential actions that mobile robots might

encounter in the real-world applications. As such, strategies based on specifying all possible behaviors would be brittle.

The approaches used in motion and path planning can be categorized to two groups of classical and heuristic based methods (Masehian and Sedighizadeh, 2007). The common classical methods are cell decomposition (CD), potential field (PF), road map and subgoal network (SN), and CI-based methods such as neural network (NN), Evolutionary Algorithms (e.g., genetic algorithm (GA), particle swarm optimization (PSO) and ant colony optimization (ACO)), and Fuzzy Logic. Due to the heuristically nature of the CI-based methods, in this study, they are categorized as heuristic-based. Elshamli *et al.* (2004) argued that the classical methods are capable of finding the optimal solution (if one exists) while sacrificing computation-time but heuristic-based methods generate solutions with good quality with the advantage of being computation-time efficient. More detailed description of these algorithms are presented in following section.

25.2.2.1. *Classical methods*

Classical methods guarantee to find a solution if such a solution exists and their main disadvantage is being computationally expensive and being incapable of coping with uncertainty. These disadvantages make the usage of classical approaches challenging in real world applications since such applications are unpredictable and uncertain. Methods such as *CD*, *PF*, Road Map, and Subgoal Network are employed in path planning and are categorized as classical methods.

CD, given the challenging nature of dealing with large environments and the challenging exploration problems associated with it in robot navigation scenarios, several studies suggested segmentation/re-scaling of the larger search spaces to several reduced spaces that are more manageable (Fujii *et al.*, 1998; Park *et al.*, 2001; Yang and Gu, 2004).

In *CD* a cell representation of the search space is utilized aiming to generate a sequence of collision-free cells that drives the robot from starting point to the destination. In this method, cells that contain obstacles are divided to new cells and only obstacle-free cells would be added to the sequence of collision-free path. Consequently, sequences of cells that start from the start position and end in the destination represent the path between these two positions (Faverjon and Tournassoud, 1987; Herman, 1986; Kondo, 1991; Lingelbach, 2004; Lozano-Perez *et al.*, 1987; Lozano-Perez and Wesley, 1979; Seda, 2007). Seda (2007) suggested following steps for using *CD* as motion planner for a robot.

- (1) Divide the search space to connected cells.
- (2) Construct a graph through adjacent cells (i.e., vertices denoting cells and edges connect cells with common boundary).
- (3) Determine destination and start cells and generate a sequence from the start to the destination cells that is obstacle-free cells.

(4) Generate a path from obtained cell sequence.

CD can be divided to sub categories such as Exact *CD* (Barbehenn and Hutchinson, 1995a; Schwartz and Sharir, 1983; Sleumer and Gurman, 1991) Approximate Cell Decomposition (Barbehenn and Hutchinson, 1995b; Conte and Zullil, 1995; Zhu and Latombe, 1991), and Probabilistic Cell Decomposition (Lingelbach, 2004; Rosell and Iniguez, 2005).

Exact *CD* decomposes the C-space to convex polygons resulting in centering points of cell boundaries connected together with lines in order to generate a path. The generated paths have the disadvantage of containing unnecessary turning points which makes the motion unnatural (Nakju *et al.*, 2007). Approximate *CD* divides the search space to a grid with higher resolution with each cell in the grid representing a segment of the environment and containing a flag to indicate if they are collision free. Although the method is easy to implement, the inability to offer natural presentation of the environment makes the approach inefficient for path planning (Sleumer and Gurman, 1991). Probabilistic *CD* (*PCD*) is similar to approximate *CD* with the exception of cells having predefined shapes and cells' boundaries not representing any physical meaning. Although approximate and probabilistic *CD* are faster in implementation, they are not reliable when the fraction of the free-space in the environment is small (Lingelbach, 2004).

Lingelbach (2004) used a *PCD*-based path planning in a scenario in which the robot is tasked to fetch milk from the refrigerator and place it on the kitchen's table. The *PCD* algorithm is decomposed to components of graph search, local planning, cell splitting, and probabilistic sampling. The algorithm identifies a collision-free path by iteratively refining the *CD* and in each step the probabilistic sampling is led to regions of interest by *CD*. The study provided comparison between *PCD* and Rapidly-exploring Random Trees method (RRD).

Pinder *et al.* (2013) studied the potential of a fuzzy controlled Air Vehicle (AV) that is coupled with an approximate *CD* method acting as the path planner in a scenario based on landing an autonomous AV on a helipad. The choice of *CD* as the path planner is made due to offering less complexity with respect to the landing helipad and the surrounding areas.

Some general drawbacks are considered for *CD* including combinatorial explosion, limited granularity and generating infeasible solutions (Seda, 2007). Road-map and PF can be considered as alternatives to this approach.

PF: In principle, PF can be imagined as the differential of two opposite forces of attraction and repulsion in a way that locations that direct the robot towards the destination introduce attraction force while other locations (including obstacles and other objects) in the environment introduce repulsive forces. The robot take the directions based on the combination of these forces in order to travel from the starting point toward the ending

location (Cheng and Zelinsky, 1995; Dunias, 1996; Latombe, 1991; Randal and McLain, 2003; Song, 2002).

Song (2002) utilized a decentralized PF-based controller for directing teams of robots in a scenario based on approaching, trapping, and transporting objects towards a destination location. Cheng and Zelinsky (1995) employed *PF* for obstacle avoidance in a simulation study. The author suggested the use of high temporary attraction forces toward random locations whenever the robot is stagnated between obstacles aiming to force the robot to get away from the locations in which it can get stuck. The suggested modification is useful for addressing weakness of *PF* in situations in which high concentration of close-by obstacles with high repulsive forces exist. Such circumstances can result in local minima convergence in *PF* (Rimon and Koditschek, 1992; Seda, 2007; Wang and Chirikjian, 2000).

Road-Map: Road-map (Retraction, Skeleton, or Highway) approach is designed as a set of paths containing areas that are collision-free. In this context, path planning can be seen as finding the shortest path from all possible collision-free paths (gathered in a graph) presented by the road-map from the starting position to the destination location (Siegwart and Nourbakhsh, 2004). Voronoi and visibility graphs are two of the most commonly used approaches for generating road-map graphs. In Visibility approach, the vertices of the graph represent either the locations of the start and the end points or they represent the vertices of polygonal obstacles. In this approach the edges represent the connection between nearby vertices in addition to the edges of the obstacles. Although the mentioned representation of the existing paths is advantageous for identifying minimum-length path, it also allows robots to get too close to the edges of the obstacles which can cause safety problems specially when obstacles with sharp edges exist in the environment. This issue can be resolved by Voronoi diagram approach in which the generated road-map graph tends to maximize the distance of the road from the obstacles. The main drawback of Voronoi diagram is its dependency to the length of localization sensory unit and their accuracy. That is due to the focus of the approach on generating road-maps in which the distances between the obstacles and the road are maximized and as a result it is not suitable for robots with short-range sensors. The execute-ability (i.e., the ability to easily be adopted and executed by physical robots) of the Voronoi diagram method makes it more favorable in real-world robotic applications (Siegwart and Nourbakhsh, 2004).

SN: The approach is commonly used to address motion planning problem and uses a list of reachable configurations from the starting location (Chen and Hwang, 1992; Faverjon and Tournassoud, 1987; Hwang and Ahuja, 1992; Randal and McLain, 2003). Chen and Hwang (1992) suggested a variation of *SN*-based on hierarchical path search in which the *SN* is utilized as the global planner and a local planner is used to assess the reach-ability of subgoals. The global and local planners are switched repeatedly until either a path is found or subgoal sequence is emptied. Glavina (1990) presented a randomized motion planning based on *SN* in which the planner generates random subgoals

configurations and a local planner connects the subgoals in order to construct a graph of free spaces connectivity. The main contribution of the proposed approach is its ability on search in the discrete free space rather than computational geometry. In a series of studies, Chen and Hwang (1992) introduced a search strategy called SANDROS (Chen and Hwang, 1992, 1998; Hwang and Chen, 1995) that is usable in motion planning. The SANDROS employs a local planner for connecting sequences of subgoals with the drawback of being inefficient for finding paths through winding tunnel. Bessire *et al.* (1993) employed hybrid of *SN* and *GA* in motion planning study. The suggested hybrid method utilized *GA* for evenly placed point-subgoals (landmarks) in the free space and used an additional procedure for connecting the landmarks to the goal.

The main drawbacks of the classical methods are their lack of adaptability and the lack of robustness to environmental changes (Elshamli *et al.*, 2004). This is the result of using sequential based searching algorithms (Elshamli *et al.*, 2004). To overcome the weakness of these approaches Heuristic based approaches are suggested.

25.2.2.2. *Heuristic-based methods*

As mentioned earlier, the reliability of the map and the ability of a robot to localize itself in the map play major roles in navigation problems. However, dynamism and unpredictability of real world applications which result in generation of unreliable maps often make the navigation task challenging. Probabilistic algorithms such as probabilistic roadmaps PRM and RRT are commonly considered to address general drawbacks of classical path planning methods such as high time complexity in environments with high dimensions and trapping in local minima. The main advantage of the probabilistic algorithms over classical methods is their high-speed implementation (Masehian and Sedighizadeh, 2007). An alternative to classical approaches is behavior-based methods (Cheng and Zelinsky, 1995). These methods use a set of simple and predefined behaviors that are designed to handle complex scenarios. The dependency of these methods to their current state which get influenced by environmental conditions and their ability to switch their behavior in a way to handle their current state makes them suitable for addressing scenarios based on dynamically changing environments. However, uncertainty in environmental readings can result in erroneous setting of their current state and consequently erroneous setting of their associated behaviors which results in dead-lock in robot's state (Cheng and Zelinsky, 1995).

Other studies also referred to the heuristic-based methods as population-based and behavior-based methods too. That is due to dealing with sub-set of solutions in an iterative manner in these methods. In these methods, the population is evolved in each of the iterations. Although these approaches do not guarantee to find a solution (in contrast to classical methods) but if such a solution is found, the required time would be shorter compared with classical methods. Below, some of these methods are presented and their

applications in robot motion planning are discussed.

Artificial Neural network (ANN) is a mathematical model inspired from biological neural networks designed to replicate brain computational capability in terms of processing information, adapting to data, tolerating noise, and identifying patterns. The ANN is a common method for controlling robot motions. The ANN capability in terms of solving problems can be tuned by its architectural design and learning paradigm. ANN consists of interconnected artificial neurons that are each capable of performing simple tasks. The neurons/nodes in ANN map input data to an output using weights and bias factors. The learning in the network is implemented by adjusting the weights and bias of nodes in an iterative manner in a way that nodes that best augment the performance get to increase their weights while others decrease them in order to increase the overall classification performance of the network. The process of modifying and adjusting the weights in ANNs is referred to as the *training algorithm*. *Feedforward* and *Feedback* are two commonly used topologies in the learning phase of ANNs. In *Feedforward* no back/return connection between different units (layers) is allowed and information flows forward in one direction, from input layer toward output layer. This is in contrast to *Feedback* ANNs where the cycles of connections between different layers is allowed. Feedforward ANNs are more time efficient in their learning phase in comparison with Feedback ANNs. The use of *NN* in robotic problems can be pictured in categories such as Interpreting the sensory data, Obstacle avoidance, and Path planning (Zou *et al.*, 2006). Zou *et al.* (2006) argued that hybrid of *NN* and other AI-based methods including fuzzy logic, knowledge-based systems, and evolutionary approaches better address the robot navigation problem in real world applications (Zou *et al.*, 2006). Yangmin and Xin (2005) employed the combination of particle swarm optimization (PSO) and Adaptive *NN* for navigation of a mobile robot. In the suggested approach PSO is used as the path planner in order to generate a smooth trajectory for the adaptive *NN* controller. The Adaptive *NN* is employed to drive the robot towards the goal location while avoiding the collision with obstacles. Pugh and Martinoli used single-layer discrete-time *ANN* in order to control a heterogeneous swarm of physical robots in an obstacle avoidance scenario (Pugh and Martinoli, 2007).

GA: Introduced by John Holland in 1975 is a population based approach that utilizes operators such as natural selection, crossover, and mutation (Han, 2007). *GA* is widely used in variety of optimization problems including classical traveling salesman, flow-shop optimization, and job shop scheduling. In these problems the aim is to optimize or find the best solution (in *GA* each solution (member of the population) is presented as a chromosome or genotype) out of a population of possibilities (Konar, 2005). *GA*'s inherently parallel search characteristic is the source of its attractiveness for developing near-optimal solutions (Elshamli *et al.*, 2004). *GA*'s steps such as Selection, Crossover, and Mutation are used in path planning problems. The operation of chromosome ranking and selecting a subset of chromosomes with high ranks for generating new population is

called Selection. The ranking of chromosomes is based on their fitness. The new population is generated by Crossover by exchanging portions of the chromosomes nominated by the Selection operator with each other. Mutation is the operation of randomizing one or a sub-set of chromosomes that are not achieving better fitness results over certain number of iterations in full or partially. Elitism is considered as maintaining the best performing chromosome of each generation in the new population.

Various studies have been done based on applying *GA* to different aspects of path planning and navigation problems ranging from dynamic multicast routing scenario (Brits *et al.*, 2003), multi path planning scenarios (Brits *et al.*, 2003; Hwang and Chen, 1995; Pugh and Zhang, 2005) path planning and finding the shortest path (Deneb, 1994), and navigation (Sadati and Taheri, 2002). The main shortcoming of the *GA* in path planning problems is its infeasibility in dynamic environments due to operating in a grid map or utilizing a fixed resolution in the search space. This results in inability to control the population diversity causing premature convergence.

PSO: Is an evolutionary approach that iteratively evolve a population of possible answers toward an optimum and it is inspired from animals' social behaviors (Masehian and Sedighizadeh, 2010). PSO favors convergence toward global optimum due to the identified social and cognitive behaviors in its swarm's implementation which influence the evolvement procedure of the particles. In PSO each member of the swarm population is referred to as particle. Each particle represents a possible solution for the problem at hand and it is modeled based on a position in the search space (a solution) and a velocity that derived the particle from its previous position to the new one in the search space. Cognitive and social components of the swarm formulation mimics attraction towards positions in the search space that either represent the best personal finding of the particle or the swarm's best finding. These locations in the search space are referred to as local and global-best positions respectively (Stacey *et al.*, 2003). The impact of this attraction force is controlled in the PSO's formulation by *Acceleration Coefficients*. In addition, another coefficient referred to as *Inertia Weight* is used in order to control the influence of the adjusted velocity in previous iteration on the new one. The particles in the swarm update their positions in the search space by adding their updated velocity to their last location in the search space. Various neighborhood topologies are suggested to address knowledge sharing among particles. The neighborhood topology can play an important role on how the swarm evolve over iterations since it affects the global best position of the swarm by dictating which particles can share their personal best findings with each other. PSO's overall performance depends on aspects such as how to adjust/control its parameters, how to address premature convergence (Pasupuleti and Battiti, 2006; Ratnaweera *et al.*, 2004; Vesterstrom and Riget, 2002). Several parameter adjustment and controlling mechanisms are suggested to address the PSO's problems including (1) linearly decreasing inertia weight (LDIW), (2) time varying inertia weight (TVIW), (3) random inertia weight (RANDIW), (4) fixed inertia weight (FIW), (5) time varying acceleration coefficient

(TVAC), (6) random acceleration coefficients (RANDAC), and (7) fix acceleration coefficients (FAC). It is also observed that in some problems the overall performance of the swarm improves if the velocity equation is modified by either ignoring one or more than one component or adding new components to the equation (Chang *et al.*, 2004; Pasupuleti and Battiti, 2006; Peer *et al.*, 2003; Peram *et al.*, 2003; Ratnaweera *et al.*, 2004; Yang and Simon, 2005). More detailed discussion on PSO can be found in Atyabi *et al.* (2007); Atyabi and Powers (2013a); Atyabi and Samadzadegan (2011).

PSO's potential in solving complex problems makes it favorable in various scenarios and domains. Modified versions of PSO are used in several path planning, searching scenarios, and robots navigation studies. Two variations of mutation-based PSO are employed for path planning in a study by Yuan-Qing *et al.* (2004) and to handle the conveying tasks in Guillaume, Frddrique and Beat study (1998), Pugh and Martinoli (2006); Pugh and Zhang (2005) used PSO with local neighborhood topology in multi-robot search optimization problem and showed that PSO is capable of outperforming GA in such problems. A modified version of PSO called DEADALUS is suggested by Suranga (2006) to handle cul de sac problem (robots' stagnations problem) in a simulated obstacle avoidance scenario. A variation of PSO that utilized a stream function called Potential Flow for obstacle avoidance proved to be feasible for robot navigation scenarios with static and dynamic obstacles (Chengyu *et al.*, 2007). Masehian and Sedighizadeh (2010) employed two variations of PSO namely Multi-Objective PSO (MOPSO) and negative PSO (NPSO) in hybrid with probabilistic roadmap (PRM) method for robot path planning. The study provided comparison between MOPSO-PRM, NPSO-PRM, and pure PRM methods and indicated that NPSO showed slight advantage over PSO and generated shorter and smoother paths in comparison to pure PRM while being more time efficient.

ACO: The first ACO algorithm referred to as the ANT System is introduced by Marco Dorigo and his colleagues (Latombe, 1991). ACO is inspired by the social behavior of the ants when foraging for a food source and the ability to guide other ants toward regions of interest using their deposited pheromones (Han, 2007). In ACO, the paths' qualities are evaluated based on the quantity of the pheromones deposited by ants that traveled that path considering Concentration and Proportion factors that indicate the length of the path and the number of ants that traveled the path respectively (Hao *et al.*, 2006). The rapid algorithm execution of ACO is one of its main advantages over GA (Han, 2007).

Several studies employed ACO in robotic path planning scenarios (Hasiao *et al.*, 2004; Shirong *et al.*, 2006; Taylor *et al.*, 2006). Hao *et al.* (2006) used the hybrid of ACO and Artificial Potential Field (APF) approach aiming to use ACO as a global path planning and APF as local path planning approach. In the algorithm, ACO and APF cooperated with each other through using pheromone trajectory of ACO in APF to prevent trapping in local minima. McLurkin and Yamins (2005) suggested 4 algorithms for role switching between a team of robots in a dynamic path planning study. Shi *et al.* (2008) used the combination of ACO and PSO in a robot path planning study in which ACO is utilized to plan paths in

robots transitable territory and PSO is utilized in order to optimize the parameters of ACO.

Riadi *et al.* (2013) reported comparison results of various benchmark functions on a variation of ACO method called ACO-PT that utilizes Prospect Theory (a widely used algorithm in economics). The results indicated considerable improvement in state transition strategy when PT algorithm is used causing overall improvement of the ACO performance. The study is inspired from earlier investigations carried in Autonomous Systems and Robotics Research Center in Salford University under Prof Nefti-Meziani's supervision with a focus on utilization of PT algorithm for modifying the decision-making process in Evolutionary-based methods such as PSO and ACO (Bunny, 2012; Riadi, 2014).

Several advantages and short comings can be named for ACO. The advantages includes inherent parallelism, positive feedback which contributes in rapidly discovering good solutions, feasibility to be used in problems similar to Traveling Salesman, and its adaptability to dynamically changing environments. The disadvantages include complexity of the theoretical analysis, possibility of generating sequences of random decisions, iteratively changing probability distribution, and uncertainty in the time to convergence.

Other heuristic-based path planning methods: this subcategory includes several other approaches that are utilized by robotic community to handle path planning problems including simulated annealing (SA), Stigmergy, Wavelets, tabu search (TS).

25.2.2.3. *Applications of classical and heuristic-based methods in global path planning*

Otte and Correll (2013) suggested a multiple parallelized tree structure called C-FOREST for generating the shortest path. The parallelization is designed in a way to allow the knowledge sharing (i.e., the shortest path found so far) between trees at each step of the algorithm. The overall performance of the C-FOREST approach is assessed against three different experimental setups including a manipulator arm, robotic team, and alpha benchmark and the results illustrated the feasibility of the proposed approach in terms of superlinear speedup with both manipulator arm and robotic team and its in-feasibility with alpha benchmark within path planning problems.

Murphy and Newman (2013) proposed a risky planning approach for path planning problem. The proposed approach utilizes probabilistic costmaps in combination with fast heuristic searches. The authors argued that often in robotic problems it is more desirable to find "good" paths quickly rather than finding the shortest path in a time consuming manner. Based on this principle the proposed approach generates pseudo-optimal paths that exceed the length of the optimal path by 30%, 20% reduction in the required search time can be achieved. The experimental results also indicate that if only on 50% of the executions finding the shortest path is targeted, 60% reduction on the required search time can be achieved.

Jaillet and Porta (2013) introduced a sampling-based path planner approach called AtlasRRT that is especially tailored to address problems with kinematics constraints. AtlasRRT utilizes a coordinated construction of an atlas and a RRT. The atlas is employed for generating branches in the C-space manifold and in order to retain the efficiency of the RRT exploration and the RRT algorithm is utilized as a tool for specifying the expansion directions for the atlas. Jaillet and Porta (2013) adapted various approaches for accelerating the convergence of asymptotically-optimal planners resulting in developing two variations of the AtlasRRT method called AtlasRRT* and AtlasBiRRT* that utilizes bidirectional search strategy, propagation of the improvements found in new samples, and tools for generating atlas that parameterizes the configuration space of manifold with high-dimensional continuation.

Roadmap is considered as one of the classical approaches used in path planning. Probabilistic roadmap approach generates a graph that represents the layout of the C-space. The created roadmap graph in offline mode is utilized during the online mode in order to rapidly generate feasible solutions. Smoothing and hybridization of multiple graphs are considered as two of the standard approaches for improving the quality of the paths generated by roadmap. An alternative approach for improving the paths quality is to use larger and denser roadmaps. Such approaches tend to generate solutions with optimal costs (asymptotic optimality) with the drawback of being cost demanding in the sense that their required construction time, storage, and transmission are inefficient.

Marble and Bekris (2013) proposed two approaches that construct roadmaps that guarantees asymptotic near-optimality through interleaving asymptotically optimal RPM* approach with graph spanners. The first method sequentially applies the graph spanner to the outputs of the PRM* and the second method eliminates some edges during the roadmap construction phase following an incremental approach. The experimental results indicated the feasibility of both approaches for generating asymptotically near-optimal paths with the advantage of reducing construction time, roadmap density.

Masoud (2013) employed harmonic potential field (HPF) approach in a simulation study. The proposed method synchronizes the control signal in order to align the gradient of the HPF and the velocity of the robot. The simulation results indicated suitability of the approach for situations with the presence of actuator noise, saturation, and uncertainty. In addition, the approach is favorable due to its simplicity and its ability to migrate the kinematic path characteristics from an HPF to the dynamic trajectory of UGV.

Workspaces containing serial manipulators with large degree of freedom are another domain that benefit from path planning approaches. Shukla *et al.* (2013), used the concept of monotonic optimality for optimizing the manipulators' paths in addition to utilizing proximity based penalizing schemes for obstacle avoidance to address path planning problem in such workspaces. The use of B-spline function model for path representation resulted in local control of the trajectory and smoother joint movement. The experimental

results indicated decrease in path length and monotonic optimality of the paths.

Martinez *et al.* (2013) explained a real-time vision based motion and tracking approach that is capable of robustly track large, rapid drogue motions within 30 frames per second. The used vision strategy approach consisted of 3 steps: (1) Hierarchical Multi-Parametric Multi-Resolution Inverse Compositional Image Alignment approach that utilizes robust tracking, (2) segmentation and template matching approach that utilizes detection, and (3) 3D position estimation. The proposed approach showed potential during the experiments based on autonomous aerial refueling problem with only losing the drogue's position when it is located out of the site of view.

Kala (2012) employed co-evolutionary Genetic Programming method for multi-robot motion planning problem in which various sources and goals are considered for each member of the robot team. The study used genetic programming for generating path in a simulated maze-like map (individual robot level) and utilized a master evolutionary method in order to facilitate overall path optimality requirements (swarm level). The feasibility of the proposed approach is assessed within several simulation studies with variety of swarm sizes.

25.2.2.4. Applications of classical and heuristic-based methods in local path planning

In real-world hazardous scenarios, it will often happen that the navigation is impossible. In such scenarios, the map is not reliable. Unreliability in these scenarios is due to the dynamic and unpredictable nature of real world applications. The problem of path planning in scenarios in which the prior generated maps and environmental informations are unreliable or incomplete is referred to as Local Path Planning. To solve such problems, behavioral-based approaches are suggested (Cheng and Zelinsky, 1995). In behavioral-based approaches, a set of simple, predefined behaviors would be designed in a way to solve complex scenarios. The advantage of these approaches is their reliability in dynamic environments due to their dependency on their current situation (state) and the set of behaviors designed for that state instead of localization and planning. Although behavior-based approaches are reliable in dynamic environments, they still have difficulties with uncertainty. Cheng and Zelinsky (1995) argued that in behavior-based approaches uncertainty causes faults in terms of chosen states. This problem is known as irresolvable state. Irresolvable states would often cause dead-lock in agents' states.

Faigl *et al.* (2013) investigated a multi-goal path planning problem in which a closed shortest path in a polygonal map is desired in a way that all goals be visited by the robot. The study takes advantage of self-organizing map (SOM) for addressing the traveling salesman problem. The authors considered three variations of SOM referred to as alternate goal, attraction, and goal interior in order to address polygonal goals. The goal interior approach is a simple adaptation of the centroids of the polygonal goals in which nodes that are not inside the polygonal goal are moved toward the centroid. The attraction point

approach aims to generate shorter paths by introducing new attraction points at the border of polygonal goals. In the alternate goal approach, the closest point to a node that is located on the border of the polygonal goal is used instead of the centroid point.

Faigl *et al.* (2013) studied the problem of active visual search (AVS) in unknown or partially known large environments using uncertain semantics of the environments. The proposed approach takes advantage of existing knowledge from possible worlds with spatially similar models in order to generate a model for the unknown environment. The proposed methodology is validated through comparison with human performance and a greedy coverage-based search strategy and showed potential in addressing the AVS problem.

Gerstmayer-Hillen *et al.* (2013) proposed a vision-based controlling system for mapping the environment in a cleaning robot application. The proposed approach is novel in the sense that instead of acquiring robot's exact position in the environment it relies on partial position estimation. That is the robot estimates its distance from the previously visited lanes and uses its orientation in the world in order to generate an estimation of its partial position. The experimental results indicated feasibility of the proposed approach with minimal overlaps or gaps between the lanes.

25.2.2.5. *Obstacle avoidance*

One of the classical problems in robotics is Obstacle Avoidance in which the task is to satisfy some control objectives while maintaining non-collision constraints. It is customary to consider obstacle avoidance as a sub-routine or extra constraint/condition in path planning and therefore categorize the methods to two categories of local and global approaches (Khansari-Zadeh and Billard, 2012). It is shown that local obstacle avoidance approaches such as the Bug's algorithm (Lumelsky and Skewis, 1990), the Vector Field Histogram (Borenstein and Koren, 1991) and the Curvature-Velocity (Simmons, 1996) are capable of providing fast responses when perturbations occur with the drawback of resulting in paths that in most cases are only locally optimal. In contrast, the global methods such as those introduced in global path planning guarantee to find a feasible path (if it exists) with the drawback of being computationally intensive which makes them unsuitable for dynamically changing environments in which the obstacles appear suddenly (Khansari-Zadeh and Billard, 2012).

This section presents a short review on some of the recently developed obstacle avoidance approaches that are less generic compared with the classical and the heuristic-based approaches discussed previously and are more problem specific.

Rufli *et al.* (2013) investigated the potential of decentralized motion planning agents using continuous control obstacle ($C^n - CO$). The presented method can be considered as an extension to reciprocal velocity obstacle (RVO) approach in such a way that the

trajectory segments consider C^n continuity instead of piecewise linearity. The approach constructs a collision free motion for a team of homogeneous agents through choosing a potential trajectory from $(C^n - CO)$ complement. In the study, $(C^n - CO)$ is a set of feedback controlled trajectories that result in collision between agents and the complement of this set is expected to provide a collision free motion. The proposed approach is infeasible for heterogeneous teams of robots that have non-circular shapes or follow different collision-avoidance strategies.

Gomez *et al.* (2013) suggested the use of fast marching square (FM^2) approach for path planning of the robot formation problem. The proposed algorithm is capable of providing safe and collision free paths under various environmental constraints including uncertainty (on both robots' and obstacles' locations) and existence of static or dynamic obstacles. FM^2 approach is considered as a path planning approach with no local minimum capable of generating smooth trajectories in addition to significantly reducing the computational load when employed under uncertain environments. In path planning based on FM^2 approach, the original Fast Marching (FM) algorithm is applied in two different phases in which, in the first phase, FM is applied to the map in order to generate a new map in which each pixel is associated with a value between 0 and 1 in which lower values represent improbable regions with value of 0 representing obstacles and walls (depicted as black regions in the map) and 1 representing highly probable regions (depicted as lighter regions in the map). In the second phase, FM is applied to the resulting map of the first phase with inclusion of time of arrival as a new axis to the 2D map. In this phase a new potential is generated that represents the propagation of an electromagnetic wave and considering the goal as the origin of the wave, the propagation is continued until the current position of the robot is reached.

Madani *et al.* (2013) conducted experiments based on moving obstacles scenarios. The proposed approach models the robot manipulators using *a priori* knowledge. The mechanism takes advantage of one step direct calculation and it does not specify any inversion model. The experimental results with a robot with 3 DOF manipulators indicated the effectiveness of the proposed approach in comparison with some other variable structure controllers.

Hardy and Campbell (2013) presented a contingency path planning approach that utilizes a path optimization algorithm that simultaneously optimizes several contingency paths. The resulting framework allows the robot to maintain a deterministic path while it is capable of predicting multiple mutually exclusive obstacles. In order to achieve fast optimization within multiple contingency paths, a spline-based representation of the contingency path is employed in the path optimization algorithm. In order to avoid the obstacles a collision probability bound is proposed that allows avoidance with uncertain relative orientation and location. The simulation results indicated the potential of the proposed approach within aspects such as safety in terms of resulting in less collision and

aggressiveness in terms of obstacle spacing.

Purcaru *et al.* (2013) a gravitational search algorithm (GSA)-based method in a motion planning study comprising partially unknown environment with the presence of static and dynamic obstacles. The feasibility of the GSA approach is assessed by providing comparison results with basic PSO.

Rakshit *et al.* (2013) presented an adaptive memetic algorithm based on hybridization of Q-learning and artificial bee colony (ABC). The choice of such hybridization is made considering global and local search abilities of ABC and Q-learning methods respectively. In addition to providing comparison results on multi-robot path planning scenarios, the study offers performance evaluation and assessment on 25 well-known benchmark functions. The results indicate feasibility of the proposed approach compared to four variations of differential evolution (DE) method.

Hinojosa *et al.* (2011) assessed the potential of a hybrid fuzzy and reinforcement learning method called generalized probabilistic fuzzy RL (GPFRL) in three robot control problems with stochastic and uncertain nature under both deterministic and continuous environments. The proposed RL approach combines the universal-function-approximation capability of fuzzy systems, while considering probability distributions over possible consequences of an action. The GPFRL modified actor-critic (AC) learning architecture while enhancing the learning structure through the use of probability measure. The proposed approach facilitates convergence through utilizing gradient-decent-weight-updating algorithm. The approach is assessed in three problems namely cart-pole balancing problem, DC-motor control problem, and mobile robot navigation against conventional methods. The results indicated the GPFRL is robust under probabilistic uncertainty in addition to offering enhanced learning speed.

Theodoridis and Hu (2006) tackled the strategic route learning problem in mobile robots using a novel algorithm called Fuzzy Sarsa (λ) Learning (FS λ L). The approach hybridizes Reinforcement Learning (RL) and Fuzzy Logic (FL) in addition to utilizing a learning algorithm called Sarsa (λ) in order to tune FL controller. The experimental design is based on route learning task in which experience is developed in exploring robot, while interacting with the environment using a discretized FL controller and later on optimal policies necessary for achieving goals and minimizing training error are learned. The results achieved from experiments indicated that FS λ L performs considerably better than conventional Sarsa (λ) method and provided higher learning rate.

25.3. Swarm Robotics and the Navigation Problem

Since developing robots with high abilities, skills, and intelligence is very expensive and complicated, using a group of simpler robots with simpler skill-sets that are expendable and are capable of cooperating with each other in order to solve complicated tasks is more reasonable. Such advantages attracted scientists and resulted in application of swarm-based approaches for solving a variety of robotic problems (Jiang and Kumar, 2013; Trueba *et al.*, 2013; Williams and Sukhatme, 2013). Coordination and Cooperation are two main components of swarm-based multi-robot approaches.

Swarm Robotics is one of the approaches utilized for coordinating multi-robot systems. In such systems a desired collective behavior emerges from interaction of robots with each other and the environment. Within a swarm of homogeneous robots this can be seen as constant sharing of information among the swarm members in order to make a decision either in the swarm or individual robot level. Within a swarm of heterogeneous robots this can be imagined as having a central controller robot that receives information from swarm members and makes the swarm level decisions and issues commands to all or individual members. This section presents recent developments on coordination and cooperation aspects of multi-robot systems.

Jiang and Kumar (2013) presented an approach for multi-robot coordination in congested systems. The authors suggested a modified behavior regulation for amplifying the robot's velocity and a direct control of robot velocity in the behavioral dynamics. The amplification of robot's velocity is achieved through utilization of interaction force among robots and helps the robot to slow down in certain places. The results achieved from simulation study indicated effectiveness of the proposed methodology for multi-robot coordination in congested systems with bottlenecks.

Kudelski *et al.* (2013) reported development of a simulated environment called RoboNetSim that models network complications and constraints in multi-robot scenarios. The developed simulator is advantageous in the sense that it provides more realistic measurements on efficiency of the networking and multi-robot communication aspects of the generated scenarios which results in better compatibility of the generated approaches for real-world environments.

Yamamoto and Okada (2013) introduced an approach for controlling the swarm that is inspired from the appearance of congestion in crossing pedestrian flows in high density urban areas. In crossing pedestrian flow problem, the pedestrians use a self-organized phenomena based on avoidance of collision with other pedestrians. The authors proposed continuum model of the crossing pedestrians flow that generates a density factor from dynamic changes of the congestion and utilized a control method in order to improve the average velocity. The principle idea in the approach is to take advantage of dynamic interactions among diagonal stripe patterns that are shaped by the crossing pedestrians to guide the person/object that is aiming to move in to the flow. The approach is validated

through simulation study.

de Lope *et al.* (2013) studied coordination of multiple robots in the sense of heterogeneous task selection. The study considered generation of decentralized solutions in the sense that robots autonomously and individually select tasks in a way that it maintains optimal task distribution. This is facilitated through utilizing Response Threshold Model and Learning Automata-based probabilistic algorithms. The authors employed random and maximum principle approaches for task selection and based on the results achieved from simulation studies concluded that the latter is a better fit for the task. In addition the results indicated noise robustness of the response threshold method and also higher time demand from the learning automata-based probabilistic approach in some cases. The additive noise influences the system in the sense of simulation of robots' erroneous estimation of the real number of pending tasks. The results also indicated feasibility of the proposed approaches for task selection/allocation in autonomous and individual manner for a team of multiple robots without the need for any central task scheduler unit.

Zhu *et al.* (2013) proposed a combination of bottom up and high level layer structure for navigation of a swarm of robots in which individual robots are independent and capable of providing some degree of autonomy and intelligence thanks to the bottom-up (low-level) layer, while a higher level layer deals with contingency and guarantees convergence in complex circumstances. The unique capability of the proposed bottom-up approach in terms of allowing the advantage of various algorithms to address different situations makes it more favorable compared with the top-down hybrid architectures.

In contrast to the multilayer level of intelligent presented in Zhu *et al.* (2013), Pimenta *et al.* (2013) presented an approach in which the principle idea is based on the utilization of a single software/algorithm for controlling a swarm of robots rather than allowing independent decision-making and behavior among the swarm members. The proposed approach aims to direct the swarm of robots (as whole) towards region of interest and is advantageous in terms of not being dependent on individual swarm members and robust to dynamic addition and deletion of swarm members. This proposed approach uses smoothed-particle hydrodynamics (SPH) simulation technique for generating an abstract representation of the swarm in the sense of incompressible fluid that is subject to external forces which provides a loose way for controlling the swarm. The results achieved from simulation and actual studies indicated efficiency of the approach with ability to deal with high number of agents (>1000 in simulation) and also smaller swarm sizes (in experiments with actual robots).

Hemakumara and Sukkarieh (2013) suggested a system identification method that is based on the dependent Gaussian process for addressing modeling constraints of stability and control derivations of UAVs. The considered modeling constraints include inability to represent a wide flight envelope and being dependent on prior knowledge of the model

structure. The proposed method addresses the control derivatives and system stability through capturing cross coupling of input parameters and it is evaluated through simulation study using a highly coupled oblique wing vehicle in addition to being utilized on a delta-wing UAV platform. The results highlighted advantages such as better identification of coupling flight modes, being more robust to uncertain estimations, and being applicable to wider flight envelopes for the proposed approach in comparison to traditional methods.

Atyabi (2009); Atyabi *et al.* (2010a, 2010b) Atyabi and Powers (2010, 2013b), investigated the potential of two modified PSO approaches called area extended PSO (AEPSO) and cooperative AEPSO (CAEPSO) for controlling a small swarm of simple robots in various scenarios. The study showed AEPSOs capability to perform better local search compared to basic PSO, random search and linear search in time dependent and dynamic domains. Moreover, due to the existence of higher level of cooperation between agents controlled by AEPSO and CAEPSO, they were robust to noise and tasks' time-dependency. The robustness is achieved through heuristics that provided knowledge sharing and a better balancing system between essential behaviors of the swarm (exploration and exploitation). Furthermore, CAEPSO showed its reliability with homogeneous agents by providing higher level of cooperation and experience/knowledge sharing resulting in reduction of illusion effect (a rare case of combinatorial uncertainty) from the environment. The cooperation between agents also allowed CAEPSO to improve the overall search performance.

Formation control is one of the popular problems in robot swarm cooperation that refers to the task of controlling a set of robots that follow a predefined path and maintains a specific formation pattern. Lin *et al.* (2013) categorized existing swarm based approaches for formation control problem to three basic categories of behavioral, virtual structure, and leader-follower. In virtual structure category the swarm is considered as a single rigid robot and therefore the path planning problem for a swarm of robots is simplified to path planning for a rigid robot which is advantageous due to the ease of implementation with the drawback of having low path planning flexibility. In behavior-based category a set of desired behaviors (movement towards destination, obstacle avoidance, maintaining the formation) are defined for each robot in order to create its trajectory. In the leader-follower category, the robot's ability depends on its role in the swarm. In such approaches one or more robots act as leaders with responsibility for moving along a predetermined trajectory, while the remaining swarm members are responsible for following the leaders, while maintaining the desired relative position from each other and the leader (Lin *et al.*, 2013).

Fukushima *et al.* (2014) reported on development of a formation control law that is capable of converting the optimal control problem to a mixed-integer quadratic programming problem through utilizing feedback linearization with a model predictive control (MPC). In addition, a branch-and-bound (B&B) approach is used in order to

address obstacle avoidance problem. The numerical experiments indicated significant computation time reduction.

Peng *et al.* (2013) extended the trajectory tracking control of a single robot to a simulation-based formation control study based on multiple non-holonomic mobile robots. The proposed methodology consists of a back-stepping technique that utilizes leader's proposed kinematic controller for real-time tracking, an angular velocity control law for addressing asymptotic stability (i.e., global stability for follower and local stability for the team formation), and a bioinspired neurodynamic approach for addressing the velocity jumps (The impractical velocity jump is considered as a traditional drawback of the back-stepping techniques). The simulation results indicated the effectiveness of the proposed approach with understanding that the utilized scenario is noiseless and no delay is considered in the operation of the mobile robots which caused occasional unexpected stability in the results.

Chand and Carnegie (2013) investigated feasibility of a hierarchy of task allocation and sensory capabilities within a heterogeneous swarm of robots for mapping and exploration problems. The proposed hierarchical system consists of various robots with different computational and sensing capabilities in which the most capable robots (in terms of computational power) are declared as the managers and other robots in the swarm with lower sensing and computation power are allocated with less demanding tasks and are coordinated with manager robots. The manager robots maintain a global information gathered from lower rank robots in the hierarchy and generates a map. Within the second and third level of the proposed hierarchy, planner and explorer robots are considered and take advantage of global maps constructed on manager robots in order to execute their path planning and map building tasks.

Martin-Ortiz *et al.* (2013) proposed an approach called selective inspection method based on auctions (SIMBA) in a swarm robotic study. The SIMBA method is inspired from competitive spirit of animal and insect communities in a sense to compensate for the lack of communications between the swarm members by introducing a competitive bidding mechanism that allows robots to make their own decisions in a competitive environment in a way to benefit the overall team gain. The proposed bidding mechanism is designed as a loop that continuously evaluate robots bids (solutions for routing problem) until all the bids become winners. The experimental results in both simulation and real-world scenarios indicated the feasibility of the proposed approach even when communication is lacking or restricted between the swarm members.

A summary of advantages and shortcomings of various classical and heuristic based path planning methods are presented in [Table 25.2](#).

25.4. Other Applications of Motion Planning in Robotic Domain

The application of navigation and motion planning often exceeds from controlling the maneuvers of simplistic wheeled robots between an start and an end location to controlling robots with complex designs and functionalities, while they are performing complicated tasks in more complex environments and under more sophisticated constraints. Other sectors that benefit from applications of the robot motion planning includes Automotive, Food industry, pharmaceutical, medical, health care, and rehabilitation, and so on (i.e., Haptics, Surgical Devices, Robotic Arms, Grasping Hands, Humanoid, and Wearable Robots). This section discusses the application of motion planning and navigation in such other robotic areas.

Table 25.2: Summary of pros and cons of classical and heuristic-based approaches utilized in robot motion planning problem.

Approach	Pros and cons
Classical	
CD	cons: Combinatorial explosion, limited granularity, generating infeasible solutions
PF	pros: Simple implementation cons: Computationally expensive, traps in local minima
Visibility Graph	pros: Advantageous for identifying minimum-length path cons: Not suitable in the presence of obstacles with sharp edges
Voroni Diagram	pros: Execute-ability, suitable in the presence of obstacles with sharp edges cons: Not suitable for robots with short range sensors
SN	pros: Requires small amount of memory cons: Its computationally intensiveness depends on the local operator used PF is suggested as the best local-operator for SN
Heuristic-based	
ANN	pros: Easy to implement, capable of approximating any input/output mapping cons: Requires extensive amount of training data, the training phase is computationally intensive
GA	pros: Easy to implement, no parameter to adjust cons: Premature convergence due to inability to control the population diversity
PSO	pros: Easy to implement, few parameters to adjust fast searching ability cons: Premature convergence, lack of justifiable rules for parameter settings
ACO	pros: Rapid algorithm execution, inherent parallelism adaptability to dynamic environments, suitable for traveling sales man type problems cons: Complexity of theoretical analysis, generating sequence of pseudo-random

25.4.1. Haptics and Surgical Devices

Several advantages can be counted for the use of haptic devices for training the novice and intermediate personals in hospitals. The general approach for the novice trainees to be trained on surgical devices and procedures is to first take theoretical study (books, lectures, video files), then under mentor-ship, and finally under real situations with required supervision. The use of multi-modal virtual reality platforms have advantages such as safety, availability, repeatability, and flexibility of the training in terms of being able to tailor the training session to more on reflect sub-skills in the procedure in which the trainee has less experience (Gosselin *et al.*, 2013).

Pose SLAM is a variation of SLAM in which only the robot path gets estimated and the landmarks are utilized only for generating relative constraints between positions. Valenci *et al.* (2013) utilized Pose-SLAM graph of constraints in order to identify a path to the goal that contains lowest accumulated pose uncertainty. The proposed approach uses pose of the Pose-SLAM map as nodes of a belief roadmap (BMR) for generating reliable routs. The authors considered the BMR since it is considered as a variation of SLAM that best address the stochastic nature of the SLAM. In BMR, the edges of the roadmap contains information regarding to the changes in the uncertainty when it is traversed. The original BMR approach has the drawback of being dependent to having known model of the environment, while this is not always possible in real world applications. In their study, Valenci *et al.* (2013) utilized the maps generated by Pose SLAM as a BMR. The authors assessed the potential of the proposed approach against standard path planning approaches and illustrated its feasibility in addition to counting advantages such as being defined in the belief space, only utilizing the uncertainty raised by moving between positions, and being scalable to large environments due to using approximated marginal covariances.

Valenci *et al.* (2013) discussed the use of CardioArm robot for diagnosing and treating ventricular tachycardia (VT) of epicardial origin. CardioArm is a snake-like medical robotic system that is highly flexible and is designed to allow physicians to view, access and perform procedures on the heart with the advantage of only requiring one access point. The potential of the CardioArm in the context of epicardial mapping is assessed through utilizing it for small surgeries on Nine animals and three human patients (the patients experienced multiple failed attempts with endocardial technique prior to the study) and the results indicated technical success for navigating, mapping, and ablation tasks on all subjects except for one of the human patients in whom the blood obscured visualization.

Arcese *et al.* (2013) presented the controller design of a microrobotic system that is

magnetically guided in blood vessels with application for performing minimal invasive medical procedures. The authors presented a nonlinear model that takes into account contact and electrostatic forces in addition to non-Newtonian behavior of the blood and generates a trajectory that minimizes the control efforts. In addition, an adaptive back-stepping control method is utilized in order to synthesize the Lyapunov stabilizing control law. The simulation results indicated the robustness of the proposed model to noise and uncertainty arising from physiological parameters.

Pisla *et al.* (2013) discussed dynamic behavior and the kinematics of a parallel hybrid surgical robot called PARASURG-9M that contains a robotic arm, a PARASURG-5M unit with five motors, and a PARASIM (a robotized surgical instrument with four motors). In comparison to serial robots, the presented parallel robot has advantages in terms of providing higher stiffness, smaller mass, and more precise manipulation. In terms of its application in an operating room, the robot is a compact solution and more affordable and uses analytical methods to address the inverse kinematics problem.

Becker *et al.* (2013) employed a virtual fixture framework combined with motion scaling behaviors in Micron platform for vein tracing and conducting surgery on retina membrane peeled with tissue contact. The authors utilized hard virtual fixtures in order to constrain the tip and ensuring that some constraints were never violated in addition to using tremor suppression filters to compensate for unwanted motions with high frequency. The proposed framework uses stereo cameras mounted on the microscope in order to generate real time virtual fixtures and task-independent behaviors to the operator.

Gosselin *et al.* (2013) discussed the importance of properly specifying and designing haptics for training sensorimotor skills. The authors developed a haptic interface with capabilities and advantages such as workspace, force, stiffness, transparency, and bandwidth. The developed platform is utilized for training sensorimotor drilling skills. The system is judged by expert surgeons and it is considered as very realistic with the ability to reproduce similar force and position behaviors as it is felt and observed in real surgeries. In addition, the results achieved from using the platform for training novice and intermediate residents indicated their ability to progress after three weeks of training.

25.4.2. Robotic Arms and Grasping Hands

In addition to localization, mapping, path planning, and obstacle avoidance other capabilities such as object detection, recognition, and manipulation are required in order for mobile robots to carry out manipulation tasks in domestic environments (Stuckler *et al.*, 2013). The application of robotic manipulators in manufacturing varies from sheet forming (Belchior *et al.*, 2013) to welding (Chen *et al.*, 2013). Stuckler *et al.* (2013) proposed an approach for grasping objects from planar surface. The proposed approach uses scene segmentation to process depth images in real time and generates feasible and collision free paths for grasping objects in segmented images. Authors suggested two

variations of grasping called side-grasp (approaching the object along vertical axis and keeping the grippers horizontally aligned) and top-grasp (approaching the object from top grasping the object with finger tips). In addition, collision free grasping method is used to sample grasping candidates (e.g., based on pose and shape) and omit infeasible grasps and solutions that include collisions and rank the remaining candidates in order to identify the most promising one. The omission occurs through a filtering mechanism that considers factors such as grasp width, object height, reachability, and collisions and the ranking of the remaining solutions is achieved through considering factors such as distance to object center, grasp width, grasp orientation, and distance from robot. The results indicate that the proposed approach is feasible with situations in which the object is spatially distinguishable (Stuckler *et al.*, 2013). A detailed review of various grasping application can be found in Sahbania *et al.* (2012).

Grasp control which is focused on the evaluation and identification of the required grasping force in order to provide stable and feasible grasping is another challenging area in this field specially in the presence of external disturbances (Lippiello *et al.*, 2013). Lippiello *et al.* (2013), investigated grasping force optimization problem for bimanual dexterous- hand and proposed an iterative formulation based approach with low computational load that takes advantage of a sub-optimal single hand optimization algorithm joined with an initial-point evaluation method (Lippiello *et al.*, 2013). The utilized iterative formulation approach reduces the computational load by avoiding evaluation of the initial point at the beginning of each iteration. The results achieved from simulations indicated the feasibility of the proposed approach.

Many degrees of freedom on robotic hands that are needed in order to imitate human hand movement causes complications in the design and controller of such devices due to the high number of motors required. Gioioso *et al.* (2013) tackled this problem through designing a synergy-based mapping approach that uses a virtual object in order to specify the mapping in the task space in addition to avoid dissimilarities between kinematics of human and robotic hands. The use of synergy-based mapping is also beneficial under uncertain conditions. The results of the simulation study with two different robotic hands with dissimilar kinematics highlighted the potential of the proposed approach specifically in terms of being more efficient in internal force mapping and direction of motion (Gioioso *et al.*, 2013).

Chen *et al.* (2013) described a J-groove joint robot used for generating welding paths. The proposed solution incorporates interpolation and local modification modules in order to address complex welding paths such as cubic B-spline curves in addition to utilizing PSO for partitioning of the path in its point inversion module. The experimental results indicated the efficiency of the approach in terms of generating high quality welding solutions in addition to the feasibility of the methodology and the resulting robotic arm for welding applications.

Zacharia *et al.* (2013) studied the implication of task scheduling and motion planning for industrial manipulators. Within the industrial systems, it is crucial to navigate the manipulators around multiple obstacles in order to reach to some task points and perform some predefined tasks. The solutions are required to be generated within time optimum sequence in the sense that the manipulators are needed to visit the task points in a collision free manner. The authors formulated an objective function through projecting the workspace on the B-Surface. The formulated function is utilized in order to minimize the required visiting time for multiple task points. The proposed approach consists of a GA for encountering the robot inverse kinematics. The simulation results indicated the capability of the proposed approach for generating near optimal touring solutions for multi-goal path planning for environments constrained by obstacles.

Baranes and Oudeyer (2013) introduced a self adaptive goal generation architecture called SAGG–RIAC for high dimensional redundant robots that allow active learning of inverse models. The architecture is novel in the sense that based on competence progress it samples task parameterizations. This novel task parameterization sampling process allows the robot to solve the task through triggering learning of low-level motor parameters. The proposed architecture is examined over three robotic setups of robotic arm (for learning the inverse kinematics), quadruped robot (for learning omnidirectional locomotion), robotic arm (learning to control a fishing rod), and the results indicated that the exploration in task space is much faster than actuator space when working with redundant robots. In addition, the results showed that the efficiency can be improved significantly if the tasks are selected based on the proposed competence progress rather than random task selection.

Popescu *et al.* (2013) studied the control problem in a hyper redundant robot arm. The utilized model consists of a chain of periodically spaced vertebrae with passive joints that provide rotation, controllable friction force, and elastic contact. The utilized controller consists of two components of “boundary torque variable structure” and “damping force control” which facilitates simple solutions for both arm and task control with guaranteed convergence in spite of existing errors caused by the model parameter’s uncertainty. The experimental results indicated the feasibility of the proposed approach for controlling hyper redundant robot arms.

Crespi *et al.* (2013) introduced a robotic system called Salamander that takes advantage of four legs and an actuated spine and is capable of walking and swimming. The authors justified the need for generating robots with actuators that allow robot’s movement similar to animals by expressing the necessity of generating robots with realistic embodiments and environmental interactions that can be used for studies in which the complex movements cannot be simulated as in sensorimotor loop and motor activity investigation in neuroscience in which such embodiment is crucial. The authors provided comparison results with real salamanders in terms of body and limb movement and illustrated similarities between the Salamander robots movement and the real salamander.

Keating and Oxman (2013) considered the existing application of robotic arm in industry as inflexible and underutilized since they are mostly used for cyclic tasks rather than as a multi-process, multi-parametric system. That is, the current utilization of the robotic arm systems in the industrial application is within the computational modeling of the environment that does not support generative design platforms due to the existing restrictions in design process and protocols. However, taking advantage of a multi-process, multi-parametric system as suggested by authors has advantages such as reduction in cost, minimization in physical footprint, and integrated performance capabilities. The study utilized the proposed approach for robotic fabrication by combining additive, formative, and subtractive fabrication technologies using a six-axis robotic arm.

Tansel *et al.* (2013) presented a two phase robot selection decision support system (ROBSEL) to assist the decision makers with the question of which robot is the best choice for the problem at hand. The aim of the system is to minimize dependency of the industrial people on expert personell for selection of suitable robots for their business. The ROBSEL takes advantage of a fuzzy analytical hierarchy process (FAHP) system to rank the feasible robots that are survived in the elimination phase in which the decision support system eliminates unsuitable choices based on the received parametrization from the users. The required parametrization is shaped based on the answers that are received from the customers on various criteria in the first phase. The proposed model is advantageous in comparison to existing systems due to the use of correlation test which minimizes the number of criteria and shapes an independent set of criteria. The resulting minimal criteria set contributes to the ROBSEL system by making the system more sensitive to changes in the criteria weights and also adding the capability of differentiating between robots with similar technical specifications.

Ignjatovic *et al.* (2013) tackled the problem of optimization of complex robotic cell in terms of energy efficiency and air compression. Various approaches are considered in the study for defining the parameters of influence that optimizes the working cycle, maximizes the productivity, and energy efficiency and their feasibility is evaluated experimentally considering factors such as manipulators' velocity, robots position in the working space, trajectory of the robot, the pressure of the compressed air, and the length of the tube. The authors described the optimal method for such a problem as a method that considers steps such as (1) establishing and identifying the primary criterion (minimum manufacturing costs, maximum productivity, ...), (2) optimizes the critical parameters that influences the electricity consumption first, and (3) adjusts the related parameters to the air compressor based on the robot and applied criterion constraints.

25.4.3. Humanoid and Wearable Robots

Kajita and Yokoi (2013) studied the foot slippage impact on humanoid platform. The

authors hypothesized that the minimization of the power caused by floor friction is the cause of the turning motion and described a model based on this friction which focuses on determining the amount of rotation from a slip by only considering the trajectory and shape of the feet. The authors suggested that the feet velocity and the friction coefficient between the soles of the robot and the floor does not affect the turning angle. The hypothesis is examined and confirmed using a HRP-2 humanoid robot.

Escande *et al.* (2013) discussed a planner method for humanoid robots for multipurpose, multi-contact motions through loosening up the possible contacts that may occur between parts of the robot and the environment. The proposed approach employs a tree representation of possible contact configurations using inverse kinematics that is formulated as an optimization problem. The proposed contact planner is a modified version of potential field planner called best first planning (BFP). The BFP approach generates a tree configuration of contact space that in each iteration only the best performing leafs expands. The best leaf is considered as the leaf with highest potential over configuration space. The planner method is examined on three scenarios of ladder climbing, passing through narrowed passages (tunnel), and sitting on the table using HRP-2 humanoid robot. The results indicate feasibility of the proposed approach in all scenarios considered in experiments.

Rouanet *et al.* (2013) studied the impact of various human-robot interfaces in training humanoid robots to recognize some objects. The study employed four interfaces that vary in terms of the type of feedback that they provide and they include media-based artifacts such as smart phone, Wiimote, and Wiimote with laser, and natural human gesture based interfaces such as Wizard-of-Oz recognition system. The study compared these interfaces with each other and evaluated their efficiency in terms of usability, learning, and user's experience. The designed experiments is based on teaching the difference between 12 visual objects and the results indicated that smart-phone interface is more suitable for non-expert users and allows them to provide training examples in a level as efficient as expert users. The study also showed that artifact-mediated teaching interfaces are more efficient for teaching the robot and offer better user's experience and usability due to the associated gesture-based interaction component.

Stegall *et al.* (2013) explored the importance of the anterior lunge DOF in robotic exoskeleton. The study employed an active leg exoskeleton (ALEX) II in the experiments and proposed two configuration to address the issue. These configurations varied in terms of their locking mechanism. In the first configuration, the anterior/posterior translation of the exoskeleton is locked while the rest of the system is left unlocked, while in the second configuration the anterior/posterior DOF is left unlocked. The experimental design is based on inviting subjects to walk for a period of time at their own pace on a treadmill and meanwhile be trained to learn the required gain template. The results indicated that the unlocked configuration approach provides better overall performance in terms of retention and adaptation.

Vitiello *et al.* (2013) reported the development of an assistive wearable robot (elbow exoskeleton) for rehabilitation purposes. The developed device is called neuro-robotics elbow exoskeleton (NEUROExos) and it is designed to assist patients with post-stroke rehabilitation. The authors counted three main features required in a rehabilitation device such as (1) comfortable human-robot interface (in order to transmit the interaction torque), (2) kinematic compatibility between the human and the device (in order to avoid the risk of human articulations overloading caused by improper torque transmission to the patient's joints), and (3) effective actuation system (in order to address both condition of human-in-charge and device-in-charge). The developed NEUROExos device addresses these required features through utilizing features such as 4 DOF passive mechanism, double-shade links, and an actuation system with compliant antagonistic.

The physical human–robot interface design in wearable devices plays an important role in their efficiency given that it determines the compatibility of the device's kinematics with the patient's skeleton, robot efficiency, and adaptation to various anthropometrics. Cempini *et al.* (2013) argued that failure to address these issues results in misalignment of the robot's and patient's joints which causes ineffective torque control and consequently inappropriate articulations' loading forces and soft tissues' loading forces. Cempini *et al.* (2013) investigated this issue and proposed a self-alignment mechanism in the context of a general analytical approach based on a kinematic analysis of human-robot skeleton for the design of exoskeletons that do not suffer from such misalignment effects.

25.5. Conclusion

This chapter presented the application of CI in the field of autonomous mobile robotics. The chapter started by explaining the relationship between CI and AI and discussed basic concepts of CI in robotic domain emphasizing on essential components of navigation such as localization and path planning. A set of classical-and behavior-based path planning methods are explained and their application on both local and global path planning problems are presented. In addition, the application of motion planning on swarm robotics is explained and two aspects of swarming such as coordination and cooperation and their applications are discussed. The chapter is concluded by presenting recent developments in robot motion planning in robotic systems with complex designs and functionalities such as those used in Automotive, Food industry, pharmaceutical, health care and rehabilitation, and so on.

Acknowledgments

This Research/research paper was supported by the GAMMA Programme which is funded through the Regional Growth Fund. The Regional Growth Fund (RGF) is a £3.2 billion fund supporting projects and programs which are using private sector investment to generate economic growth as well as creating sustainable jobs between now and the mid-2020s. For more information, please go to www.bis.gov.uk/rgf.

References

- Arcese, L., Fruchard, M. and Ferreira, A. (2013). Adaptive controller and observer for a magnetic microrobot. *IEEE T. Robot.*, 29, pp. 1060–1057.
- Atyabi, A., PhonAmnuaisuk, S., Chin, K. H. and Samadzadegan, S. (2007). Particle swarm optimizations: A critical review. In *Proceeding of Conf IKT07, Third Conference of Information and Knowledge Technology*, Ferdowsi University, Iran, pp. 1–7.
- Atyabi, A., Amnuaisuk, S. P. and Ho, C. K. (2010a). Applying area extension pso in robotic swarm. *J. Intell. Robot. Syst.*, Springer, 58, pp. 253–285.
- Atyabi, A., Amnuaisuk, S. P. and Ho, C. K. (2010b). Navigating a robotic swarm in an uncharted 2d landscape. *Appl. Soft Comput.*, Elsevier, 10, pp. 149–169.
- Atyabi, A. and Powers, D. M. W. (2010). The use of area extended particle swarm optimization (aepso) in swarm robotics. In *11th International conference on Control Automation Robotics & Vision, ICARCV*, pp. 591–596.
- Atyabi, A. and Powers, D. M. W. (2013a). Review of classical and heuristic-based navigation and path planning approaches. *International Journal of Advancements in Computing Technology, IJACT*, 5, pp. 1–14.
- Atyabi, A. and Powers, D. M. W. (2013b). Cooperative area extension of pso-transfer learning vs. uncertainty in a simulated swarm robotics. In *11th International Conference on Informatics in Control, Automation and Robotics, ICINCO*, pp. 177–184.
- Atyabi, A. and Samadzadegan, S. (2011). Particle swarm optimization: A survey. In Walters, L. P. (ed.), *Applications of Swarm Intelligence*. Hauppauge, USA: Nova Publishers, pp. 167–178.
- Atyabi, A. (2009). Navigating agents in uncertain environments using particle swarm optimization. MSc Thesis. Multimedia University of Malaysia.
- Aulinas, J., Petillot, Y., Salvi, J. and Llado, X. (2008). The slam problem: A survey. In *Proceedings of the 2008 Conference on Artificial Intelligence Research and Development: Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence*, pp. 363–371.
- Baranes, A. and Oudeyer, P.-Y. (2013). Active learning of inverse models with intrinsically motivated goal exploration in robots. *IEEE T. Robot.*, 29, pp. 308–320.
- Barbehenn, M. and Hutchinson, S. (1995a). Toward an exact incremental geometric robot motion planner. In *IEEE/RSJ Int. C. Int. Robot.*, pp. 39–44.
- Barbehenn, M. and Hutchinson, S. (1995b). Efficient search and hierarchical motion planning by dynamically maintaining single-source shortest paths trees. In *IEEE T. Robotics Autom.*, 11(2), pp. 198–214.
- Becker, B., MacLachlan, R., Lobes, L. and Hager, G. (2013). Vision-based control of a handheld surgical micromanipulator with virtual fixtures. *IEEE T. Robot.*, 29, pp. 674–683.
- Belchior, J., Guillo, M., Courteille, E., Maurine, P., Leotoing, L. and Guines, D. (2013). Offline compensation of the tool path deviations on robotic machining: Application to incremental sheet forming. *Robot. Cim-Int. Manuf.*, 29, pp. 58–69.
- Bessire, P., Ahuactzin, J., Talbi, E. and Mazer, E. (1993). The adriane's clew. algorithm: Global planning with local methods. In *Proc. 1993 IEEE/RSJ Int. C. Int. Robot.*, IEEE, pp. 1373–1380.
- Bezdek, J. (1994). What is computational intelligence?. In Zurada, J. M., Marks II, R. J. and Robinson, C. J. (eds.), *Computational Intelligence, Imitating Life*. IEEE Computer Society Press, pp. 1–12.
- Borenstein, J. and Koren, Y. (1991). The vector field histogram fast obstacle avoidance for mobile robots. *IEEE T. Robot. Autom.*, 7, pp. 278–288.
- Brits, R., Engelbrecht, A. and Bergh, F. (2003). Scalability of niche pso. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium, SIS '03.*, pp. 228–234.
- Bullinaria, J. and Li, X. (2007). An introduction to computational intelligence techniques for robot control. *Ind. Robot.*, 34(4), pp. 295–302.
- Bunny, M. N. (2012). The use of prospect theory framework in constrained multi-objective particle swarm optimisation. Ph.D. Thesis. School of Computing, Science and Engineering College of Science and Technology University of

Salford, Salford, UK.

- Burgard, W., Fox, D., Jans, H., Matenar, C. and Thrun, S. (1999). Sonar-based mapping with mobile robots using em. In *Proceedings — 16th International Conference on Machine Learning*.
- Cempini, M., Rossi, S. D., Lenzi, T., Vitiello, N. and Carrozza, M. (2013). Self-alignment mechanism for assistive wearable robots: A kinetostatic compatibility method. *IEEE T. Robot.*, 29, pp. 236–250.
- Chand, P. and Carnegie, D. A. (2013). Mapping and exploration in a hierarchical heterogeneous multi-robot system using limited capability robots. *J. Robot. Auton. Syst.*, 61, pp. 565–579.
- Chang, B., Ratnaweera, A., Halgamuge, S. and Watson, H. (2004). Particle swarm optimization for protein motif discovery. In *Genet. Program. Evol. M.*, pp. 203–214.
- Chen, C., Hu, S. and Shen, J. (2013). An approach to the path planning of tube-sphere intersection welds with the robot dedicated to j-groove joints. *Robot. Cim.-Int. Manuf.*, 29, pp. 41–48.
- Chen, P. and Hwang, Y. (1992). Sandros: A motion planner with performance proportional to task difficulty. *Proc. IEEE Int. Conf. Robot.*, 3, pp. 2346–2353.
- Chen, P. and Hwang, Y. (1998). Sandros: A dynamic graph search algorithm for motion planning. *IEEE T. Robotic. Autom.*, 14(3), pp. 390–403.
- Cheng, G. and Zelinsky, A. (1995). A physically grounded search in a behavior based robot. In *Proc. Eighth Australian Joint Conference on Artificial Intelligence*, pp. 547–554.
- Chengyu, H., Xiangning, W., Qingzhong, L. and Yongji, W. (2007). Autonomous robot path planning based on swarm intelligence and stream functions. In *Springer, ICES2007, LNCS 4684*, pp. 277–284.
- Colle, E. and Galerne, S. (2013). Mobile robot localization by multiangulation using set inversion. *Robot. Auton. Syst.*, 61, pp. 39–48.
- Conte, G. and Zullil, R. (1995). Hierarchical path planning in a multi-robot environment with a simple navigation function. *IEEE T. Syst., Man, Cyb.*, 25(4), pp. 651–654.
- Crespi, A., Karakasiliotis, K., Guignard, A. and Ijspeert, A. (2013). Salamandra robotic ii: An amphibious robot to study salamander-like swimming and walking gaits. *Robot. Auton. Syst.*, 61, pp. 49–73.
- Davison, A. and Murray, D. (2002). Simultaneous localization and map-building using active vision. *IEEE T. Pattern Anal.*, 24(7), pp. 865–880.
- de Lope, J., Maravall, D. and Quinonez, Y. (2013). Response threshold models and stochastic learning automata for self-coordination of heterogeneous multi-task distribution in multi-robot systems. *Robot. Auton. Syst.*, 61, pp. 714–720.
- Deneb, (1994). Telegrip user manual. *Deneb Robotics Inc.*
- Drevelle, V. and Bonnifait, P. (2014). Localization confidence domains via set inversion on short-term trajectory. In *IEEE T. Robot.*, pp. 1–13.
- Dunias, P. (1996). Autonomous robots using artificial potential field. In *CIP-Data library Technische universiteit Eindhoven*.
- Elshamli, A., Abdullah, H. and Areibi, S. (2004). Genetic algorithm for dynamic path planning. In *Proc. IEEE CCECE 2004*, pp. 677–680.
- Escande, A., Kheddar, A. and Miossec, S. (2013). Planning contact points for humanoid robots. *Robot. Auton. Syst.*, 61, pp. 428–442.
- Estrada, C., Neira, J. and Tardos, J. (2005). Hierarchical slam: Real-time accurate mapping of large environments. *IEEE T. Robot.*, 21(4), pp. 588–596.
- Eustice, R. (2005). Large-area visually augmented navigation for autonomous underwater vehicles. PhD Thesis.
- Eustice, R., Singh, H., Leonard, J. and Walter, M. (2006). Visually mapping the rms titanic: Conservative covariance estimates for slam information filters. *IEEE T. Robot.*, 22(6), pp. 1100–1114.
- Faigl, J., Vonasek, V. and Preucil, L. (2013). Visiting convex regions in a polygonal map. *J. Robot. Auton. Syst.*, 61, pp. 1070–1083.
- Faverjon, B. and Tournassoud, P. (1987). A local-based approach for path planning of manipulators with a high number

- of degrees of freedom. In *Proc. 1987 IEEE Int. Conf. Robot.*, IEEE, pp. 1152–1159.
- Fogel, D. (1995). Review of computational intelligence imitating life. *IEEE T. on Neural. Netwov.*, 6, pp. 1562–1565.
- Fujii, T., Arai, Y., Asama, H. and Endo, I. (1998). Multilayered reinforcement learning for complicated collision avoidance problems. *Proc. IEEE Int. Conf. Robot.*, 3, pp. 2186–2191.
- Fukuda, T. and Kubota, N. (1997). Computational intelligence in robotics and mechatronics. In *23rd International Conference on Industrial Electronics, Control and Instrumentation, IECON'97*, pp. 1517–1528.
- Fukushima, H., Kon, K. and Matsuno, F. (2014). Model predictive formation control using branch-and-bound compatible with collision avoidance problems. *IEEE T. Robot.*, pp. 1–10.
- Gerstmayer-Hillen, L., Roben, M. K. F., Kreft, S., Venjakob, D. and Moller, R. (2013). Dense topological maps and partial pose estimation for visual control of an autonomous cleaning robot. *J. Robot. Auton. Syst.*, 61, pp. 497–516.
- Gioioso, G., Salvietti, G., Malvezzi, M. and Prattichizzo, D. (2013). Mapping synergies from human robotic hands with dissimilar kinematics: An approach in object domain. *IEEE T. Robot.*, 29, pp. 825–837.
- Glavina, B. (1990). Solving findpath by combination of goal-directed and randomized search. In *Proc. 1990 IEEE Int. Conf. Robot.*, IEEE, pp. 1176–1181.
- Gomez, J. V., Lumbier, A., Garrido, S. and Moreno, L. (2013). Planning robot formations with fast marching square including uncertainty. *J. Robot. Auton. Syst.*, 61, pp. 137–152.
- Gosselin, F., Bouchigny, S., Megard, C., Taha, F. and Delcampe, P. (2013). Haptic systems for training sensorimotor skills: A use case in surgery. *Robot. Auton. Syst.*, 61, pp. 380–389.
- Gu, D., Hu, H., Reynolds, J. and Tsang, E. (2003). Ga-based learning in behaviour based robotics. In *Proc. IEEE International Symposium on Computational Intelligence in Robotic and Autonomous*, pp. 101–106.
- Guillaume, B., Frddrique, B. and Beat, H. (1998). Multi-robot path-planning based on implicit cooperation in a robotic swarm. In *Proc. of the Second Int. Conf. Auton. Agents*. Minneapolis, United States, pp. 39–46.
- Guivant, J. and Nebot, E. (2001). Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE T. Robot. Autom.*, 17(3).
- Han, K. (2007). Collision free path planning algorithms for robot navigation problem. In M.Sc Thesis. Department of Electrical and Computer Engineering, Missouri-Columbia University, Columbia.
- Hao, M., Yantao, T. and Linan, Z. (2006). A hybrid ant colony optimization algorithm for path planning of robot in dynamic environment. In *Int. J. Inf. Tech.*, 12(3), pp. 78–88.
- Hardy, J. and Campbell, M. (2013). Contingency planning over probabilistic obstacle avoidance for autonomous road vehicles. *IEEE T. Robot.*, 29, pp. 913–929.
- Hasiao, Y., Chnang, C. and Chien, C. (2004). Ant colony optimization for best path planning. In *Proc. IEEE/ISCIT'04*, pp. 109–113.
- Hemakumara, P. and Sukkarieh, S. (2013). Localization in urban environments using a panoramic gist descriptor. *IEEE T. Robot.*, 29, pp. 813–824.
- Herman, M. (1986). Fast, three-dimensional, collision-free motion planning. In *Proc. IEEE Int. Conf. Robot.*, New York, pp. 1056–1063.
- Hinojosa, W. M., Nefti-Meziani, S. and Kaymak, U. (2011). Systems control with generalized probabilistic fuzzy-reinforcement learning. *IEEE T. Fuzzy Syst.*, 19(1), pp. 51–64.
- Holland, J. (1975). Adaptation in natural and artificial systems. In *University of Michigan Press, Ann Arbor*.
- Hoshino, S. and Seki, H. (2013). Multi-robot coordination for jams in congested systems. *Robot. Auton. Syst.*, 61, pp. 808–820.
- Hwang, Y. and Ahuja, N. (1992). Gross motion planning — a survey. In *ACM Comput. Surv.*, 24, pp. 219–291.
- Hwang, Y. and Chen, P. (1995). A heuristic and complete planner for the classical mover's problem. In *Proc. 1995 IEEE Int. Conf. Robot.* IEEE, pp. 729–736.
- Ignjatovic, I., Komendal, T., Seslija, D. and Malisa, V. (2013). Optimization of compressed air and electricity consumption in a complex robotic cell. *Robot. Cim-Int. Manuf.*, 29, pp. 70–76.

- Jaillet, L. and Porta, J. (2013). Path planning under kinematic constraints by rapidly exploring manifolds. *IEEE T. Robot.*, 29, pp. 105–117.
- Jensfelt, P., Kragic, D., Folkesson, J. and Bjorkman, M. (2006). A framework for vision based bearing only 3D slam. In *Proc. — IEEE Int. Conf. Robot., and Automation, ICRA*, pp. 1944–1950.
- Jiang, Q. and Kumar, V. (2013). The inverse kinematics of cooperative transport with multiple aerial robots. *IEEE T. Robot.*, 29, pp. 136–145.
- Kala, R. (2012). Multi-robot path planning using co-evolutionary genetic programming. *Expert Syst. Appl.*, 39(3), pp. 3817–3831.
- Keating, S. and Oxman, N. (2013). Compound fabrication: A multi-functional robotic platform for digital design and fabrication. *Robot. Cim-Int. Manuf.*, 29, pp. 439–448.
- Khansari-Zadeh, S. and Billard, A. (2012). A dynamical system approach to realtime obstacle avoidance. *Springer Verlag Auton. Robot.*, 32(4), pp. 433–454.
- Kim, A. and Eustice, R. (2013). Real-time visual slam for autonomous underwater hull inspection using visual saliency. *IEEE T. Robot.*, 29, pp. 719–733.
- Kim, J. and Sukkarieh, S. (2003). Airborne simultaneous localisation and map building. *Proc. — IEEE Int. Conf. Robot.*, 1, pp. 406–411.
- Kim, J. and Sukkarieh, S. (2004). Autonomous airborne navigation in unknown terrain environments. *IEEE T. Aero. Elec. Sys.*, 40(3), pp. 1031–1045.
- Konar, A. (2005). Computational intelligence principles, techniques and applications. Springer Verlag.
- Kondo, K. (1991). Motion planning with six degrees of freedom by multi-strategic, bidirectional heuristic free space enumeration. *IEEE T. Robotic. Autom.*, 7(3), pp. 267–277.
- Kudelski, M., Gambardella, L. and Caro, G. D. (2013). RobonetSim: An integrated framework for multi-robot and network simulation. *Robot. Auton. Syst.*, 61, pp. 483–496.
- Latombe, J. (1991). Robot motion planning. *Kluwer Int. Ser. Eng. C.*, London.
- Lin, C., Chen, K., Hsiao, P. and Chuang, W. (2013). Motion planning of swarm robots using potential-based genetic algorithm. *Int. J. Innov. Comput., I. ICIC*, 9, pp. 305–318.
- Lingelbach, F. (2004). Path planning using probabilistic cell decomposition. *Int. Conf. Robot.*, pp. 467–472.
- Lippiello, V., Siciliano, B. and Villani, L. (2013). A grasping force optimization algorithm for multiarm robots with multifingered hands. *IEEE T. Robot.*, 29, pp. 55–67.
- Lozano-Perez, T., Jones, J., Mazer, E., O'donnell, P., Grimson, E., Tournassoud, P. and Lanusse, A. (1987). Handey a robot system that recognizes, plans, and manipulates. In *Proc. IEEE Int. Conf. Robot.*, New York, pp. 843–849.
- Lozano-Perez, T. and Wesley, M. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. In *Commun. ACM*, 22, pp. 560–570.
- Lu, T. (2013). Indoor odour source localisation using robot: Initial location and surge distance matter. *Robot. Auton. Syst.*, 61, pp. 637–647.
- Lumelsky, V. and Skewis, T. (1990). Incorporating range sensing in the robot navigation function. *IEEE T. Syst. Man Cyb.*, 20, pp. 1058–1069.
- Madani, T., Daachi, B. and Benallegue, A. (2013). Adaptive variable structure controller of redundant robots with mobile/fixed obstacles avoidance. *J. Robot. Auton. Syst.*, 61, pp. 555–564.
- Marble, J. and Bekris, K. E. (2013). Asymptotically near-optimal planning with probabilistic roadmap spanners. *IEEE T. Robot.*, 29, pp. 432–444.
- Marks, R. (1993). Computational versus artificial. *IEEE T. Neural. Netw.*, 4, pp. 737–739.
- Martinez, C., Richardson, T., Thomas, P., du Bois, J. and Campoy, P. (2013). A vision-based strategy for autonomous aerial refueling tasks. *J. Robot. Auton. Syst.*, 61, pp. 876–895.
- Martin-Ortiz, M., de Lope, J. and de la Paz, F. (2013). Action based method for graphiclike maps inspection by multi-robot system in simulated and real environments. *J. Robot. Auton. Syst.*, 61, pp. 676–681.

- Masehian, E. and Sedighzadeh, D. (2007). Classic and heuristic approaches in robot motion planning — a chronological review. *Proc. Wrld. Acad. Sci., E.*, 23, pp. 101–106.
- Masehian, E. and Sedighzadeh, D. (2010). Multi-objective pso-and npso-based algorithms for robot path planning. *Adv. Electr. Comput. Eng.*, 10(4), pp. 69–76.
- Masoud, A. A. (2013). A harmonic potential field approach for joint planning and control of a rigid, seperable nonholonomic, mobile robot. *J. Robot. Auton. Syst.*, 61, pp. 593–615.
- McLurkin, J. and Yamins, D. (2005). Dynamic task assignment in robot swarms. In *Proc. of Robotics: Science and Systems*, Cambridge, USA.
- Mirkhani, M., Forsati, R., Shahri, A. and Moayedikia, A. (2013). A novel efficient algorithm for mobile robot localization. *Robot. Auton. Syst.*, 61, pp. 920–931.
- Miura, K., Kanehiro, F., Kajita, S. and Yokoi, K. (2013). Slip-turn for biped robots. *IEEE T. Robot.*, 29, pp. 875–887.
- Montemerlo, M., Thrun, S., Koller, D. and Wegbreit, B. (2002). Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the National Conference on Artificial Intelligence*, pp. 593–598.
- Montemerlo, M., Thrun, S., Koller, D. and Wegbreit, B. (2003). Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *18th Int. Joint Conf. Artif. IJCAI*, Acapulco, Mexico, pp. 1151–1156.
- Montemerlo, M. and FastSLAM, S. T. (2007). A scalabale method for the simultaneous localizaton and mapping problem in robotics. *Spr. Tra. Adv. Robot.*, 27.
- Murillo, A., Singh, G., Kosecka, J. and Guerrero, J. (2013). Localization in urban environments using a panoramic gist descriptor. *IEEE T. Robot.*, 29, pp. 146–160.
- Murphy, L. and Newman, P. (2013). Risky planning on probabilistic costmaps for path planning in outdoor environments. *IEEE T. Robot.*, 29, pp. 445–457.
- Nakju, L., Chanki, K. and Chung, W. (2007). A practical path planner for the robotic vacuum cleaner in rectilinear environments. *IEEE T. Consum. Electr.*, 53, pp. 519–527.
- Neuzil, P., Cerny, S., Svanidze, O., Bohuslavek, J., Plasil, P., Jehlicka, P., Holy, F., Petru, J., Kuenzler, R. and Sediva, L. (2013). Single-site access robot-assisted epicardial mapping with a snake robot: prepration and first clinical experience. *J. Robot. Surg.*, 7, pp. 103–111.
- Newman, P. and Leonard, J. (2003). Consistent, convergent and constant-time slam. In *Int. Joint Conf. Artif. IJCAI*, pp. 1143–1150.
- Newman, P. (1999). On the structure and solution of the simultaneous localization and mapping problem. PhD Thesis, University of Sydney.
- Otte, M. and Correll, N. (2013). C-forest: Parallel shortest path planning with superlinear speedup. *IEEE T. Robot.*, 29, pp. 798–806.
- Ozsoyeller, D., Beveridge, A. and Isler, V. (2014). Symmetric rendezvous search on the line with an unknown initial distance. *IEEE T. Robot.*, pp. 1–14.
- Park, K., Kim, Y. and Kim, J. (2001). Modular q-learning based multi-agent cooperation for robot soccer. *Robot. Auton. Syst.*, 35, pp. 109–122.
- Pasupuleti, S. and Battiti, R. (2006). The gregarious particle swarm optimizer (g-psos). In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, Seattle, Washington, USA, pp. 67–74.
- Peer, E., Bergh, F. and Engelbrecht, A. (2003). Using neighbourhood with the guaranteed convergence pso. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium SIS'03*, pp. 235–242.
- Peng, Z., Wen, G., Rahmani, A. and Yu, Y. (2013). Leader-follower formation control of non-holomonic mobile robots based on bioinspired neurodynamic based approach. *J. Robot. Auton. Syst.*, 61, pp. 988–996.
- Peram, T., Veeramachaneni, K. and Mohan, C. (2003). Fitness distance ratio based particle swarm optimization (fdr-psos). In *Swarm Intelligence Symposium, SIS'03*, pp. 174–181.
- Pimenta, L., Pereira, G., Michael, N., Mesquita, R., Bosque, M., Chaimowicz, L. and Kumar, V. (2013). Swarm

- coordination based on smoothed particle hydrodynamics technique. *IEEE T. Robot.*, 29, pp. 383–399.
- Pinder, J. M., Theodoridis, T. and Nefti-Meziani, S. (2013). Autonomous aerial vehicle using a single camera for flight stabilisation. In *International Workshop on Human-Machine Systems, Cyborgs and Enhancing Devices, IEEE Robotics & Automation Society*, pp. 1–2.
- Pisla, D., Gherman, B., Vaida, C., Suciu, M. and Plitea, N. (2013). An active hybrid parallel robot for minimally invasive surgery. *Robot. Cim-Int. Manuf.*, 29, pp. 203–221.
- Pizarro, O., Eustice, R. and Singh, H. (2004). Large area 3D reconstructions from underwater surveys. *Ocean'04 — MTS/IEEE Techno-Ocean: Bridges across the Oceans — Conference Proceedings*. 2, pp. 678–687.
- Poole, D., Mackworth, A. and Goebel, R. (1998). Computational intelligence: A logical approach. *Oxford University Press*.
- Popescu, N., Popescu, D. and Ivanescu, M. (2013). A spatial weight error control for class of hyper-redundant robots. *IEEE T. Robot.*, 29, pp. 1043–1050.
- Pugh, J. and Martinoli, A. (2006). Multi-robot learning with particle swarm optimization. In *International Conference on Autonomous Agents and Multi Agent Systems AAMAS'06. Hakodate, Japan*, pp. 441–448.
- Pugh, J. and Martinoli, A. (2007). Parallel learning in heterogeneous multi-robot swarms. In *IEEE C. Evol. Computat. CEC'07. Singapore*, pp. 3839–3846.
- Pugh, J. and Zhang, Y. (2005). Particle swarm optimization for unsupervised robotic learning. In *Proceedings of 2005 IEEE in Swarm Intelligence Symposium, SIS05*, pp. 92–99.
- Purcaru, C., Precup, R.-E., Iercan, D., Fedorovici, L.-O., David, R.-C. and Dragan, F. (2013). Optimal robot path planning using gravitational search algorithm. *Int. J. Artif. Intell.*, 10(S13), pp. 1–20.
- Qin, Y.-Q., Sun, D.-B., Li, N. and Cen, Y.-G. (2004). Path planing for mobile robot using the particle swarm optimization with mutation operator. In *Proc. International Conference on Machine Learning and Cybernetics. Shanghai*, pp. 2473–2478.
- Rakshit, P., Konar, A., Das, S. and Nagar, A. K. (2013). Abc-tdql: An adaptive memetic algorithm. In *Proc. 2013 IEEE Workshop on Hybrid Intelligent Models and Applications. Singapore*, pp. 35–42.
- Randal, W. and McLain, T. (2003). Motion planning using potential fields. In Brigham Young University, Provo, Utah.
- Ratnaweera, A., Halgamuge, S. and Watson, H. (2004). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE T. Evolut. Comput.*, 8(3): pp. 240–255.
- Riadi, I., Nefti-Meziani, S., Theodoridis, T. and Wahab, M. N. A. (2013). Ant colony optimization with prospect theory for continuous optimization problems. In *International Workshop on Human-Machine Systems, Cyborgs and Enhancing Devices, IEEE Robotics & Automation Society*, pp. 1–2.
- Riadi, I. (2014). Cognitive ant colony optimization: A new framework in swarm intelligence. PhD Thesis. School of Computing, Science and Engineering College of Science and Technology University of Salford, Salford, UK.
- Rimon, E. and Koditschek, D. (1992). Exact robot navigation using artificial potential functions. *IEEE T. Robotic. Autom.*, 8(5), pp. 501–518.
- Rosell, J. and Iniguez, P. (2005). Path planning using harmonic functions and probabilistic cell decomposition. *Int. Conf. Robot.*, pp. 1803–1808.
- Rouanet, P., Oudeyer, P., Danieau, F. and Filliat, D. (2013). The impact of human-robot interfaces on the learning of visual objects. *IEEE T. Robot.*, 29, pp. 525–541.
- Roy, N., Burgard, W., Fox, D. and Thrun, S. (1999). Coastal navigation-robot motion with uncertainty. In *IEEE/RSJ Int. Conf. Robot.*
- Rufli, M., Alson-Mora, J. and Siegwart, R. (2013). Reciprocal collision avoidance with motion continuity constraints. *IEEE T. Robot.* 29.
- Sadati, N. and Taheri, J. (2002). Genetic algorithm in robot path planning problem in crisp and fuzzified environments. In *Proc. IEEE ICIT'02*, Bangkok, Thailand.
- Sahbania, A., El-Khoury, S. and Bidaud, P. (2012). An overview of 3d object grasp synthesis algorithms. *Robot. Auton. Syst.*, 60, pp. 326–336.

- Schwartz, J. and Sharir, M. (1983). On the piano movers problem I: The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Commun. Pure and Appl. Math.* 36, pp. 345–398.
- Se, S., Lowe, D. and Little, J. (2002). Mobile robot localization and mapping with uncertainty using scale invariant visual landmarks. *Int. J. Robot. Res.*, 21(8), pp. 735–758.
- Seda, M. (2007). Roadmap method vs. cell decomposition in robot motion planning. In *Proceedings of 6th WSEAS International Conf. on Signal Processing, Robotics and Automation*, Grees, pp. 127–132.
- Sez, J., Hogue, A., Escolano, F. and Jenkin, M. (2006). Underwater 3D slam through entropy minimization. In *Proc. — IEEE Int. Conf. Robot. 2006*, pp. 3562–3567.
- Shi, C., Bu, Y. and Liu, J. (2008). Mobile robot path planning in three-dimensional environment based on aco-pso hybrid algorithm. In *Proceedings of the 2008 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. Xi'an, China, pp. 252–256.
- Shirong, L., Linbo, M. and Jinshou, Y. (2006). Path planning based on ant colony algorithm and distributed local navigation for multi-robot systems. In *Proc. IEEE Int. Conf. on Mechatronics and Automation*, pp. 1733–1738.
- Shukla, A., Singla, E., Wahi, P. and Dasgupta, B. (2013). A direct variational method for planning monotonically optimal paths for redundant manipulators in constrained workspaces. *J. Robot. Auton. Syst.*, 61, pp. 209–220.
- Siegwart, R. and Nourbakhsh, I. (2004). Introduction to autonomous mobile robots. In MIT Press.
- Simmons, R. (1996). The curvature-velocity method for local obstacle avoidance. *Proc. of the IEEE Int. Conf. Robot.*, 4, pp. 3375–3382.
- Sleumer, N. and Gurman, N. (1991). Exact cell decomposition of arrangements used for path planning in robotics. Swiss Federal Institute of Technology.
- Song, P. (2002). A potential field based approach to multi-robot manipulation general robotics. In *IEEE Int. Conf. Robot., ICRA'02*.
- Stacey, A., Jancic, M. and Grundy, I. (2003). Particle swarm optimization with mutation. In *IEEE Evol. Computat., CEC'03*, pp. 1425–1430.
- Stegall, P., Winfree, K., Zanotto, D. and Agrawal, S. (2013). Rehabilitation exoskeleton design: Exploring effect of the anterior lunge degree of freedom. *IEEE T. Robot.*, 29, pp. 838–846.
- Stuckler, J., Steffens, R., Holz, D. and Behnke, S. (2013). Efficient 3d object perception and grasp planning for mobile manipulation in domestic environments. *J. Robot. Auton. Syst.*, 61, pp. 1106–1115.
- Suranga, H. (2006). Distributed online evolution for swarm robotics. In *Auton. Agent. and Multi-Ag.*
- Tansel, Y., Yurdakul, M. and Dengiz, B. (2013). Development of a decision support system for robot selection. *Robot. Cim-Int. Manuf.*, 29, pp. 142–157.
- Taylor, N., Mohamad, M. and Dunnigan, M. (2006). Articulated robot motion planning using ant colony. In *Proc. IEEE Int. Conf. Optimization. Intel. Sys.*, pp. 690–695.
- Theodoridis, T. and Hu, H. (2006). The fuzzy sarsa (?) learning approach applied to a strategic route learning robot behaviour. In *IEEE/RSJ Int. Conf. Intell. Robots Sys. (IROS'06)*, pp. 1767–1772.
- Trueba, P., Prieto, A., Bellas, F., Caamano, P. and Duro, R. (2013). Specialization analysis of embodied evolution for robotic collective tasks. *Robot. Auton. Syst.*, 61, pp. 682–693.
- Valenci, R., Morta, M. and Andrade-Cetto, J. (2013). Planning reliable paths with pose slam. *IEEE T. Robot.*, 29, pp. 1050–1059.
- Vesterstrom, J. and Riget, J. (2002). Particle swarms extensions for improved local, multi-modal and dynamic search in numerical optimization. Master's thesis. Department of Computer Science, University of Aarhus, Aarhus C, Denmark.
- Vitiello, N., Lenzi, T., Roccella, S., Rossi, S. D., Cattin, E., Giovacchini, F., Vecchi, F. and Carrozza, M. (2013). Neuroexos: A powered elbow exoskeleton for physical rehabilitation. *IEEE T. Robot.*, 29, pp. 220–235.
- Wang, Y. and Chirikjian, G. (2000). A new potential field method for robot path planning. In *Proc. 2000 IEEE Int. Conf. Robot.*, pp. 977–982.
- Wang, L. (2002). Computational intelligence in autonomous mobile robotics — a review. In *Proceedings of 2002*

International Symposium on Micromechatronics and Human Science, 2002. *MHS 2002*, pp. 227–235. doi: 10.1109/MHS.2002.1058040.

- Webster, S., Walls, J., Whitcomb, L. and Eustice, R. (2014). Decentralized extended information filter for single-beacon cooperative acoustic navigation: Theory and experiments. *IEEE T. Robot.*, 29, pp. 957–974.
- Williams, R. and Sukhatme, S. (2013). Constrained interaction and coordination in proximity-limited multiagent systems. *IEEE T. Robot.*, 29, pp. 930–944.
- Williams, S. B., Newman, P., Dissanayake, M. and Durrant-Whyte, H. (2000). Autonomous underwater simultaneous localisation and map building. In *Proc. IEEE Int. Conf. Robot.*, San Francisco, USA, pp. 1143–1150.
- Włodzisław, D. (2007). What is computational intelligence and what could it become? *Challenges for Computational Intelligence*, pp. 1–11.
- Yamamoto, K. and Okada, M. (2013). Control of swarm behaviour in crossing pedestrians based on temporal/spatial frequencies. *Robot. Auton. Syst.*, 61, pp. 1036–1048.
- Yang, E. and Gu, D. (2004). Multi-agent reinforcement learning for multi-robot systems: A survey. In Technical report, Department of Computer Science, University of Essex Technical Report CSM-404, UK.
- Yang, C. and Simon, D. (2005). A new particle swarm optimization technique. In *18th International Conference on Systems Engineering, ICSEng 2005*, pp. 164–169.
- Yangmin, L. and Xin, C. (2005). Mobile robot navigation using particle swarm optimization and adaptive nn. In *Springer, ICNC 2005*, pp. 628–631.
- Yuan-Qing, Q., De-Bao, S., Ning, L. and Yi-Gang, C. (2004). Path planning for mobile robot using the particle swarm optimization with mutation operator. In *Proceedings of the Third International Conference on Machine Learning and Cybemetics*.
- Zacharia, P., Xidias, E. and Aspragathos, N. (2013). Task scheduling and motion planning for an industrial manipulator. *Robot. Cim-Int. Manuf.*, 29, pp. 449–462.
- Zhu, Y., Zhang, T., Song, J. and Li, X. (2013). A hybrid navigation strategy for multiple mobile robots. *Robot. Cim-Int. Manuf.*, 29, pp. 129–141.
- Zhu, D. and Latombe, J. (1991). New heuristic algorithms for efficient hierarchical path planning. *IEEE T. Robot. Autom.*, 7(1), pp. 9–20.
- Zou, A., Hou, Z., Fu, S. and Tan, M. (2006). Neural networks for mobile robot navigation: A survey. In Berlin, Heidelberg, Springer-Verlag, pp. 1218–1226.

¹Oxford Dictionaries: <http://www.oxforddictionaries.com/definition/english/navigation>.

²Vocabulary Dictionary: <http://www.vocabulary.com/dictionary/localization>.

³Cyclopaedia: <http://en.cyclopaedia.net/wiki/Navigation-problem>.

Chapter 26

Selected Automotive Applications of Computational Intelligence

Mahmoud Abou-Nasr, Fazal Syed and Dimitar Filev

With the increasing demand on the energy resources and the drive towards environmentally friendly vehicles, embedded computers have been long employed within the complex automotive architecture for various tasks like engine control, anti-lock braking, fault detection, and diagnostics. Our automobiles have evolved over the years to be more powerful, easier to drive and control, safer and more connected than ever. Recently, computational intelligence (CI) paradigms have found their way in our automobiles, helping us navigate, park, and intelligently control the cruising speeds of our cars. Intelligent drive assist features and autonomous vehicles are becoming a part of our current vocabulary. In this chapter, we introduce the CI paradigm within the framework of the other traditional computational paradigms. We then present applications of CI in automotive. The first application discusses a NN that learns, from sample data, how to detect intruders who may be hiding in the vehicle based on their heartbeat. The second is about NNs that learn to optimize fuel consumption while maintaining an ultra-capacitor state of charge at an average target level. The third application is about a fuzzy rule-based system for minimizing high voltage batteries power limits. In the last application, we present hybrid learning models for learning the preferences of the drivers, in the context of the driver aware vehicle concept.

26.1. Introduction

As much as computers and computational devices are part of our everyday life, we still aspire for a future in which we do not have to program them. One answer to this was the logic programming paradigms in which we tell our computers our goals and let them figure out ways to achieve these goals. Even with some successes in early expert systems, there is still a lot to be desired. Despite of the advances in computational speed and algorithms, computers are still no match to our brains in some tasks like hand-eye coordination; face recognition, handwriting recognition, and natural speech recognition to name a few. Also, our brains are far better in handling uncertainty and noisy inputs and are remarkably adaptable to ever changing contexts.

With inspirations from nature, biology, cognition, and computational theories a set of computational paradigms were born to address these tasks, that are challenging to our traditional computational paradigms yet seemingly done relatively at ease inside our brains. An umbrella term for this set of paradigms is computational intelligence (CI). CI covers disciplines such as artificial neural networks, fuzzy logic, and evolutionary computation. It embraces nature inspired algorithms that are derived from the principles and foundations of ant colonies, swarm intelligence, and artificial immune system. It has applications in many fields such as image and speech processing, natural language processing, computer vision, and data mining.

In the following sections, we will show several examples of CI applications in the automotive industry. The first two examples cover NN-based CI applications. We will start by an application in which a NN is learned from example data how to detect an intruder hiding in a vehicle based on his/her heartbeat. Next, we present NNs that are learned to optimize the fuel economy in a vehicle with a hybrid powertrain (internal combustion engine and an ultra-capacitor), while maintaining the average state of charge of the ultra-capacitor at a preferred target level. The third example CI application relates to fuzzy control as a tool for incorporating additional subjective knowledge and intelligence to the conventional hybrid electric vehicle (HEV) control algorithms. In our last CI application example, we present a Hybrid Model combining a set of discrete Markov chains and an associated fuzzy rule-base of contextual conditions. We discuss the structure of the Hybrid Model and the methodology for learning and predicting driver preferences and its application to the problem estimating the next driving destination based on the information about the trips between the most frequent stop locations.

26.2. Vehicle Intrusion Detection

The first application that we present is detecting an intruder, who may be lurking in a vehicle, waiting to ambush an unsuspecting lonely driver as he/she is entering in. We sought a system that may alleviate this scenario and allows one to query his/her vehicle, before approaching it, to make sure that the vehicle is vacant of intruders. The system we present is based on detecting the intruder heartbeat. The system utilizes signals from a sensor that responds to the intruder heartbeat, breathing, or both (Eagen *et al.*, 2008).

This detection problem is not straight forward and has many challenges. One challenge is the wide range of weather conditions and the variety of contextual surroundings which may affect and interfere with the sensor signals and can even excite the sensor into quasi-periodic oscillations that resemble a beating heart. Another is the wide range of heart beats and breathing patterns of different individuals. In addition, the space for packaging and resources such as memory and CPU available for embedded system implementation are limited, favoring compact signal processing algorithms. The problem is further compounded by the absence of a known signature differentiating an unoccupied vehicle from an occupied one. Figure 26.1 illustrates several five second intervals of conditioned sensor signals, some of them are taken while the vehicle was occupied (top panel) and the others are taken while the vehicle is unoccupied (bottom panel).

A block diagram of an intruder detection system that is based on recurrent neural networks is shown in Figure 26.2.

The first block in the diagram is a sensor which is a highly sensitive accelerometer with a wide dynamic range capable of responding to the minute vibrations caused by the human heart when carefully placed in a proper location inside the vehicle. Next, the signal is filtered and amplified in the signal conditioning block. The conditioned signal is then sampled and converted to a digital signal in the A/D block and then preprocessed (mainly scaled) in the preprocessing block in order for the trained neural network in the NN block to render its decision on whether the vehicle is occupied or not.

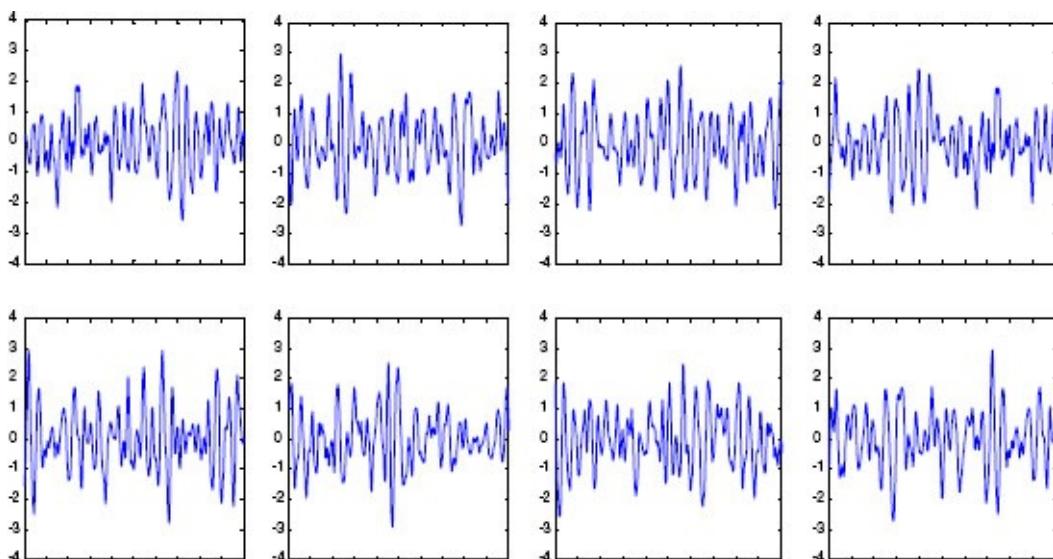


Figure 26.1: Various five seconds intervals of conditioned sensor signals representing an occupied (top panel) and an unoccupied (bottom panel) vehicle. It is difficult to visually discern a distinguishing signature for occupancy. The horizontal axis is time (0.5 s), and the vertical axis is the voltage (mv) proportional to the accelerometer signal.

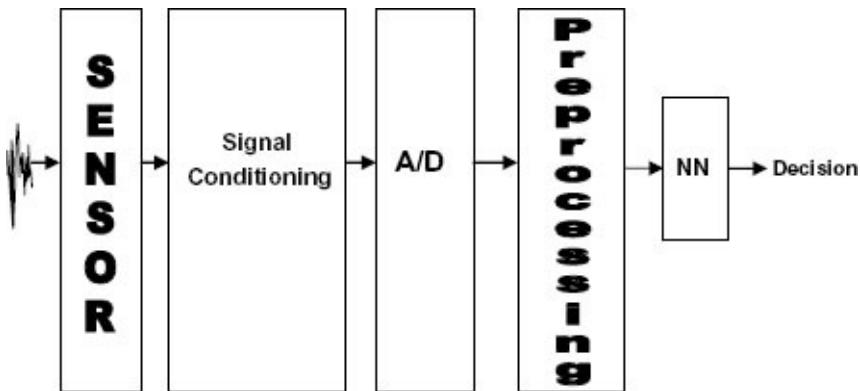


Figure 26.2: Block diagram of the intruder detection system.

26.2.1. Data-Driven Intruder Detection Neural Network Development

In a data-driven problem solving approach like neural networks data acquisition is of paramount importance. Methods and procedures for collecting and accurately labeling the data have to be decided upon and followed meticulously to ensure successful neural network development. Mislabelled data can affect NN training process and can drive the conclusions in the wrong direction. In addition, the collected dataset should be representative of all possible conditions that the detection NN may encounter in real-life.

For this problem, several instrumented vehicles were used to acquire data representing a variety of individuals, of different ages, genders, weights, heart rates who hid in different locations inside the car. Data acquisition sessions were structured so that recordings of when the car was occupied and when it was empty were collected in consecutive sessions or at least within a very short interval of time to minimize the potential for varying external conditions that may mislead the NN training process. This process was repeated in different locations both in Europe and in the United States e.g., parking lots, garages, and different levels of parking structures. Also it was repeated in different weather conditions (wind, rain, snow) and varying proximities to different traffic patterns (heavy, moderate, and low or no traffic).

We have experimented with different recurrent neural architectures in terms of number of layers and number of nodes within each layer. The node activation functions in all of our experiments/tests were bipolar sigmoids. Our initial experiments found that we can obtain about the same performance from all architectures ($\sim 95\%$ accuracy, $\sim 2.4\%$ false negative). Embedded implementation for vehicle deployment imposes constraints on CPU computations and memory utilization. Hence, we focused more on smaller architectures in our development. As expected, we have found that the detection performance of the NN is directly related to the richness and variety of the data in the training set. The recurrent networks were trained with extended Kalman filter (EKF) training algorithms. For more detail and description of this training method please see Feldkamp and Puskorius (1998).

26.2.2. Intruder Detection NN Performance Examples

In Figure 26.3, we show the response of a trained NN with an architecture that is comprised of one input which is the conditioned and normalized sensor signal, six bipolar sigmoid nodes which are also fully recurrent in the first layer and three bipolar sigmoid nodes in the hidden layer and one bipolar sigmoid node as an output node. Panel (a) is the instantaneous (every 1/100 s) response of this network to the sensor signal that is shown in panel (b) from an occupied car. The NN indicates the detection of a person in the vehicle, with a positive output value. Panel (c) on the other hand is the instantaneous response of the same trained network to the sensor signal that is shown in panel (d) from an unoccupied car. In this case the NN indicates that it did not detect a person with a negative output value.

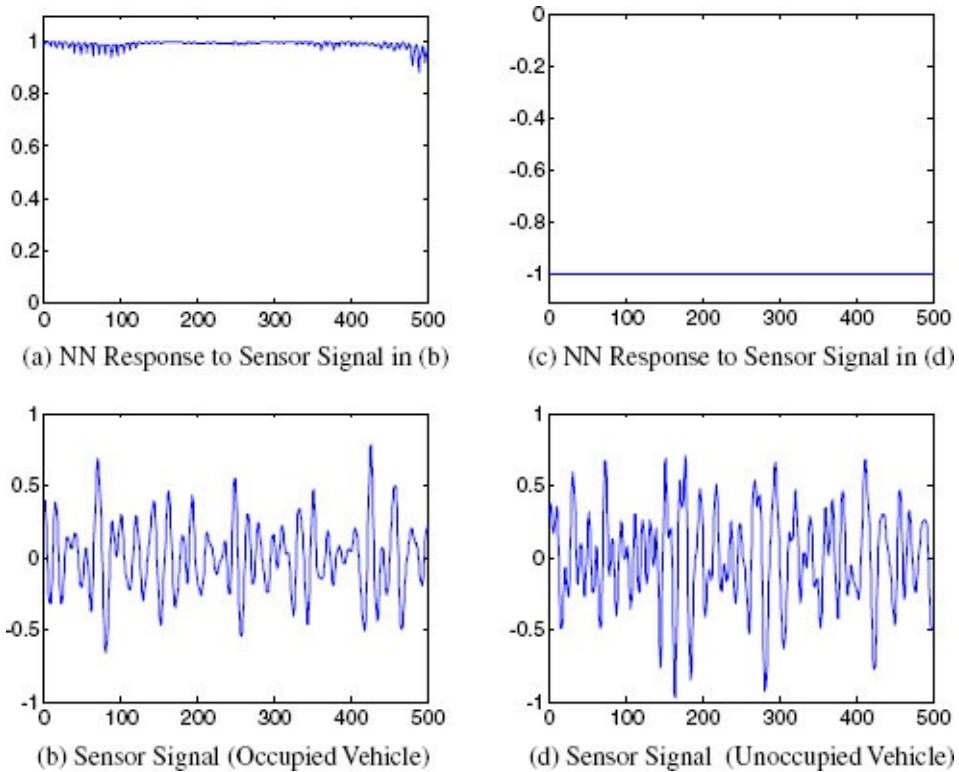


Figure 26.3: Trained neural network response to sensor signals. Panel (a) is the normalized NN Response to the normalized conditioned sensor signal from an occupied vehicle that is shown in Panel (b). Panel (c) is the normalized NN response to the normalized conditioned sensor signal from an unoccupied vehicle that is shown in panel (d). The horizontal axis in all figures is time (0.01 s). The vertical axis in figures (c) and (d) is the normalized NN output while the vertical axis in figures (b) and (d) is the normalized NN input representing the conditioned sensor signal.

26.3. Energy Management in a Hybrid Powertrain

The next application that we present is the problem of energy management in a hybrid powertrain with an ultra-capacitor as shown in [Figure 26.4](#). In this configuration, the primary energy source is an internal combustion engine (ICE) and the power to the wheels may be delivered through either the ICE or an electric motor connected to the ultra-capacitor or both.

Ultra-capacitors have high specific power and efficiency even at low temperatures, a quick response, efficient regenerative braking energy acceptance, and long projected cycle and calendar life. On the other hand, they have low energy storage capacity (low specific energy) and they self-discharge which may limit their use in serious hybrid applications. Unlike HEVs with conventional batteries, ultra-capacitors can be easily discharged during regular driving. Obviously, when an ultra-capacitor is discharged, it cannot supplement the energy power which may lead to drivability issues of the sort of a degraded response during an acceleration event, especially if the primary energy source (ICE in this case) is downsized or unable to respond quickly. A fully charged ultra-capacitor is not desirable either, as it will not be able to harvest energy from regenerative braking events.

For these reasons, incorporating ultra-capacitors in a hybridization strategy has its own challenges which could be even harder than the ones encountered in typical HEV's with batteries.

To overcome the ultra-capacitor limitations, HEV powertrain control strategies must optimally manage their charging and discharging behavior while driving. In this regard, a control strategy aims at prescribing a power split between the ICE engine and the electric motor in such a way as to sustain (on average) the state of charge of the ultra-capacitor, while attaining reduction in fuel consumption. The strategy must be sophisticated enough to be able to incorporate knowledge of the driving patterns, vehicle conditions, and possibly road terrain.

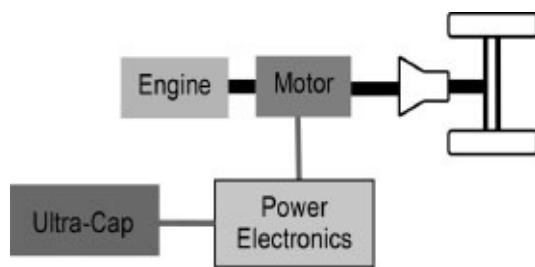


Figure 26.4: Schematic of an HEV configuration with an ultra-capacitor.

Researchers have explored different techniques to achieve these objectives in energy management control strategies, including dynamic programming (DP), rule-based, and model predictive control. In the following we elucidate a technique for achieving these two objectives in an energy management strategy for HEV with ultra-capacitor.

The technique is based on recurrent (dynamic) NNs which are employed in two modalities: A plant model and an optimizer. The plant model NN learns the information

regarding fuel consumption and the ultra-capacitor state of charge as a function of the vehicle speed, requested wheel power, and power split factor between the ICE and the electric motor. An approximation to the optimum value of this split factor is determined by the optimizer NN online (as the vehicle is driven) in real-time, after training the optimizer network offline on sample drive cycles.

NNs have successful applications in scientific, industrial, and financial applications as classifiers, pattern recognizers, and in regression and function approximation. In control applications they have been traditionally employed for model identification when analytical models are not feasible or too complex. Researchers have also proposed frameworks for applying NNs beyond their traditional successful applications. One such application is approximate dynamic programming (ADP) for obtaining approximate solutions to DP problems. In Abou-Nasr and Filev (2013); Balakrishnan *et al.* (2008); Prokhorov and Wunsch (1997); Werbos (1992), example frameworks for obtaining these solutions with NNs are presented. In the following, we will formulate the hybrid powertrain energy management problem as a DP problem. Then, we will obtain an approximate solution to it with NNs.

26.3.1. ADP with Recurrent Neural Networks

In deterministic, discrete, dynamic programming problems, an objective function J that is the sum of a utility function values U over a horizon of N sequential events is to be minimized (or maximized).

$$J = \sum_{n=1}^N U(\mathbf{x}(n), \mathbf{u}(n)). \quad (1)$$

The sequence of controls (policy) $\mathbf{u}^*(n)$ that minimizes (or maximizes) J in Equation (1) as a function of the system states $\mathbf{x}(n)$ is the what we seek as a solution to the dynamic programming problem, and is obtained by the Bellman principle of optimality recursion:

$$J(\mathbf{x}(n)) = \max(U(\mathbf{x}(n), \mathbf{u}(n)) + (J(\mathbf{x}(n+1))). \quad (2)$$

From Equation (2), it follows that in order for the sequence $\mathbf{u}^*(n)$ to be optimal; all the individual controls in each step of the sequence have to be optimal. Another way of stating this is that we can obtain the sequence $\mathbf{u}^*(n)$ by searching for the individual optimum control for the upcoming step. With a recurrent (dynamic) NN, one may approximate with high fidelity the utility function U . This recurrent NN will have the inputs of $\mathbf{x}(n)$ and $\mathbf{u}(n)$ and an output that represents the utility function $U(n)$ as depicted in [Figure 26.5](#).

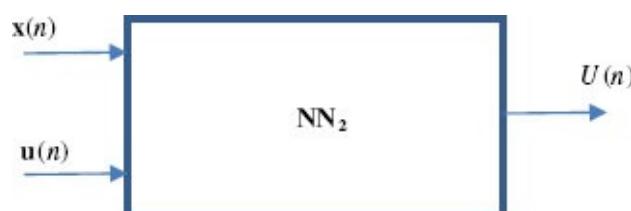


Figure 26.5: Approximating the utility function U with a recurrent NN.

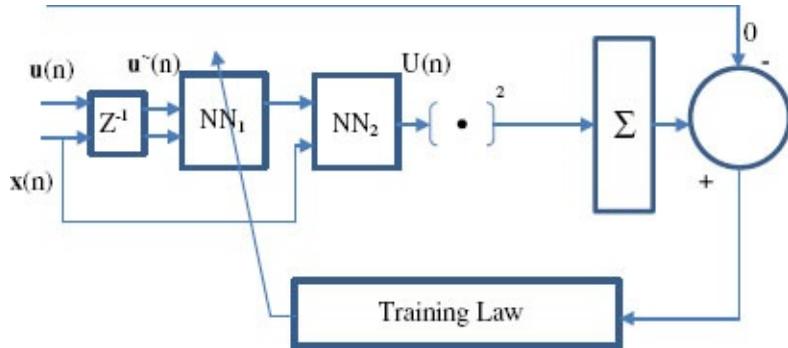


Figure 26.6: Approximate solution to a dynamic programming problem with NNs. The Z^{-1} is a unit time delay.

With the help of another recurrent NN that is initialized to random weights that we shall refer to as the optimizer network in Figure 26.6, we transform the problem into a supervised learning problem. In this supervised learning problem, we are training the optimizer network to find the sequence $u(n)$ that produces a sum of the square of the utility function $U(n)$ that is as close as possible to zero, since the minimum of a quadratic function is greater than or equal to zero. This sequence is an approximate solution to the dynamic programming problem. Handling problem constraints can be readily done by including them within the objective function weighted by Lagrange multipliers.

26.3.2. Hybrid Powertrain Energy Management with Neural Networks

The problem of hybrid powertrain energy management with an ultra-capacitor can be posed as a dynamic programming problem in which the objective is to minimize the powertrain fuel consumption over a driven trajectory, while maintaining the ultra-capacitor state of charge at a certain target level.

$$J(n) = \sum_{n=1}^N F(n) \quad n = 1 \rightarrow N,$$

s.t.

$$\frac{1}{N} \sum C(n) = C_T \quad n = 1 \rightarrow N, \quad (3)$$

$F(n)$ is the fuel consumption rate at instance n , $C(n)$ is the state of charge of the ultra-capacitor at instance n , C_T is the ultra-capacitor target average state of charge and N is the duration of the drive cycle.

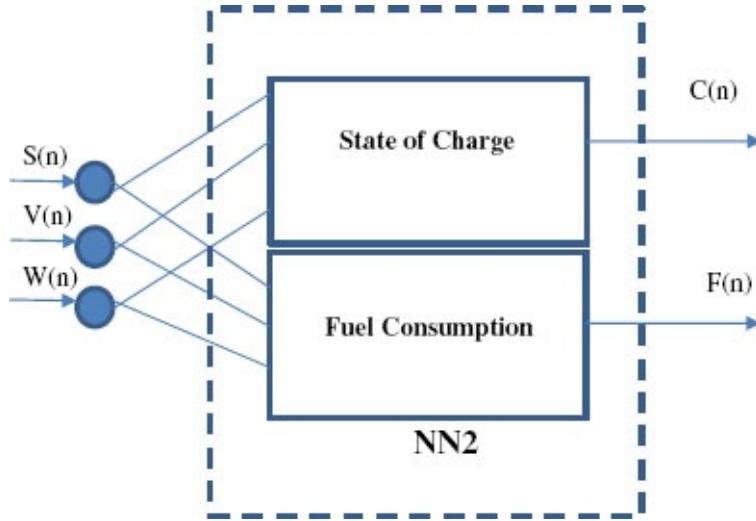
The fuel consumption rate which is also a function of the vehicle speed $V(n)$, the wheels power $W(n)$, and the split factor $S(n)$. The utility function neural network can be extended from Figure 26.5 to this particular problem as in Figure 26.7 where,

$$u(n) = S(n), \quad (4)$$

$$x(n) = (V(n), W(n)), \quad (5)$$

$$U(n) = F(n). \quad (6)$$

In order to implement the constraint on the ultra-capacitor state of charge, we first train another recurrent neural network to represent the functional relationship between the state of charge and the inputs of $S(n)$, $V(n)$, and $W(n)$ as depicted in [Figure 26.7](#). Equation (7) depicts how the output of the state of charge network is combined with the output of the fuel network to form the output of the Σ block of [Figure 26.6](#).



[Figure 26.7:](#) The NN_2 neural network for hybrid powertrain energy management.

$$sum = F^2(n) + \lambda(\bar{C}(n) - C_T)^2, \quad (7)$$

λ is a Lagrange multiplier and $\bar{C}(n)$ is obtained by Equation (8).

$$\bar{C}(n) = \alpha C(n) + (1 - \alpha)\bar{C}(n - 1), \quad (8)$$

where $\bar{C}(n)$ is the exponentially weighted moving average of $C(n)$ over the last $(1/\alpha - 1)$ samples.

26.3.3. Comparison of the ADP with NN to Dynamic Programming (DP)

In the following we compare the approximate solution obtained by the trained NN to the solution obtained by DP. For the drive cycle of [Figure 26.8](#), the split factor between the ICE and the ultra-capacitor as prescribed by DP is shown in [Figure 26.9](#) and the one prescribed by the NN is shown in [Figure 26.10](#). The corresponding state of charge of the ultra-capacitor for the DP solution is shown in [Figure 26.11](#) and the one corresponding to the NN solution is shown in [Figure 26.12](#).

With the prescribed controls (split factor) shown in [Figure 26.10](#), the NN has saved 10% of fuel, and the DP solution has saved 13% of fuel as compared to driving with ICE all the time. The savings could be improved further by improving the NN_2 models of fuel consumption and the ultra-capacitor state of charge. On the other hand, the trained neural optimizer allows us to obtain an approximate solution for a DP problem online on a

sample by sample basis, as opposed to the traditional DP solutions which require the knowledge of the whole path before solving the optimization problem, and can be impractical if the problem has more than a few dimensions. Thus, we conclude that the our approximate solution to the hybrid powertrain energy management problem is very competitive with the dynamic programming solution in terms of fuel savings, but also has advantages in terms of applicability in real-world problems that more often than none will have high dimensionality, and seldom afford us the luxury of previewing the path ahead nor the time traditionally required by DP to find a solution.

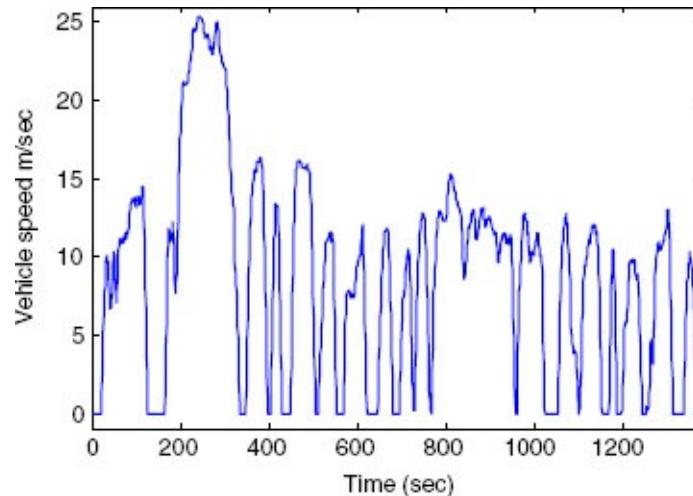


Figure 26.8: Drive cycle employed in the comparison of the DP solution to the NN solution.

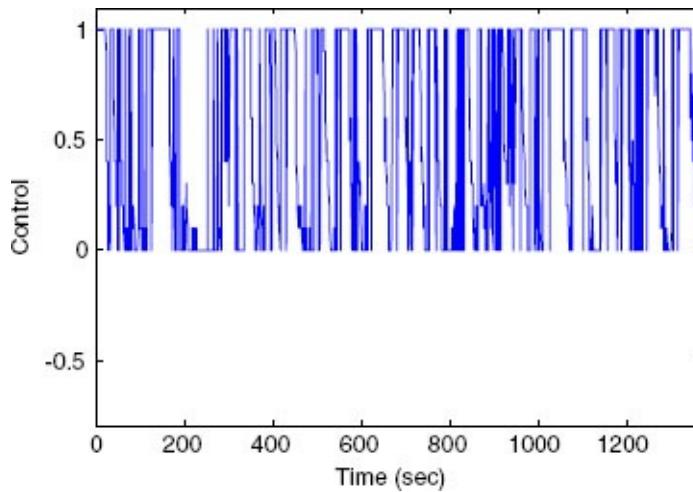


Figure 26.9: Split factor (control) prescribed by DP over the drive cycle of [Figure 26.8](#).

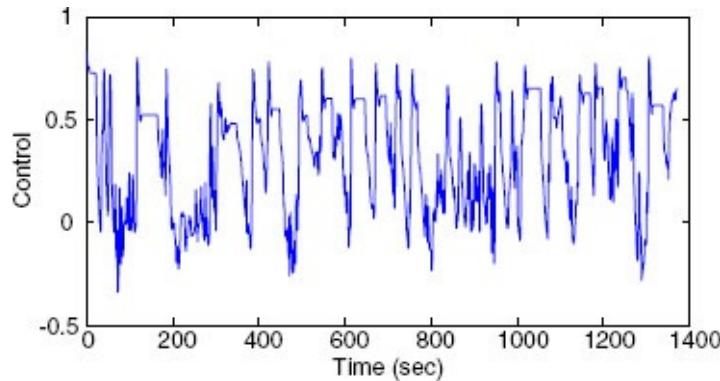
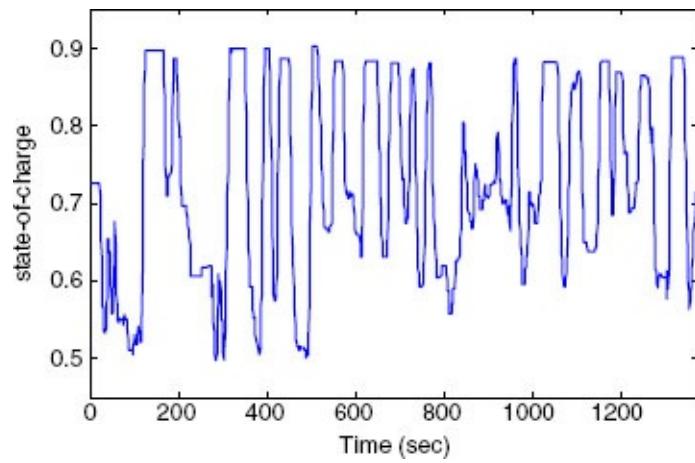


Figure 26.10: Split factor (control) prescribed by NN_1 over the drive cycle of [Figure 26.8](#).



>Figure 26.11: State of charge prescribed by DP over the drive cycle of [Figure 26.8](#).

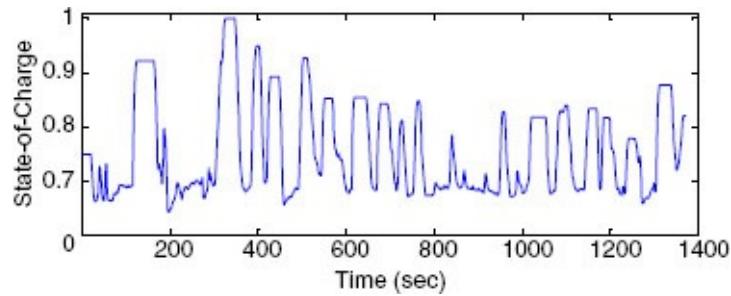


Figure 26.12: State of charge prescribed by NN_1 over the drive cycle of [Figure 26.8](#).

26.4. A Fuzzy Rule-Based Control System for Minimizing High Voltage Battery Power Limits Violations

In a conventional gasoline internal combustion engine based vehicle, the delivered engine torque or power can be different from commanded engine torque or power. When the perceived acceleration by the driver, as a result of the delivered torque or power, does not match what he or she is expecting, then the driver compensates for it by modifying his or her foot position on the accelerator pedal. In such systems, the control system does not need to estimate engine torque or power for delivering desired vehicle performance and any mismatch in the performance is compensated by the driver accordingly. However, in a power-split HEV, the wheel torque command is calculated from the actual engine torque or power. If the engine power can be estimated perfectly, then the control system can command or deliver the wheel torque such that the battery power limits are not violated. However, if the engine power or torque estimate is not accurate, then, the wheel torque command is incorrect and the high voltage (HV) battery power limits may be violated. Even in this scenario, any mismatch in desired vehicle acceleration can be compensated by the modifying the accelerator pedal, but the battery power violation cannot be addressed by the driver. Thus a control system is used to ensure that battery power limits are not violated, while meeting the driver desired vehicle acceleration or performance (McGee *et al.*, 2007).

The driveline vibrations or disturbances can result in driveline oscillations or vehicle speed oscillations (Farshidianfar *et al.*, 2001). Power-split is more prone to this issue since there is direct mechanical coupling between engine, generator, and motor to the driveline. As there is no mechanism of passive damping of such disturbances in power-split HEV, this requires use of sophisticated active damping control system approach to suppress the driveline oscillations in a power-split hybrid electric vehicle. As shown in Syed *et al.* (2009), such an approach is very effective in damping the driveline oscillations. Since part of such an approach requires the use of high-order narrow band notch filter resulting in real-time filtering of actual engine power, such filtering results in inaccuracy or mismatch of engine power estimate during transient event due to the phase delays introduced by filtering. These inaccuracies in the engine power estimate can ultimately, under extreme conditions result in violation of HV battery power limits.

In this section, an intelligent controller that can differentiate between various driving conditions and minimize HV battery power limit violations is described. This approach for improving the HV battery power controls in power-split HEV uses fuzzy weight-scheduling to determine appropriate weight for the Feedback Weighted Narrow Band Notch Filter used for engine power estimation based on the system's operating conditions. This fuzzy logic-based controller's performance is validated through simulation environment. The fuzzy logic-based controller's behaviors in simulation environment for the driving conditions that severely impacts HV battery power control or results in

violations are compared with the performances of the original simple weighted engine power estimation controller (SWEC) Syed *et al.* (2009).

26.4.1. HV Battery and Power Limitations

There are three power producing devices in a power-split HEV powertrain—the generator, the motor, and the engine (McGee *et al.*, 2007; Syed *et al.*, 2006). A vehicle system controller (VSC) control system determines the desired engine speed, desired engine torque, and desired wheel torque to meet the driver demand.

For a given driver demand, this system also has two degrees of freedom which provides more choices available in the selection of desired engine speed (due to the e-CVT nature of power-split architecture) and desired engine torque (due to the motor) as compared to conventional vehicles. The existence of extra degree of freedom also imposes additional constraint on power limit that must be met. Power-split HEV architecture consists of the HV battery, the generator, and the motor inverters, and the HV wiring that connects the two. The HV battery acts as a buffer on the HV bus. Any difference in the power consumed by the two electric machines is reflected as power discharged from or power charged into the HV battery. Due to the nature of the HV battery, there are limits on how much power can be put into or taken out of the HV battery (McGee *et al.*, 2007; Syed *et al.*, 2006). Violation of these limits can result in damage of the HV battery and/or reduction of HV battery life, both of which can have serious consequences in terms of vehicle warranty cost. It is important in the power split HEV that the HV battery's discharge and charge power limits violations are minimized (McGee *et al.*, 2007).

26.4.2. Advanced Wheel Torque Controller and its impact on HV Battery Power Limits Control

The wheel torque controller main control blocks are described in Syed *et al.* (2009). A sophisticated wheel torque control system was designed in Syed *et al.* (2009) that is capable of actively damping the driveline oscillations in a HEV. This controller for determining desired wheel torque is represented by the following simplified equations (Syed *et al.*, 2009):

$$P_{sys_min} = P_{elec_chg_lim} + P_{eng_min}, \quad (9)$$

$$P_{sys_max} = P_{elec_dch_lim} + P_{eng_max}, \quad (10)$$

$$P_{mod} = \begin{cases} P_{sys_max}, & \text{if } P_{drv} + P_{loss} > P_{sys_max} \\ P_{drv} + P_{loss}, & \text{if } P_{sys_min} \leq P_{drv} + P_{loss} \leq P_{sys_max} \\ P_{sys_min}, & \text{if } P_{drv} + P_{loss} < P_{sys_min}, \end{cases} \quad (11)$$

$$P_{elec_des} = \begin{cases} P_{elec_dch_lim}, & \text{if } f(P_{mod}, P_{eng_act}) > P_{elec_dch_lim} \\ f(P_{mod}, P_{eng_act}), & \text{if } P_{elec_chg_lim} \leq f(P_{mod}, P_{eng_act}) \leq P_{elec_dch_lim} \\ P_{elec_chg_lim}, & \text{if } f(P_{mod}, P_{eng_act}) < P_{elec_chg_lim}, \end{cases} \quad (12)$$

$$T_{wh_des} = \frac{P_{elec_des} - P_{loss} + P_{eng_act}}{\omega_{wheel}}, \quad (13)$$

where ω_{wheel} is the wheel speed, T_{wh_des} is the desired wheel torque, P_{loss} is the system power losses, $P_{elec_chg_lim}$ is the electrical charge power limit, $P_{elec_dch_lim}$ is the electrical discharge power limit, P_{eng_min} is the minimum engine power, P_{eng_max} is the maximum engine power, and P_{elec_des} is the desired electrical power such that it is bounded between the electrical charge power limit ($P_{elec_chg_lim}$) and the electrical discharge power limits ($P_{elec_dch_lim}$). Note that $P_{elec_chg_lim} \leq 0$ and $P_{eng_min} \leq 0$. Similarly, $P_{elec_dch_lim} \geq 0$ and $P_{eng_max} \geq 0$. Also note that $P_{elec_chg_lim}$ includes the HV battery charge power limit ($P_{bat_chg_lim}$) constraint and $P_{elec_dch_lim}$ includes the HV battery discharge power limit ($P_{bat_dch_lim}$) constraint. Equation (9) represents the minimum system power (P_{sys_min}) which is the sum of electrical charge power limit ($P_{bat_chg_lim}$) and the minimum engine power (P_{eng_min}). Similarly, Equation (10) represents the maximum system power (P_{sys_max}) which is the sum of electrical discharge power limit ($P_{bat_dch_lim}$) and the maximum engine power (P_{eng_max}). Equation (11) represents the modified driver power requested (P_{mod}) which is clipped sum of driver power (P_{drv}) and the system power losses (P_{loss}) and is limited between the minimum and maximum system power. Equation (12) represents the desired electrical power (P_{elec_des}) which is the difference between the modified driver power (P_{mod}) and the actual engine power (P_{eng_act}) and is limited between the electrical charge and discharge power limits. Equation (13) represents the desired wheel torque (T_{wh_des}) which is determined from the sum of desired electrical power (P_{elec_des}) and actual engine power (P_{eng_act}) minus the system power losses (P_{loss}) at a given wheel speed (ω_{wheel}).

The actual engine power in this controller, as described in Syed *et al.* (2009), is estimated as follows:

$$P_{eng_act} = \left(\frac{T'_{wh_des}}{k_{m2w}} \omega_{mot} + T_{mot} \omega_{mot} + T_{gen} \omega_{gen} \right) \times \left((1 - W) + W \frac{b_1 + b_2 z^{-1} + b_3 z^{-2} + b_4 z^{-3} + b_5 z^{-4} + b_6 z^{-5}}{a_1 + a_2 z^{-1} + a_3 z^{-2} + a_4 z^{-3} + a_5 z^{-4} + a_6 z^{-5}} \right), \quad (14)$$

$$W = \begin{cases} 1, & \text{if } f(\omega_{mot}) > 1 \\ f(\omega_{mot}), & \text{if } 0 \leq f(\omega_{mot}) \leq 1 \\ 0, & \text{if } f(\omega_{mot}) < 0, \end{cases} \quad (15)$$

where T_{mot} is the motor torque, T_{gen} is the generator torque, ω_{mot} is the actual motor speed,

ω_{gen} is the actual generator speed, k_{m2w} is the gear ratio between motor and the wheel (driveshaft), and W is the notch filtering weight for deciding the contribution of notch (band stop) filtering, with $b_1 \dots b_6$ and $a_1 \dots a_6$ representing the coefficients for poles and zeros of the sixth order narrow band notch filter respectively. The actual delivered wheel torque estimate, T'_{wh_des} , is the result of the desired wheel torque (T_{wh_des}) command from VSC to transaxle controller module (TCM) which is responsible for delivering the desired wheel torque. The actual delivered wheel torque estimate is described as $T'_{wh_des} = z^{-k} T_{wh_des}$ with k representing the time step delay in z-domain, which is the time that TCM takes to deliver the requested desired wheel torque command from VSC. Equation (15) can be rewritten as follows:

$$P_{eng_act} = P_{eng_act_inst} \times \left((1 - W) + W \frac{b_1 + b_2 z^{-1} + b_3 z^{-2} + b_4 z^{-3} + b_5 z^{-4} + b_6 z^{-5}}{a_1 + a_2 z^{-1} + a_3 z^{-2} + a_4 z^{-3} + a_5 z^{-4} + a_6 z^{-5}} \right), \quad (16)$$

where

$$P_{eng_act_inst} = \left(\frac{T'_{wh_des}}{k_{m2w}} \omega_{mot} + T_{mot} \omega_{mot} + T_{gen} \omega_{gen} \right) \quad (17)$$

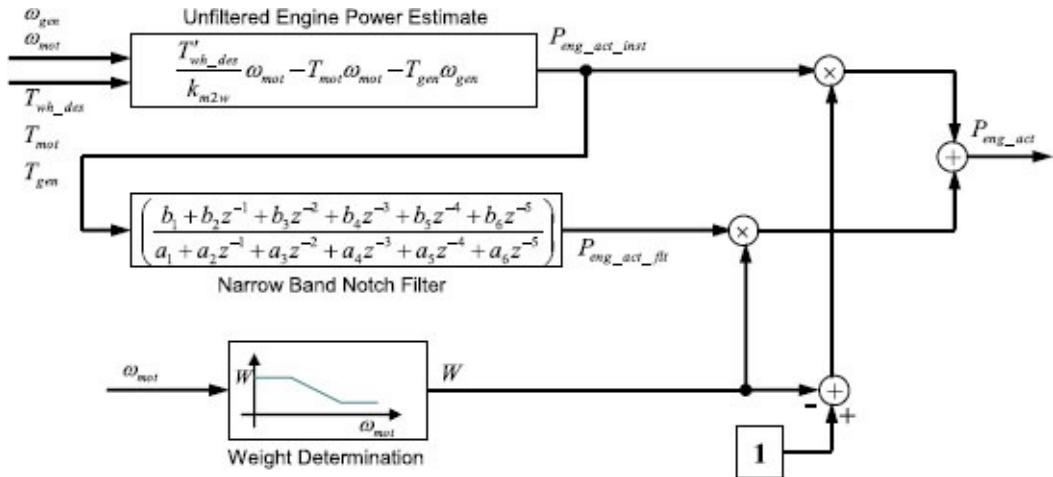


Figure 26.13: Simple weight engine power estimation control system.

Note that $P_{eng_act_inst}$ is the unfiltered actual instantaneous engine power.

Figure 26.13 shows the controller for determining the engine power estimate. Note that a Feedback Weighted Narrow Band Notch Filter is used for engine power estimation which determines the final weighted engine power estimate by taking the weighed contribution from the actual engine power estimate and the filtered engine power estimate (Syed *et al.*, 2009). As mentioned in Syed *et al.* (2009), one of the key factor to improve drivability (suppress driveline oscillations) is to achieve decoupling between VSC and TCM. This was achieved by using a narrow band notch (band stop) filter [$\times 4$]. It is clear that if the actual engine power were to change fast or quick, there will be an error between estimated engine power and actual engine power which can result in HV battery power limits violations under certain conditions. This can mathematically be written as follows:

$$P_{eng_act_error} = P_{eng_act_inst} - P_{eng_act}. \quad (18)$$

Since, actual HV battery power is the sum of generator power and the motor power, the actual HV battery power can be represented as follows:

$$P_{bat_act} = T_{mot}\omega_{mot} + T_{gen}\omega_{gen} + P_{loss}. \quad (19)$$

Solving Equations (19) and (17) results in following:

$$P_{bat_act} = P_{eng_act_inst} - \frac{T'_{wh_des}}{k_{m2w}}\omega_{mot} + P_{loss}. \quad (20)$$

Substituting the value of $P_{eng_act_inst}$ from Equation (18) into Equation (20) results in following equation for actual HV battery power:

$$P_{bat_act} = P_{eng_act_error} + P_{eng_act} - \frac{T'_{wh_des}}{k_{m2w}}\omega_{mot} + P_{loss}. \quad (21)$$

It is clear from Equation (21) that any non-zero value of $P_{eng_act_error}$ will appear in actual HV battery power, hence consequently can result in HV battery power limits violations (More probable when HV battery charge or discharge limits are low).

In order to address this issue, a simple weight determination process in the previously designed advanced wheel torque controller design (Syed *et al.*, 2009) was used to determine the value of the weight, W (between 0 and 1), in Equation (15) to minimize the inaccuracies in actual engine power estimation error ($P_{eng_act_error}$), and was calibrated and tuned as a function of vehicle speed (motor speed) to minimize the phase delays or essentially the inaccuracies in $P_{eng_act_error}$ thereby minimizing the effect on transient actual HV battery power (Note that if there are no inaccuracies in $P_{eng_act_error}$, then $P_{eng_act_error}$ will be zero). However, as we will demonstrate later that the use of such simple weighted engine power estimation to improve transient HV battery power control is not the most efficient approach for minimizing HV battery power limits violations.

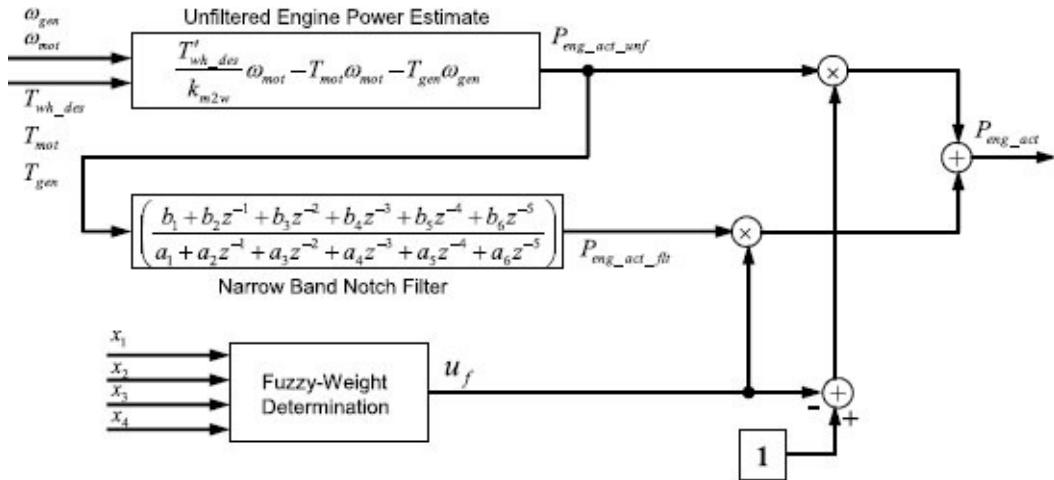
Use of simple weighted engine power estimation could result in violations of HV battery power limits. The amplitude of these violations are critical as they can degrade HV battery and impact HV battery life (McGee *et al.*, 2007). Since fuzzy-control systems do not require a mathematical model of the system to be controlled, and allow for the development of a knowledge based nonlinear controller, a desirable candidate for improving engine power estimation or minimizing HV battery power limits violations is to use fuzzy-logic based nonlinear control system.

26.4.3. Fuzzy Weight Scheduling Engine Power Estimation for the Power-Split HEV — Controller Design

As mentioned in [Section 26.4.2](#), to improve drivability (suppress driveline oscillations)

decoupling between VSC and TCM is necessary. This was achieved by using a narrow band notch (band stop) filter which results in phase delays and hence inaccuracies in the engine power estimation, especially during transient conditions. Since all driving conditions do not result in impacting drivability due to driveline oscillations, an intelligent controller can be designed which can detect such scenarios and conditions, and schedule appropriate weights for feedback weighted narrow band notch filter used for the estimation of engine power. This intelligent fuzzy weight-scheduling engine power estimation controller is shown in [Figure 26.14](#).

The fuzzy weight-scheduling engine power estimation controller block in [Figure 26.14](#) improves engine power estimation and hence results in improved HV battery power limits control. In other words, this controller is capable of minimizing HV battery power limits violations. The fuzzy weight-scheduling controller improves HV battery power limits management behavior in a power-split HEV by utilizing the human control knowledge and experience to intuitively construct an intelligent controller so that the resulting controller will emulate the desired control behavior to a certain extent (Ying, 2000). A multiple input, single output (MISO) Mamdani fuzzy weight-scheduling controller (Ying, 2000) is utilized. The formulation of the proposed fuzzy weight-scheduling controller can be described as follows:



[Figure 26.14](#): Fuzzy weight-scheduling engine power estimation control system.

$$P_{eng_act} = \left(\frac{T'_{wh_des}}{k_{m2w}} \omega_{mot} + T_{mot}\omega_{mot} + T_{gen}\omega_{gen} \right) \times \left((1 - u_f) + u_f \frac{b_1 + b_2z^{-1} + b_3z^{-2} + b_4z^{-3} + b_5z^{-4} + b_6z^{-5}}{a_1 + a_2z^{-1} + a_3z^{-2} + a_4z^{-3} + a_5z^{-4} + a_6z^{-5}} \right), \quad (22)$$

where u_f is the dynamically adjusted fuzzy scheduled weight output of the MISO fuzzy logic weight-scheduler controller.

Equation (22) can be re-written as follows:

$$P_{eng_act} = (1 - u_f) P_{eng_act_inst} \times u_f P_{eng_act_flt}, \quad (23)$$

where

$$P_{eng_act_inst} = \left(\frac{T'_{wh_des}}{k_{m2w}} \omega_{mot} + T_{mot} \omega_{mot} + T_{gen} \omega_{gen} \right), \quad (24)$$

$$\begin{aligned} P_{eng_act_flt} &= \left(\frac{T'_{wh_des}}{k_{m2w}} \omega_{mot} + T_{mot} \omega_{mot} + T_{gen} \omega_{gen} \right) \\ &\times \frac{b_1 + b_2 z^{-1} + b_3 z^{-2} + b_4 z^{-3} + b_5 z^{-4} + b_6 z^{-5}}{a_1 + a_2 z^{-1} + a_3 z^{-2} + a_4 z^{-3} + a_5 z^{-4} + a_6 z^{-5}}, \end{aligned} \quad (25)$$

where $P_{eng_act_inst}$ is the pre-filtered (or unfiltered) engine power estimate and $P_{eng_act_flt}$ is the filtered engine power estimate. Comparing Equations (14) and (22) it can be seen that fuzzy weight scheduling engine power estimation controller replaces the term W with the fuzzy scheduled weight output u_f .

At this point, it is important to emphasize that if $u_f = W$ under all operating conditions, then the fuzzy weight-scheduling engine power estimation controller reduces to SWEC.

To effectively design a fuzzy-logic based weight-scheduler, input variables, output variables, and input and output fuzzy sets need to be defined. Firstly, since the driver demand wheel torque (T_{mod}) can affect drivability, it was used as a predictive input to determine if filtering of the actual engine power is needed to achieve improved drivability. On the other hand, to ensure that HV battery power is within its limits effectively, the magnitude of the error between the pre-filtered engine power estimate and the final engine power estimate was selected as one of the other inputs to the fuzzy logic weight-scheduler. Similarly, HV battery power limits provide a buffer to amount of inaccuracy in engine power estimate that can be tolerated, therefore the difference between the HV battery discharge power limit ($P_{bat_dch_lim}$) and desired electrical power (P_{elec_des}) was used as one of the other inputs. Finally, the motor speed (or vehicle speed) was used as the final input, because it can be used as the predictor of expected drivability issue at various vehicle or motor speeds. The input variables for the controller can be written as follows:

$$x_1(n) = T_{mod}(n) = \frac{P_{mod}(n)}{k_{m2w} \omega_{mot}(n)}, \quad (26)$$

$$x_2(n) = |P_{eng_act_unf}(n) - P_{eng_act}(n)|, \quad (27)$$

$$x_3(n) = P_{bat_dch_lim}(n) - P_{elec_des}(n), \quad (28)$$

$$x_4(n) = \omega_{mot}(n). \quad (29)$$

The input fuzzy sets or membership functions for $x_1(n)$, $x_2(n)$, $x_3(n)$, and $x_4(n)$ are chosen as trapezoidal and triangular functions. The input fuzzy sets or membership functions for $x_1(n)$, $x_3(n)$, and $x_4(n)$ are chosen to be low and high trapezoidal functions where b_{1L} , b_{3L} , b_{4L} representing the low value of the trapezoidal function for the three input membership

functions and b_{1H} , b_{3H} , b_{4H} representing the high value of the trapezoidal functions for the three input membership functions. The input fuzzy sets or membership functions for $x_2(n)$ are chosen to be low, medium, and high trapezoidal/triangular functions where, b_{2L} and b_{2H} represents the low and high value of the trapezoidal functions respectively for the $x_2(n)$ input membership function, and b_{2M} , representing the medium value of the triangular functions for the $x_2(n)$ input membership function. The output fuzzy sets are of singleton type for the fuzzy scheduled weight representing maximum (max or h_{MX}), high (h_H), medium (med or h_{ME}), low (h_L), minimum (min or h_{MN}), and zero (h_Z) values.

The fuzzy rules are laid out in a manner such that they can distinguish between various HEV powertrain behaviors and makes decision of the current and future states of the powertrain. This way these fuzzy rules can anticipate and optimize to cover conditions where HV battery power violations can be improved while maintaining improved drivability of the vehicle. A sample description of these fuzzy rules is shown in [Table 26.1](#).

Among these fuzzy rules, some of the rules are intended to cover both steady state and transient conditions under various driving conditions, such as scenarios where HV battery limits or driver demand torque or vehicle speed (motor speed) conditions are important for acceptable drivability and reduced HV battery power limits violations. For example, Rules 1, where $x_1(n)$ is either low or high and $x_2(n)$, $x_3(n)$, and $x_4(n)$ are all low, depict a steady state, or close to a steady state predictive condition where, under current state, the error from filtering effect of the engine power estimate is low but the vehicle speed and the difference between the HV battery discharge power limit and the desired electrical power are also low. Under such conditions, if the driver were to perform a transient event the drivability will be severely affected due to driveline oscillations unless a very high or max (h_{MX}) fuzzy-scheduled weight is used. Similarly, rules where $x_1(n)$ is low, $x_2(n)$ is high, $x_3(n)$ is either low or high, and $x_4(n)$ is high describes conditions, where due to low driver demand torque and high vehicle speeds, the impact of zero fuzzy-scheduled weight (h_{MX}) on drivability is negligible but since it is a transient event, using such a lower fuzzy-scheduled weight will significantly reduce HV battery power limits violation or in other words, it will improve the transient HV battery power control. In summary, these rules provide a method to schedule weights for engine power estimation which can significantly improve HV battery power control (or reduce HV battery power limits violations) without compromising the vehicle's drivability.

Table 26.1: Some fuzzy rules for the fuzzy weight-scheduler.

Rule No.	If $x_1(n)$ is	if $x_2(n)$ is	if $x_3(n)$ is	if $x_4(n)$ is	Then $u_f(n)$ is	Example Explanation
1	low	low	low	low	Max	Steady state condition where drivability can be improved and there is no concern of HV battery power control.
2	low	low	low	high	Med	Steady state condition with less chance of drivability issues due to higher vehicle speed and so HV battery power control can be improved.
:	:	:	:	:	:	:
24	high	high	high	high	Min	Fast transient condition where HV battery power controls can be improved, as wheel torque is high but HV battery power limits are also available.

If Ω represents the total number of fuzzy rules ($\Omega = 24$ in our case) and $\mu_j(x_i, \tilde{A}_{i,j})$ represents the combined membership value from the antecedent of the j th rule, the output, $u_f(n)$, of the fuzzy weight scheduler can be written as follows (Ying, 2000) when the centroid defuzzifier is employed.

$$u_f(n) = \frac{\sum_{j=1}^{\Omega} \mu_j(x_1, \tilde{A}_{1,j}) \mu_j(x_2, \tilde{A}_{2,j}) \mu_j(x_3, \tilde{A}_{3,j}) \mu_j(x_4, \tilde{A}_{4,j}) \tilde{h}_j}{\sum_{j=1}^{\Omega} \mu_j(x_1, \tilde{A}_{1,j}) \mu_j(x_2, \tilde{A}_{2,j}) \mu_j(x_3, \tilde{A}_{3,j}) \mu_j(x_4, \tilde{A}_{4,j})}, \quad (30)$$

where x_i represents all the inputs ($i = 1 \dots 4$) and $\tilde{A}_{i,j}$ is a vector involving all the input fuzzy sets and \tilde{h}_j represents the output fuzzy set for the j th rule.

Using Equation (23), the complete fuzzy controller system for the engine power estimation can be described by the following equation:

$$P_{eng_act} = \left(1 - \frac{\sum_{j=1}^{\Omega} \mu_j(x_1, \tilde{A}_{1,j}) \mu_j(x_2, \tilde{A}_{2,j}) \mu_j(x_3, \tilde{A}_{3,j}) \mu_j(x_4, \tilde{A}_{4,j}) \tilde{h}_j}{\sum_{j=1}^{\Omega} \mu_j(x_1, \tilde{A}_{1,j}) \mu_j(x_2, \tilde{A}_{2,j}) \mu_j(x_3, \tilde{A}_{3,j}) \mu_j(x_4, \tilde{A}_{4,j})} \right) P_{eng_act_inst} \\ \times \left(\frac{\sum_{j=1}^{\Omega} \mu_j(x_1, \tilde{A}_{1,j}) \mu_j(x_2, \tilde{A}_{2,j}) \mu_j(x_3, \tilde{A}_{3,j}) \tilde{h}_j \mu_j(x_4, \tilde{A}_{4,j}) \tilde{h}_j}{\sum_{j=1}^{\Omega} \mu_j(x_1, \tilde{A}_{1,j}) \mu_j(x_2, \tilde{A}_{2,j}) \mu_j(x_3, \tilde{A}_{3,j}) \mu_j(x_4, \tilde{A}_{4,j})} \right) P_{eng_act_flt}. \quad (31)$$

26.4.4. Evaluation of Fuzzy Control System

To evaluate the improvements from the fuzzy weight-scheduling engine power estimation

controller, simulation environment to study the SWEC and fuzzy weight-scheduling engine power estimation controller (FWEC) is presented. After the description of the simulation environment, simulation results using the simple weighted and fuzzy weight-scheduling engine power estimation controller is presented. Finally experimental results using a test vehicle at test track is used to compare the simple weighted engine power estimation controller and fuzzy weight-scheduling engine power estimation controller. A validated simulation environment (Syed *et al.*, 2006) consisting of the vehicle model for a power-split HEV together with the subsystem models of the various hybrid specific systems was used for the testing and evaluation of this fuzzy weight-scheduling engine power estimation control system. In order to study the performance of the controllers, a custom test that could emphasize on the controller's HV battery power limits violations was devised. In this test, the vehicle is tested at the speeds of 10 mi/hour, 20 mi/hour, 30 mi/hour, 40 mi/hour, 50 mi/hour, and 60 mi/hour where the HV battery power limits are set to 0 kW, and the driver demand, driver accelerator pedal position, is changed from 0% to 100% and then from 100% back to 0% at a particular vehicle speed. This results in large changes in the desired engine power and hence will also result in large changes in the actual engine power. Such a test emphasizes on the change of actual engine power to monitor the HV battery power limits violations.

To study the improvements of the fuzzy weight-scheduling engine power estimation controller, it is compared to the performance of simple weighted engine power estimation controller. A validated simulation environment described in Syed *et al.* (2006) is used. To test the effectiveness of the fuzzy weight-scheduling engine power estimation controller versus the simple weighted engine power estimation controller, simulations using the test conditions described earlier were performed using these two controllers.

Figure 26.15 shows the HV battery power limits violations using the simple weighted engine power estimation controller. Note that in this test the HV battery power limits were set to zero (0 kW) which is a very extreme case and then the driver demand changed dramatically between its minimum and maximum by changing the driver accelerator pedal position from 0% to 100% and then back from 100% to 0% at a particular vehicle speed. These maneuvers results in large changes in the desired engine power and so also result in large changes in actual engine power. Plot [A] in Figure 26.15, shows the results of HV battery power violation during this maneuver at 10 mi/hour. Similarly other plots in Figure 26.16 shows the HV battery power violations at speeds of 20 mi/hour, 30 mi/hour, 40 mi/hour, 50 mi/hour, and 60 mi/hour. As a result of actual engine power change, the HV battery power limits violations ranged from about -7 kW to 7 kW or the average range of power limit violations (average of the magnitude of HV battery discharge power limit violation and HV battery charge power limit violation) was around 7 kW under worst cases. It is important to emphasize that such HV power limits violations can reduce or degrade HV battery life overtime.

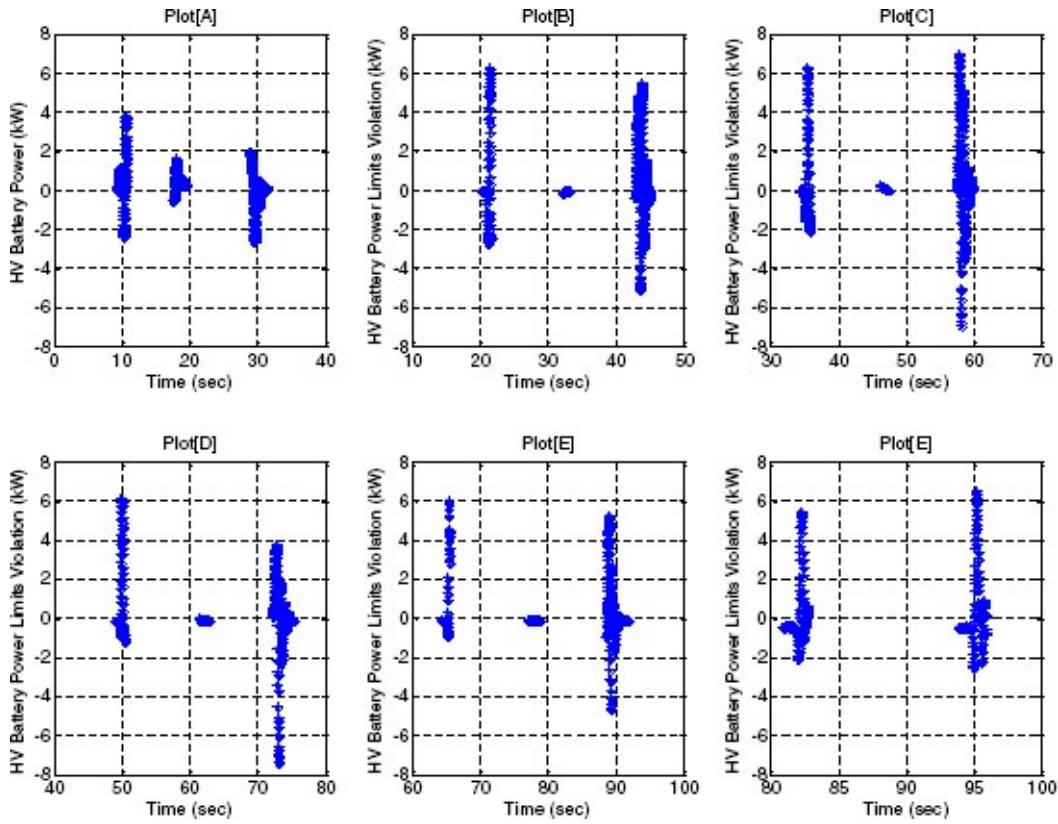


Figure 26.15: Simulation results for HV battery power limits violation using the simple weighted engine power estimation control system during the custom test. Plot[A] shows the results at vehicle speeds of around 10 mi/hr, plot[B] shows the results at vehicle speeds of around 20 mi/hr, plot[C] shows the results at vehicle speeds of around 30 mi/hr, plot[D] shows the results at vehicle speeds of around 40 mi/hr, plot[E] shows the results at vehicle speeds of around 50 mi/hr and plot[F] shows the results at vehicle speeds of around 60 mi/hr.

Next the simulations are performed for the same custom test using the fuzzy weight schedule engine power estimation control system. Figure 26.16 shows the HV battery power limits violations. It can be seen from these results that the HV battery power limits violations are reduced and they ranged on from about -5 kW to 5 kW or the average range of power limit violations was around 5 kW under worst cases. Such an improvement or reduction in HV power limits violations can improve HV battery life overtime. These simulation results are described in Table 26.2.

It can be seen from Table 26.2 that using the fuzzy weight-scheduling engine power estimation controller, improvements or reduction in HV battery power limits violations are seen at all vehicle speeds from 10 mi/hr to 60 mi/hr and the overall average power limit violations reductions ranged from 17 to 55%. Such huge improvement can provide ability to improving HV battery life and decreasing HV battery damages or degradations.

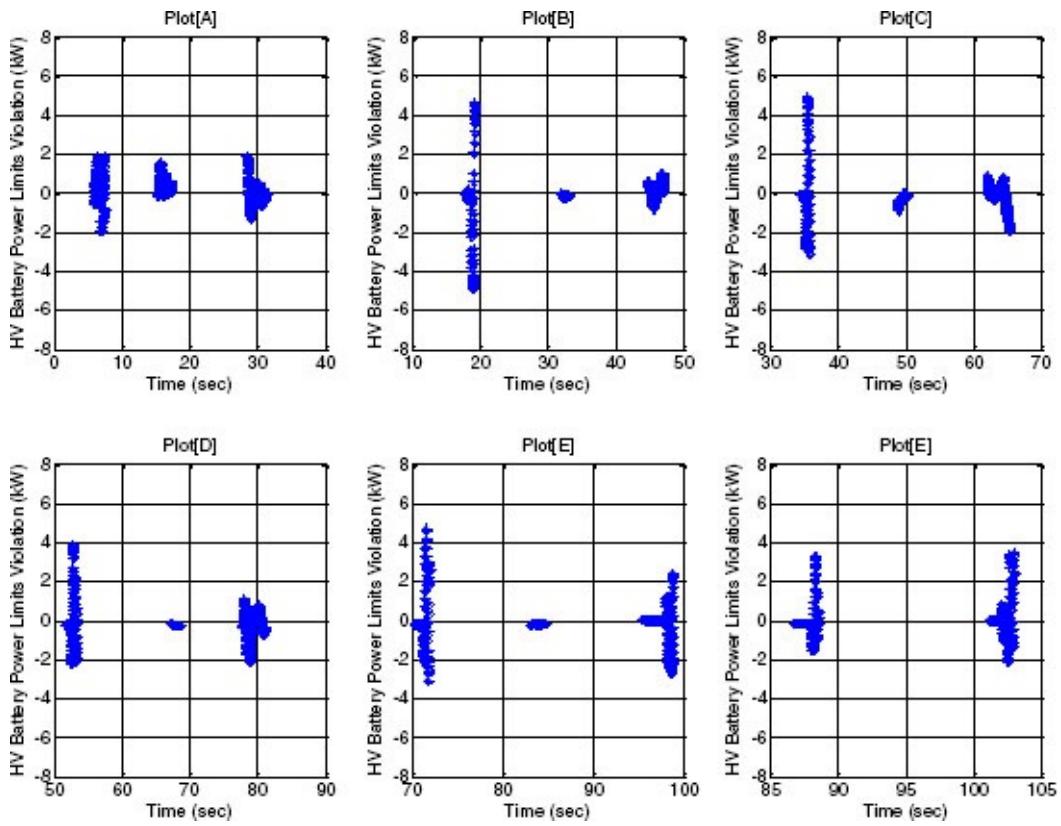


Figure 26.16: Simulation results for HV battery power limits violation using the fuzzy weight-scheduled engine power estimation control system during the custom test. Plot[A] shows the results at vehicle speeds of around 10 mi/hr, plot[B] shows the results at vehicle speeds of around 20 mi/hr, plot[C] shows the results at vehicle speeds of around 30 mi/hr, plot[D] shows the results at vehicle speeds of around 40 mi/hr, plot[E] shows the results at vehicle speeds of around 50 mi/hr and plot[F] shows the results at vehicle speeds of around 60 mi/hr.

26.5. Hybrid Models for Contextual Learning of Driver's Preferences

The concept of a “driver-aware” vehicle determines a type of vehicle that is capable to estimate and learn driver’s preferences. This presumes well-developed algorithms for perception and learning driver’s actions and decision. The learning mechanism is driver specific and is able to identify and summarize the driver preferred actions in different situations and states. The final goal is to incrementally develop and update families of models that characterize the driver under diverse set of conditions. The models are used by the vehicle systems to anticipate driver’s actions and to proactively respond/advise the driver for the most appropriate actions under specific circumstances. Examples include models for contextual learning preferred destinations, most likely music selection, level of acceptance of advisory information, etc.

Table 26.2: Simulation test results comparison for HV battery power limits (charge and discharge limit) violations using SWEC versus FWEC.

Vehicle Speed (mi/hr)	HV battery charge limit violation	HV battery discharge limit violation	HV battery average violation	HV battery range violation	HV battery charge limit violation	HV battery discharge limit violation	HV battery average violation	HV battery power limit violation
	using SWEC (kW)	using SWEC (kW)	using SWEC (kW)	using FWEC (kW)	using FWEC (kW)	using FWEC (kW)	using FWEC (kW)	improvement using FWEC (%)
10	-2.65	3.78	3.22	-1.92	1.89	1.91	40.68	
20	-5.14	6.26	5.70	-4.86	4.61	4.74	16.84	
30	-6.98	6.93	6.96	-3.10	4.92	4.01	42.39	
40	-7.41	6.10	6.76	-2.20	3.83	3.02	55.33	
50	-4.72	5.97	5.35	-3.15	4.71	3.93	26.54	
60	-2.56	6.48	4.52	-2.10	3.50	2.80	38.05	

One of the main advantages of introducing models for contextual learning is the opportunity for better utilization of the flexibility of the current vehicle systems. The goal is to satisfy not only the hypothetical nominal driver but to meet the requirements of as many as possible individual members of the customer group that is targeted by a specific vehicle model. An obvious area of implementation of this concept is the vehicle infotainment system. A good understanding of the driver preferred destinations and music/news/route selections under different conditions can introduce a level of intelligence that has a significant impact on the driver, while minimally affecting the critical vehicle attributes related to performance, comfort, and safety. One can expect that the final goal will be achieving personalized vehicle performance, i.e., performance that depends on the specific driver’s choice. In a broad sense this may lead to the creation of customizable and reconfigurable vehicles — an automotive equivalent of the growing trend of customization of information, news, and entertainment.

26.5.1. Switching Contextual Model of Most Frequent Destinations

One specific contextual model for estimating the most likely destinations was introduced in Filev *et al.* (2011). Alternative techniques for predicting most frequent routes and destinations have been proposed in Ashbrook and Starner (2003); Krumm and Horvitz (2012); Krumm *et al.* (2013). The contextual model assumes a finite set of M frequent destinations and provides an estimate for the next most likely driving destination. In this section we extend and generalize this model.

If we consider the selection of the next destination as a random process in which the decision for the next destination depends only on the current destination then we can model the mechanism of selecting new destinations as a Markov process with states corresponding to the set of destinations. For a finite set of destinations $S = \{\bar{x}_j, j = 1, \dots, M\}$ we model the decision-making process as a finite state Markov chain with transition probabilities (Figure 26.17)

$$\pi_{ij} = P(x^+ = \bar{x}_j | x = \bar{x}_i), \quad (32)$$

defining an $(M \times M)$ size transition probability matrix Π :

$$\Pi = \{\pi_{ij}, i, j = 1, \dots, M\}. \quad (33)$$

Consequently, the rows of the transition probability matrix are positive and sum to one, i.e.,:

$$\sum_{j=1}^M \pi_{ij} = 1, \quad i = 1, \dots, M; \quad \pi_{ij} \geq 0. \quad (34)$$

For a given $x = \bar{x}_i$ the next state inferred by the Markov chain is the one that maximizes the probability distribution that is conditioned by the state \bar{x}_i :

$$x^+ = \bar{x}_j, \quad \text{if } x = \bar{x}_i, \quad j \in \arg \max_t \pi_{it}. \quad (35)$$

In order to include the option of different decision strategies under alternative contextual conditions, e.g., time of the day, day of the week, the mood, and emotional state of the driver, the traffic, road, and weather conditions, etc., we introduced the Switching Contextual Model of Most Frequent Destinations (Filev *et al.*, 2011).

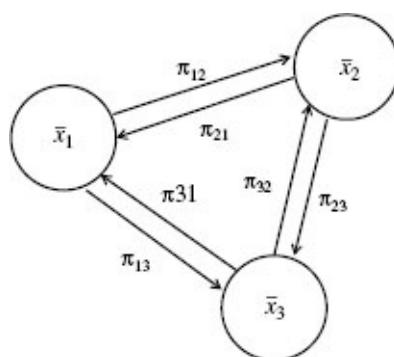


Figure 26.17: Markov chain model of most frequent destination (no context).

Due to the hybrid nature of this model — finite state Markov Model combined with continuous inputs — we partition the inputs into intervals. Assuming that the two inputs — *Time of the Day* and *Day of the Week* denoted by U_1, U_2 — are defined on universes u_1 and u_2 that are partitioned into S_1 and S_2 countable set of disjunct intervals $I_p, p = \{1, S_1\}$, and $I_q, q = \{1, S_2\}$ we can decompose the input space into a countable set of granules. The input granules are defined in the Cartesian product space $(u_1 \times u_2)$. The total number of granules is $S = S_1 S_2$. Each granule $\bar{u}_s = I_p \times I_q, s = \{1, S\}$, is formed by the Cartesian products of the intervals $I_p, p = \{1, S_1\}$, and $I_q, q = \{1, S_2\}$; the input granule \bar{u}_s expresses a combination of conditions belonging to a Cartesian product of intervals $I_p \times I_q$.

Using the granular decomposition of the input space we can define Contextual Model of Most Frequent Destination as a set of conditional probabilities:

$$\pi_{ij}^{(s)} = P(x^+ = \bar{x}_j | x = \bar{x}_i, U = \bar{u}_s), \quad i, j = \{1, m\}, s = \{1, S\}. \quad (36)$$

The model (36) represents probabilities of transitioning from the current destination \bar{x}_i to a new destination \bar{x}_j under the condition that the pair of random inputs $U = (U_1, U_2)$ take values from the input granule $\bar{u}_s = I_p \times I_q, s = \{1, S\}$.

It is easy to see that for all input pairs $U = (U_1, U_2)$ belonging to the same input granule, i.e., $U \in \bar{u}_s$, the contextual model is essentially identical to the Markov model $\Pi^{(s)}$ expressing the transitions between the Markov states under the condition defined by the granule \bar{u}_s . Thus, the contextual model can be viewed as composed by s switching Markov models $\Pi^{(s)}$ each one associated with one of the s granules of input conditions. This contextual model is equivalent to a collection of s Boolean logic rules with disjunct predicates:

$$\text{If } U \in \bar{u}_s \text{ Then } \pi_{ij}^{(s)} = P(x^+ = \bar{x}_j | x = \bar{x}_i), \quad i, j = \{1, m\}, s = \{1, S\}. \quad (37)$$

The role of the granular encoding of the input space is to determine a switching mechanism linking the current pair of conditions $U = (U_1, U_2)$ to the corresponding transition probability matrix $\Pi^{(s)}, s = \{1, S\}$. The specific transition probability matrix $\Pi^{(s)}$ defines the transitions between the states for that given pair of conditions. For a given pair of conditions $U = (U_1, U_2)$ and current state \bar{x}_i the next state \bar{x}_j predicted based conditional probability (36) can be expressed as follows:

$$\begin{aligned} j &= \arg \max_k (\Pi_{ik}^{(s)}), \quad \text{where } U_1 \in I_p \text{ and } U_2 \in I_q \text{ and} \\ s &= \text{ind} [\{(p, q)\} \equiv \{1, S_1\} \times \{1, S_2\}]. \end{aligned}$$

Similarly, for a given pair of conditions $U = (U_1, U_2)$, the elements π_{ij} of the corresponding transition probability matrix $\Pi^{(s)}$ are learned according to the algorithm for real time learning of Markov Models (Filev and Kolmanovsky, 2010):

$$\pi_{ij}^{(s)}(k) = \frac{F_{ij}^{(s)}(k)}{F_{oi}^{(s)}(k)} = \frac{F_{ij}^{(s)}(k-1) + \alpha(f_{ij}(k) - F_{ij}^{(s)}(k-1))}{F_{oi}^{(s)}(k-1) + \alpha(f_i(k) - F_{oi}^{(s)}(k-1))}, \quad (38)$$

where

$$F_{ij}^{(s)}(k) = F_{ij}^{(s)}(k-1) + \alpha(f_{ij}(k) - F_{ij}^{(s)}(k-1)), \quad (39)$$

$$F_{oi}^{(s)}(k) = F_{oi}^{(s)}(k-1) + \alpha(f_i(k) - F_{oi}^{(s)}(k-1)) \quad (40)$$

and

- (i) $F_{ij}^{(s)}$ is the moving average of all transition events $f_{ij}(k)$ from state \bar{x}_i to state \bar{x}_j (a transition event takes the values 1 or 0 if the transition occurs or not):
i.e.,:

$$f_{ij}(k) = \begin{cases} 1 & \text{if } x^+ = \bar{x}_j, x = \bar{x}_i \\ 0 & \text{otherwise} \end{cases} \quad (41)$$

- (ii) $F_{oi}^{(s)}$ is the mean the moving average of the transition events that start from state i :

$$f_i(k) = \begin{cases} 1 & \text{if } x = \bar{x}_i \\ 0 & \text{otherwise} \end{cases}. \quad (42)$$

The forgetting factor α assigns weight to the recent observed transition, exponentially distributes the rest of the weights, and defines the memory depth (the length of the moving window) of the weighted averaging aggregating operator. It can be shown that the memory depth K_α is approximately reciprocal to the forgetting factor, i.e., $K_\alpha = 1/\alpha$. Therefore, by properly choosing the parameter α we can calculate the transition probabilities within a specified moving window K_α addressing the possible nonstationarity of the data and updating the transition probabilities online. That is recursively calculated over the transition events belonging to the soft interval $\{k - K_\alpha + 1, k]$ where symbol $\{$ indicates a soft lower interval limit which includes values with lower indexes than $(k - K_\alpha)$ that have relatively low contribution.

The Switching Contextual Model assumes interval partitioning of the inputs. Its switching mechanism activates only one of the Markov models and cannot account for the cases when the conditions are very similar, e.g., 2:55 pm and 3:01 pm on weekend. It also cannot interpret efficiently the uncertainties in defining the context. In the following, we provide a generalization of the results from the previous section by introducing a fuzzy partitioning of the input variables.

26.5.2. Hybrid Contextual Model

We assume multiple factors determining the context, i.e., r inputs U_1, U_2, \dots, U_r defined

over the universes u_1, u_2, \dots, u_r that are partitioned into S_1, S_2, \dots, S_r fuzzy subsets $E_{11}, E_{12} \dots, E_{1s1}; \dots; E_{r1}, E_{r2} \dots, E_{rsr}$. This fuzzy partitioning decomposes the Cartesian input space $(u_1 \times u_2 \times \dots \times u_r)$ input space into a countable set of $S = \prod_{i=1}^r S_i$ granules $\bar{u}_s = E_{1s} \times E_{2s} \times \dots \times E_{rs}$. Each input granule \bar{u}_s expresses a combination of conditions belonging to the Cartesian product of the fuzzy subsets $E_{1s} \times E_{2s} \times \dots \times E_{rs}$.

Using the granular decomposition of the input space we can define the Hybrid Contextual Model as a family of rules with fuzzy predicates and a set of Markov models as consequents. The antecedents of the rules provide a probabilistic description of the context while the consequents represent the corresponding models of transitional probabilities that are associated with these conditions.

$$\begin{aligned} &\text{IF } U_1 \text{ is } E_{1s} \text{ AND } U_2 \text{ is } E_{2s} \text{ AND } \dots \text{ AND } U_r \text{ is } E_{rs} \\ &\text{THEN } \Pi^{(s)}, s = [1, 2, \dots, S]. \end{aligned} \quad (43)$$

This set of rules resembles the well-known Takagi–Sugeno model with the Markov models replacing the conventional linear equation type consequents. Following the theory of approximate reasoning, the Hybrid Contextual Model is a Markov model that is obtained by the weighted average of the individual models $\Pi^{(s)}$:

$$\Pi = \sum_{s=1}^S w_s \Pi^{(s)}, \quad (44)$$

with weights

$$w_s = \frac{\tau_s}{\sum_{j=1}^S \tau_j}, \quad s = [1, 2, \dots, S], \quad (45)$$

that are determined by the degrees of matching the inputs to the antecedents of the rules, i.e., the degrees of firing the rules τ_s :

$$\tau_s = \prod_{j=1}^r E_{js}(U_j), \quad s = [1, 2, \dots, S].$$

where by $E_{js}(U_j)$ is denoted the membership grade of the U_j th input with respect to the E_{js} th fuzzy subset (Yager and Filev, 1994).

Proposition. If each of the granules $\bar{u}_s = E_{1s} \times E_{2s} \times \dots \times E_{rs}$ is associated with a Markov model then the Hybrid Contextual Model is also a Markov model.

Proof: Assuming the transition probability matrices $\Pi^{(s)}, s = [1, 2, \dots, S]$ satisfy condition (3), then for each raw of the transition probability matrix Π of Hybrid Contextual Model Equation (44) we have:

$$\sum_{j=1}^M \pi_{ij} = \sum_{s=1}^S \sum_{j=1}^M w_s \pi_{ij}^{(s)} = 1, \quad i = 1, \dots, M. \quad (46)$$

We have used that the weights w_s are positive and sum to one as implied by (16).

According to Equation (45), the Hybrid Contextual Model interpolates and provides smooth transition between the Markov models that are associated with individual contexts. The switching contextual model can be considered a special case of it for the cases when the fuzzy subsets degenerate into disjunct intervals.

[Figure 26.18](#) shows an example of a collection of 56 Markov models of 10 frequent stop locations associated with a fuzzy partitioning of the two conditions — Time of the Day and Day of the Week into 8, respectively 7 fuzzy subsets. In reality not all of the 56 Markov models include the complete set of states since only some of the stop locations are visited under certain conditions. These active states along with their transition probabilities are estimated during the learning phase. For example, in [Figure 26.18](#) the models #3 and #56 do not have states #4 and #6, respectively #1, #2, and #9.

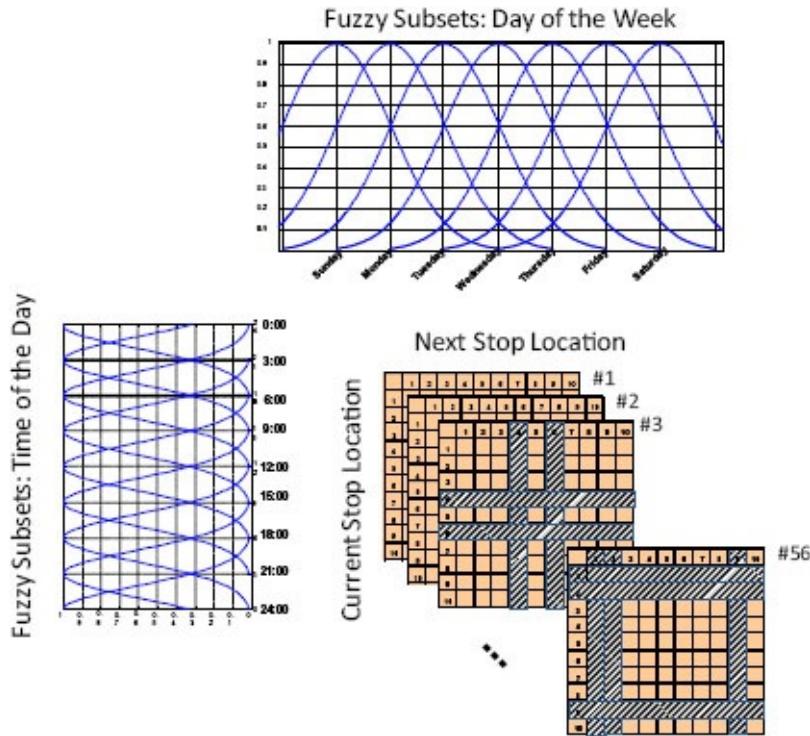


Figure 26.18: Hybrid contextual model of most frequent destinations—input/contextual space is granulated into 56 regions; corresponding Markov models include only the active Markov states (the visited stop locations under specific contextual conditions).

26.5.3. Online Learning the Hybrid Contextual Model

The interpolative nature of the Hybrid Contextual Model affects the algorithm for its learning from data. While the switching model can be learned as a collection of independent Markov models since only one of them is active when the specific set of conditions is satisfied, the Hybrid Contextual Model requires parallel updating of all of the individual Markov models.

During the learning the weights w_s by Equation (45) provide different interpretation of the new information about the current context and the transitions between the Markov states. In order derive the algorithm for learning the Hybrid Contextual Model we consider the following cases:

If the current input set of input values $U_1(k), U_2(k), \dots, U_r(k)$ completely matches a set of conditions that are captured by a granule, e.g., $\bar{u}_s = E_{1s} \times E_{2s} \times \dots \times E_{rs}$, then the corresponding Markov model transition probability matrix $\Pi^{(s)}$ is updated with the information for the specific transition between the according to expressions (38)–(42) assuming that a transition between states \bar{x}_i and \bar{x}_j has occurred; the rest of the transition probabilities in $\Pi^{(s)}$ remain unchanged. If the rest of the conditions are not matched, the remaining transition probability matrices $(t), t = 1, 2, \dots, S, t \neq s$, are not updated. Note that this assumption corresponds to the following set of weights:

$$w_s = 1; \quad w_t = 0, \quad t = 1, 2, \dots, S, \quad t \neq s. \quad (47)$$

If the current input set of input values $U_1(k), U_2(k), \dots, U_r(k)$ partially matches the conditions in multiple granules, then the corresponding Markov model transition probability matrices are updated with the information for the specific transition between the according to expressions (38)–(42) assuming that a transition between states \bar{x}_i and \bar{x}_j has occurred; the rest of the transition probabilities in $\Pi_{(s)}$ remain unchanged. Note that this assumption corresponds to a set of non-zero weights $w_s, s = 1, 2, \dots, S$.

We combine the three cases of matching by scaling the learning rate α in (39) and (40) with the corresponding weight $w_s, s = 1, 2, \dots, S$, as follows:

$$F_{ij}^{(s)}(k) = F_{ij}^{(s)}(k - 1) + \alpha w_s (f_{ij}(k) - F_{ij}^{(s)}(k - 1)), \quad (48)$$

$$F_{oi}^{(s)}(k) = F_{oi}^{(s)}(k - 1) + \alpha w_s (f_i(k) - F_{oi}^{(s)}(k - 1)), \quad (49)$$

$$\pi_{ij}^{(s)}(k) = \frac{F_{ij}^{(s)}(k)}{F_{oi}^{(s)}(k)} = \frac{F_{ij}^{(s)}(k - 1) + \alpha w_s (f_{ij}(k) - F_{ij}^{(s)}(k - 1))}{F_{oi}^{(s)}(k - 1) + \alpha w_s (f_i(k) - F_{oi}^{(s)}(k - 1))}. \quad (50)$$

The algorithm for online learning the Hybrid Contextual Model was validated on a dataset including 623 trips over 48 unique start–stop location combinations and incremental learning of the transition probabilities. The granularity of contextual conditions — Time of the Day and Day of the Week — was 3 hours and 1 day corresponding to 8, respectively 7 fuzzy subsets consistent with the partitioning shown in [Figure 26.18](#). The algorithm demonstrated prediction rate of 71.95%. Prediction accuracy varied between different drivers based on the percentage of repetitive trips and similarity of corresponding contextual information (Filev *et al.*, 2011).

26.6. Conclusions

In this chapter, we presented several diverse automotive applications of computational intelligence. Our goal was to demonstrate the potential of CI for solving wide range of practical problems related to driver safety (intrusion detection), neural net-based dynamic optimization of hybrid powertrain energy management of HEV that was coupled with an ultra-capacitor, fuzzy control of HV battery in power-split HEV, and learning driving patterns and prediction of the next destination. These cases clearly show that the methodology of computational intelligence combined with conventional techniques, e.g., dynamic programming, control theory, Markov models, etc., provides a strong foundation and opportunities for developing new original technical solutions, and innovative vehicle systems, and features.

References

- Abou-Nasr, M. A. and Filev, D. P. (2013). Dynamic adaptation of a vehicle's cruising speed with recurrent neural networks for enhancing fuel economy, 2013 *IEEE International Conference on Cybernetics (CYBCONF)*, pp. 249–254.
- Ashbrook, D. and Starner, T. (2003). Using GPS to learn significant locations and predict movement across multiple users. *Pers. and Ubiquit. Comput.*, 7(5), pp. 275–286.
- Balakrishnan, S. N., Ding, J. and Lewis, F. L. (2008). Issues on stability of ADP feedback controllers for dynamical systems. *IEEE T. Syst. Man Cy. B*, 38(4), pp. 913–917.
- Eagen, C., Feldkamp, L., Ebenstein, S. et al. (2008). System and method for detecting presence of a human in a vehicle, US20060089753 A1701/1.
- Farshidianfar, A. Ebrahimi, M. and Bartlett, H. (2001). Hybrid modelling and simulation of the torsional vibration of vehicle driveline systems. *P. I. MECH. ENG. D-J. Aut.*, 215, pp. 217–229.
- Feldkamp, L. A. and Puskorius, G. V. (1998). A signal processing framework based on dynamic neural networks with application to problems in adaptation, filtering, and classification. *Proc. IEEE*, 86(11), pp. 2259–2277.
- Filev, D. and Kolmanovsky, I. (2010). Markov chain modeling approaches for on board applications. *Proc. Am. Control Conf. (ACC)*, pp. 4139–4145.
- Filev, D., Tseng, F., Kristinsson, J. and McGee, R. (2011). Contextual On-Board Learning and Prediction of Vehicle Destinations, In *2011 Symp. on Intelligent Vehicles and Vehicular Systems, IEEE Symp. Comput. Intell.*, April 11–15, 2011, Paris, France.
- Krumm, J. and Horvitz, E. (2012). Some help on the way: Opportunistic routing under uncertainty. *Proc. 2012 ACM Conf. Ubiquit. Comput. UbiComp'12*, pp. 371–380.
- Krumm, J., Gruen, R. and Delling, D. (2013). From destination prediction to route prediction. *Journal of Location Based Services*, 7, pp. 98–120.
- McGee, R., Syed, F., Hunter, S. and Ramaswamy, D. (2007). Power control for the escape and mariner hybrids. *SAE 2007 World Congress and Exhibition, Paper No. 2007-01-0282*.
- Pettersson, M. and Nielsen, L. (2000). Gear shifting by engine control. *IEEE T. Cont. Syst. T.*, 8(3), pp. 495–507.
- Prokhorov, D. V. and Wunsch II, D. C. (1997). Adaptive critic designs. *IEEE T. Neural. Networ.*, 8(5), pp. 997–1007.
- Syed, F., Kuang, M., Czubay, J. and Ying, H. (2006). Power-split hybrid electric vehicle model development and validation. *IEEE T. Veh. Technol.*, 55(6), pp. 1731–1747.
- Syed, F., Ying, H. and Kuang, M. (2009). Active damping wheel-torque control system to reduce driveline oscillations in a power-split hybrid electric vehicle. *IEEE T. Veh. Technol.*, 58(9), pp. 4769–4785.
- Werbos, P. J. (1992). Approximate dynamic programming for real-time control and neural modeling, In White, D. A. and Sofge, D. A. (eds.), *Handbook of Intelligent Control*, New York: Van Nostrand Reinhold, Ch. 13.
- Yager, R. and Filev, D. (1994). *Essentials of Fuzzy Modeling and Control*. New York: John Wiley & Sons.
- Ying, H. (2000). *Fuzzy Control and Modeling*. New York: IEEE Press.

Index

A

- abstraction, 54
- accumulated one-step ahead error, 119
- accuracy, 121
- Ant Colony Optimization (ACO), 349, 587–599, 671, 682–683
- active learning, 115
- actor/critic structure, 425
- Adaptive Control of Thought-Rational (ACT-R), 328
- adaptive control, 401
- adaptive forgetting, 105
- adaptive network-based fuzzy inference system (ANFIS), 154
- adaptive gain, 229
- adaptive law, 226, 229
- adaptive MA, 608, 612
- adaptive two-layered hybrid algorithm, 616
- addition-to-one-unity, 112
- aggregation of fuzzy sets, 30
- AIS applications, 570
- AIS textbooks, 571
- AIS conferences, 570–571
- AIS multidisciplinarity, 562
- AIS tools, 565–568
- alarm monitoring and rationalization, 807
- algorithm, 865–867, 869, 888, 892, 895–896
- alleles, 514
- all-pairs classification, 82
- alpha-cut, 16, 71

alternative fuzzy FDD methods, 269
anomaly detection, 549, 551–555
antecedent, 188, 191, 193–194, 197, 199, 203
applications of ECoS, 763, 766
architecture of ECoS, 759–760
artificial immune network, 559
Artificial Immune System (AIS), 549
A-TLHA, 616–619
attractor, 338
automated modeling, 149
automotive, 867–868, 889, 896
average percentual deviation, 122
axiomatic set theory, 45
axis parallel rules, 74

B

Bat algorithm (BA), 587–589, 596
backpropagation algorithm, 573
Baldwinian evolution, 615
Battery, 876–878, 880–881, 883–889, 896
Bayesian and Gaussian tools, 565–566
Bee Swarm Optimization (BCO), 587–588, 594–595, 598, 600
Big Data, 68
bioinformatics, 763
bio-inspired, 671
biological immune system inspirations, 548
Boltzmann machine, 323
Braitenberg vehicle, 345

C

canonical evolutionary MA, 622
canonical microcircuit, 334

capacitor, 865, 867, 871–874, 896
cardinality, 18, 54
catalytic cracking, 816
catalytic reforming, 814
categories of logical operators, 28
Canonical Evolutionary Memetic Algorithm (C-EMA), 622–623, 626
crude oil distillation, 811
crude switch, 813
curse of dimensionality, 103, 105, 283, 551
cybernetics, 319
cylindrical extension, 34
Center of Gravity (CoG), 71, 145
characteristic contour/spread, 75
chromosome, 513
class confidence levels, 80
class score, 83
classification rules, 79
classification, 185, 203, 862
clonal algorithm, 556, 559
clonal selection, 556–559
clustering, 174
Co-evolutionary MAs (co-EMA), 624–626
co-evolutionary MA, 624
cognitive science, 317
coincidence horizon, 213, 215
compactness, 191, 193, 195
competitive exclusion, 548
completeness, 112, 192–193
complexity reduction, 111
computability, 54

computation, 866, 868
computational neuro-genetic models, 394
computational demands, 118
confidence intervals, 109
confidence, 185, 189, 191, 193
conflict, 109
confusion matrix, 121
connectionism, 319
consequent, 185–186, 188–189, 193–195
consequents learning, 92
conserved self pattern recognition algorithm, 543
consistency, 53, 111, 193
constraint, 868, 872–873, 877–878
contextual learning, 888–889
contextual model, 889–896
continuous stirred-tank reactor, 220
control horizon, 214
control law, 215, 229
control, 865–866, 870–888, 896
controller tuning, 808
convergence analysis/ensurance, 107
convexity, 17
cooperation, 349
core, 15
cortical area, 334
cortical column, 333
cortical layer, 333
coverage, 38, 112
crossover, 161

D

danger model, 554–556
Darwinism, 512
database integrity, 651
database learning, 646–647
data reconciliation, 804
dataset, 868
data stream mining, 70
data stream, 84, 180
data-driven FDD, 242
data-driven methods, 138
Distributed Control System (DCS), 803
decision boundary, 79
decremental learning, 104
deep learning, 323
defuzzification, 71, 190–191
degree of purity, 80
degree of membership to a fuzzy set, 6
delay, 215
delta learning rule, 322
Direct Evolutionary Memetic Algorithm (D-EMA), 620–623
dendritic cell algorithm, 554, 572
Dynamic Evolving Neural-Fuzzy Inference Algorithm (DENFIS), 388–390, 758, 762, 765
Dynamic eSNN (DeSNN), 391, 395
destination, 866, 889–891, 894, 896
detection, 865–869, 896
diploid cell, 514
direct evolutionary MA, 621
direct fuzzy model reference adaptive control, 225
distinctiveness of granulation, 53
distinguishability, 111

disturbances, 228

deoxyribonucleic acid (DNA), 513–514

drift, 198–199

drift and shift in data streams, 103

drift handling, 103

driver, 866, 876, 879, 883–886, 888–890, 896

dynamic model identification, 808

dynamic programming, 407

dynamic update of learning parameters, 117

dynamic, 867, 870–872, 874, 876, 882, 896

E

electroencephalogram (EEG), 343

embedded, 196

embodied cognition, 326

emergency manager, 807

energy, 865, 870–873, 876, 896

episodic memory, 340

epistasis, 514

epsilon dominance, 653

equality, 18

error bars, 109

error bound, 107

ETS+, 803, 807, 809, 814, 817, 819

evaluation measures, 119

evolution strategies, 520

evolution, 510

evolutionary algorithms, 509

evolutionary computation, 509

evolutionary MA, 621

evolutionary operation, 517

evolutionary optimization of ANN, 747, 749
evolutionary optimization of ECoS, 765
evolutionary programming, 518
evolving ANN architecture, 748
evolving ANN connection weights, 749–750
Evolving Connectionist System (ECoS), 386–388, 757, 765
Evolving Fuzzy Neural Network (EFuNN), 388, 761
Evolving Fuzzy System (EFS), 154, 180
Evolving Intelligent Systems (EIS), 68
evolving Spiking Neural Networks (eSNN), 390
experimentation function, 55
experimentation, 56
expert systems, 324, 801
extended fuzzy classification model, 80
extended Takagi–Sugeno fuzzy systems, 75

F

Fault Detection (FD), 245
fault identification, 245
fault isolation, 245
FDD benchmark, 250
feature, 195–197, 199–200
feature importance levels, 106, 113
feature reactivation, 105
feedback control, 401
filter, 195
finite axiomatization, 49
first-order MA, 616–617
first survey, 548
fitness function, 160
flooding, 812

functional imaging (fMRI), 343
forgetting factor, 87
frame of cognition, 37
fuel economy, 866
fuel consumption, 865, 870–871, 873–874
fully-train-and-then-test, 120
functional mappings, 55
furnace temperature, 812
fuzzy, 865–866, 876–877, 881–888, 892–894, 896
fuzzy basis function networks, 73
fuzzy classification model, 79
fuzzy classifier, 79–83, 252, 662
fuzzy control, 672–673, 682, 685
fuzzy controllers, 697
fuzzy decision trees, 78
fuzzy FDD, 249
fuzzy inference, 71, 75
fuzzy model-based predictive control, 213
fuzzy models, 697–699, 704, 714, 716–717, 720, 722–724
fuzzy pattern trees, 78
fuzzy recognition, 551, 566
fuzzy regression system, 656
fuzzy relations, 32
fuzzy rule and condition selection, 644, 646
fuzzy rule interpretability, 640
fuzzy rule learning, 636, 644
fuzzy rule-based classifier, 638
fuzzy rule-based system, 137, 637
fuzzy sets, 6, 565–566
fuzzy system generation, 635

fuzzy systems, 70–73, 75–79, 81–82, 84, 567

G

General Adaptive Two-Layered Hybrid Algorithm (GA-TLHA), 618–620

Gaussian function, 73, 191, 200

Gauss–Newton method, 90–91

General Evolutionary Memetic Algorithm (G-EMA), 626–627

gene, 514

general adaptive two-layered hybrid algorithm, 618

general type-2 fuzzy set, 76

generalized multivariate Gaussian distribution, 74

generalized policy iteration, 414

generalized rules, 74

generalized Takagi–Sugeno fuzzy system, 73–74

Genetic Algorithm (GA), 525, 671, 734–735

Genetic Fuzzy System (GFS), 159

genetic learning, 166

genetic linkage, 514

genetic tuning, 162

genetic-based machine learning, 557

genotype, 514

global learning, 87

Granular Computing (GrC), 43, 733

granularity, 39

granulation of data, 36

granulation process, 50

granule, 891, 893

gravitation concept, 81

ground truth, 115

Grow and Learn (GAL) networks, 755–757

H

haploid cell, 514
heartbeat, 865–866
Hebbian learning rule, 320
Hessian matrix, 87
heuristics, 607
hybrid electric vehicle (HEV), 870–871, 876–878, 881, 884, 886, 896
hidden Markov model, 347
hierarchical fuzzy systems, 78–79
Hopfield network, 342
human problem solving, 44
human-centered information processing, 43
human-inspired evolving machines, 114
hybrid AIS, 567–568
hybrid approaches, 564–569
hybrid controllers, 808
hybrid intelligent systems, 731
hybrid models, 888, 893
hybrid optimal adaptive controller, 431
hybrid, 865–866, 870–873, 876–877, 886
hydrogen partial pressure, 814–815
hydro-treating, 813
hyper-computation, 56
hyper-heuristic algorithm, 607–608
hypervolume, 653

I

ignorance, 108–109
image processing, 346, 764
imbalanced dataset classification, 650, 662
imbalanced learning problems, 82
immune response, 549

immune system cognitive paradigm, 562
immune system ecology, 570
immunoengineering, 549, 569–570
inclusion, 19
incremental, 198–199, 204
incremental clustering, 69
incremental learning, 84–85
incremental optimization of nonlinear parameters, 89–91
incremental partitioning of the feature space, 92–95
incremental regularization, 107
incremental smoothing of consequents, 107
incremental/decremental active learning, 115–116
infectious non-self model, 551–554
inference, 186, 190–191, 201
infinite axiomatization, 47
information granulation, 44–45
information loss, 74
information theory, 565–566
integral reinforcement learning, 427
intelligence, 57
intelligent, 865, 877, 881–882, 889
intentionality, 325
interleaved-test-and-then-train, 119
internal model, 317
interpretability, 113
interpretability-accuracy tradeoff, 649, 659–660
interpretation of fuzzy sets, 7
interpretation of model predictions, 108
interval type-2 fuzzy set, 76
intruder, 866–869

intuitive set theory, 44

inverse covariance matrix, 93

inverse weighted Hessian matrix, 88

J

Jacobian matrix, 91

K

Kalman filter gain, 88

Kalman, 869

kernel functions, 565–566

kernel trick, 446, 478

kernel, 447–448

knowledge contraction, 92

knowledge expansion, 92, 113

knowledge-based approach, 139

L

Lagrange, 872, 874

Lamarckism, 511

Lamarckian evolution, 615, 624, 626

large category, 50

Latent Dirichlet Allocation (LDA), 347

life cycle assessment (LCA), 805

leakage, 229

learning algorithm, 443, 469, 618, 749–750, 757–762, 841

learning in dynamic environments, 68

learning rule, 322, 341–342, 362

learning, 187–188, 192, 865–866, 872, 888–889, 892, 895–897

least-squares error criterion, 86–87

local field potential (LFP), 343

leaky-integrate-and-fire neuron (LIF), 335, 391

linear, 187

Linear Matrix Inequalities (LMI), 218

linguistic interpretability, 111, 114

linguistic variables, 36

local learning, 87, 113

locus, 514

long-term depression, 321

long-term potentiation, 321

Liquefied Petroleum Gas (LPG), 815

Lyapunov function, 231

M

memetic algorithm (MA), 607–609, 612, 616–617, 619–630

machine learning systems, 114, 587

maintenance, 807

Mamdani fuzzy systems, 70–71

Mamdani-type, 188

Markov Decision Process, 348, 404

Markov, 866, 890–896

matrix, 890–891, 894–895

McCulloch–Pitts model, 319

Mean Absolute Error, 122

Mean of Maximum, 71

magnetoencephalogram (MEG), 343

membership function, 6–13

meme, 608–610

memeplex, 608–610

memetic evolution, 608, 611

memory systems, 329

memotype, 609

mental representation, 317

mereology, 51

meta-heuristic algorithm, 607
methodological solipsism, 325
model architecture, 69
model output increment, 215
model parameter adaptation, 85
model, 866, 871, 874, 881, 886, 888–896
model-based FDD, 246
modular Neural Networks, 731–732, 734, 741
motivation for constructive ANN, 752
Multivariable Predictive Control (MPC), 808–809
multilayer perceptron, 322
multi-model fuzzy classification model, 81–82
multi-objective evolutionary algorithms, 641–642
multi-objective evolutionary fuzzy system, 635, 643, 651, 653
multi-objective evolutionary optimization, 640
multi-objective, 732, 735
mutation, 161

N

Nash games, 435
Natural Language Processing, 331
Nature-inspired approaches, 567
Nature-inspired optimization, 697–699, 713, 723, 724
negative selection, 550–551
NeuCom, 397
NeuCube, 394
neuro-fuzzy hybrid intelligent systems, 385
Neural Networks (NN), 731–732, 812
neural, 866–869, 871–874, 896
neuro-fuzzy approach, 154
neuro-fuzzy systems, 77

neuromorphic implementations, 396
neuron, 332
next generation Evolving Intelligent Systems, 114
noise level, 103, 107
nonlinear, 187–188
non-parametric statistical test, 655
normality, 14
normalization, 15
normalized degree of activation, 73
number of (learning) parameters, 99
number of neurons per layer, 748

O

object recognition, 346
objective evidence, 55
objective function, 697–699, 703, 706, 709–713, 717–720, 722–724, 737
objective increment, 215
oil and gas, 802
oil refinery, 809
oil, 810
online dimensionality reduction, 105–107
online feature weighting, 106
online learning approach, 178
open benchmarks for FDD, 271
operations with fuzzy sets, 21
operations, 805
operator advisory applications, 806
optimization, 186, 189, 192–193, 213, 556, 559
optimizer, 871–872, 874, 884
other AIS models, 567

P

Page-Hinkley test, 105
parameter drift, 231
parameter tuning, 151, 159
parameter vector, 706, 708, 710, 717, 719–720, 722–723
parasitic dynamics, 227–228
Pareto Archived Evolution Strategy (PAES), 642–643
Pareto front, 653–654
participatory learning, 94
partition, 192, 194, 197
patch-clamp, 342
pattern, 185–187, 189
pattern recognition, 731, 733
perceptron, 322
performance, 192
performance criteria, 224
periodic holdout procedure, 120
permutation problem, 752
phase code, 335
phenotype, 514
physical computation, 56
planning and scheduling, 805
plasticity-stability dilemma, 85
Programmable Logic Controller (PLC), 802
partial least squares (PLS), 801, 812
plug-and-play functionality, 116–117
policy iteration, 410
power, powertrain, 865–866, 870–873, 876–889, 896–897
probabilistic population coding (PPC), 339
precise evolving fuzzy modeling, 71
prediction horizon, 213

prediction, 896
preference relation matrix, 83
probabilistic model, 338
probability, 565, 890–891, 894–895
problems with constructive ANN, 757
problems with evolutionary optimization of ANN, 751–752
process changes, 84
process control, 807
process engineering, 805
process industry, 801–803, 809, 819
process integration, 805
projection concept, 75
projection of fuzzy relations, 34
proper class, 47
pseudo-streams, 84
particle swarm optimization (PSO), 587–588, 593–594, 598–599, 671, 688–690
psychophysics, 343

Q

Q-learning, 414
Quantum inspired Evolutionary Algorithms (QiEA), 393

R

R4 network, 754–755
rate code, 336
reactor temperature, 814–815
receding horizon, 214
receptive field, 339
recognition problem, 443
recurrent, 867–869, 871–873
recursive algorithm, 85
recursive fuzzily weighted least squares (RFWLS), 87–88

recursive learning of linear parameters, 86–89
recursive learning of nonlinear parameters, 89–91
recursive least squares, 91, 107, 128, 156, 409, 469
reference model, 215, 228
reference trajectory, 213, 215
regression, 656
reinforcement learning, 341
relative degree, 216, 235
reliability, 108–110
representation theorem, 17
representer theorem, 456
reservoir-based computation, 391
residual set, 233
Resource Allocating Network (RAN), 753–754
Reproducing Kernel Hilbert space (RKHS), 449
robotics, 562, 572, 598, 821
root mean squared error, 122
rule (parameters) initialization, 88
rule consequent, 70
rule evolution criterion, 88, 93, 106
rule evolution, 93–94, 106
rule firing degree, 70
rule importance levels, 113
rule learning, 92, 171, 198–200, 636, 644, 656
rule merging, 95
rule pruning, 92, 95, 111
rule significance, 94
rule, 865–866, 871, 876, 884–885, 891, 893

S

sample selection, 115

sampling hypothesis, 339
Supervisory Control and Data Acquisition (SCADA), 802
second order MA, 616, 626
selection, 161
self-learning computer systems, 69
semantics of information abstraction, 60
semi-supervised learning, 85
separating surface, 447, 545, 479
sequential hybrid algorithm, 614
set, 5, 186–198
standard genetic algorithm (SGA), 612
sequential hybrid algorithm (SHA), 614
shape-space, 551
short-term plasticity, 340
swarm intelligence (SI), 587
sigmoid, 869
similarity concept, 93
Simple Evolving Connectionist System (SECoS), 758, 761
simplicity, 111
small category, 51
smooth forgetting, 103
State, Operator and Result (SOAR), 328
soft on-line dimension reduction, 106
soft rule pruning mechanism, 113
software sensors, 807
SPAN, 395
spatial velocity, 814–815
specificity, 19, 39
speech recognition, 763
spike timing dependent plasticity, 321

split-and-merge concepts for rules, 117
stability analysis, 218, 230
stability of Evolving Fuzzy Systems, 99
Stackelberg game, 435
standard genetic algorithm, 612
state, 865–866, 868, 870–871, 873–876, 884, 890–892, 894–895
static models, 67
steepest descent, 90–91
strictly described memotype, 610
structural optimization, 766
structure evolution (different concepts), 94
structured combination, 60
structured thinking, 60
sub-optimality, 88
subtractive clustering, 175
supervised learning, 322, 387, 502, 548, 872
support of a rule/cluster, 94
Support Vector Machine (SVM), 323, 441, 443, 473
support, 15
symbolic architectures, 327
symbolic representation, 318
synapse, 332
synchronous optimal adaptive control, 433

T

Takagi—Sugeno fuzzy model, 211
Takagi—Sugeno fuzzy systems, 72–75, 79
template-based approach, 151
temporal code, 335
theory of types, 51
thermal cracking, 817

time-variant faults, 244
two-layered hybrid algorithms (TLHA), 614
t-norm, 70
toll-like receptor algorithm, 553
topology selection, 745
tracking error, 226, 229
training algorithm, 753, 764, 832
training parameters, 749, 766
training, 868–869, 871–872
transformation of semantics, 50
transition, transitional, transitioning, 890–896
transparent, 192–193, 196
triangular norms and conorms, 24
true optimal incremental solutions, 72
type-2 fuzzy logic, 671–674, 692–693
type-2 fuzzy systems, 76
type-reducer, 201, 203
types of faults, 243

U

universal approximators, 69
universal grammar, 331
Universal Turing Machine (UTM), 54
unlearning effect, 99
unsupervised FDD, 265
unsupervised learning, 345
useability of Evolving Fuzzy Systems, 114
user uncertainties, 108
utility function, 347

V

value iteration, 410

variability of the version space, 108

V-detector, 550

vehicle, 867–871, 873–874, 876–878, 881, 883–889, 896

very large data bases (VLDB), 68

video processing, 763

visbreaking, 817

visual interpretability, 114

Von Neumann–Bernays–Goedel (NBG) axiomatization, 47

W

water/chlorine balance, 815

weakly described memotype, 610

weight, 868, 873–874, 877, 879–888, 892–895

weighted least-squares criterion, 87

winner-takes-it-all, 80

working memory, 341

wrapper, 196

Z

Zermello–Fraenkel (ZF) axiomatization, 45

zeroth order MA, 616