

Mircea Negoita, Bernd Reusch (Eds.)

Real World Applications of Computational Intelligence

Studies in Fuzziness and Soft Computing, Volume 179

Editor-in-chief

Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

Further volumes of this series
can be found on our homepage:
springeronline.com

- Vol. 165. A.F. Rocha, E. Massad,
A. Pereira Jr.
*The Brain: From Fuzzy Arithmetic to
Quantum Computing*, 2005
ISBN 3-540-21858-0
- Vol. 166. W.E. Hart, N. Krasnogor,
J.E. Smith (Eds.)
Recent Advances in Memetic Algorithms,
2005
ISBN 3-540-22904-3
- Vol. 167. Y. Jin (Ed.)
*Knowledge Incorporation in Evolutionary
Computation*, 2005
ISBN 3-540-22902-7
- Vol. 168. Yap P. Tan, Kim H. Yap,
Lipo Wang (Eds.)
*Intelligent Multimedia Processing with Soft
Computing*, 2005
ISBN 3-540-22902-7
- Vol. 169. C.R. Bector, Suresh Chandra
*Fuzzy Mathematical Programming and
Fuzzy Matrix Games*, 2005
ISBN 3-540-23729-1
- Vol. 170. Martin Pelikan
*Hierarchical Bayesian Optimization
Algorithm*, 2005
ISBN 3-540-23774-7
- Vol. 171. James J. Buckley
Simulating Fuzzy Systems, 2005
ISBN 3-540-24116-7

Vol. 172. Patricia Melin, Oscar Castillo
*Hybrid Intelligent Systems for Pattern
Recognition Using Soft Computing*, 2005
ISBN 3-540-24121-3

Vol. 173. Bogdan Gabrys, Kauko Leiviskä,
Jens Strackeljan (Eds.)
Do Smart Adaptive Systems Exist?, 2005
ISBN 3-540-24077-2

Vol. 174. Mircea Negoita, Daniel Neagu,
Vasile Palade
*Computational Intelligence: Engineering of
Hybrid Systems*, 2005
ISBN 3-540-23219-2

Vol. 175. Anna Maria Gil-Lafuente
Fuzzy Logic in Financial Analysis, 2005
ISBN 3-540-23213-3

Vol. 176. Udo Seiffert, Lakhmi C. Jain,
Patric Schweizer (Eds.)
*Bioinformatics Using Computational
Intelligence Paradigms*, 2005
ISBN 3-540-22901-9

Vol. 177. Lipo Wang (Ed.)
*Support Vector Machines: Theory and
Applications*, 2005
ISBN 3-540-24388-7

Vol. 178. Claude Ghaoui, Mitu Jain,
Vivek Bannore, Lakhmi C. Jain (Eds.)
Knowledge-Based Virtual Education, 2005
ISBN 3-540-25045-X

Vol. 179. Mircea Negoita,
Bernd Reusch (Eds.)
*Real World Applications of Computational
Intelligence*, 2005
ISBN 3-540-25006-9

Mircea Negoita
Bernd Reusch
(Eds.)

Real World Applications of Computational Intelligence



Professor Mircea Gh. Negoita
Wellington Institute of Technology
School Information Technology
Centre for Computational Intelligence
Te Puni Mail Centre Buick Street
Private Bag 39803, Wellington
New Zealand
E-mail: mircea.negoita@weltec.ac.nz

Professor Bernd Reusch
University of Dortmund
FB Informatik
LS Informatik I
Postfach 50 05 00
44221 Dortmund
Germany
E-mail: Bernd.Reusch@udo.edu

Library of Congress Control Number: 2005921892

ISSN print edition: 1434-9922
ISSN electronic edition: 1860-0808
ISBN-10 3-540-25006-9 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-25006-7 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable for prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media
springeronline.com
© Springer-Verlag Berlin Heidelberg 2005
Printed in The Netherlands

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typesetting: by the authors and TechBooks using a Springer L^AT_EX macro package
Cover design: E. Kirchner, Springer Heidelberg

Printed on acid-free paper SPIN: 11364160 89/TechBooks 5 4 3 2 1 0

To our lovely wives,
Doina Negoita and Ute Reusch

Foreword

Computational Intelligence (CI) has emerged as a novel and highly diversified paradigm supporting the design, analysis and deployment of intelligent systems. The intellectual landscape of CI is enormously rich. The discipline of CI brings together technologies of granular computing including fuzzy sets and rough sets, neural networks, and evolutionary optimisation. The strength of CI hinges on the synergy between these technologies. The synergy (or hybridisation, as this term has been in frequent usage as well) helps exploit the advantages of the contributing technologies while reducing their possible limitations. Given the complementary nature of the research objectives of the three pillars of CI (with fuzzy sets focused on building granular interfaces and supporting the formation of the logic blueprints of intelligent systems, neural networks endowing the systems with very much required plasticity and learning abilities and evolutionary computing forming a unified environment of global optimisation), the hybridisation is the crux of the panoply of the ongoing developments. The progress in CI is rapid. The individual technologies evolve quite quickly paving a way to new interesting and truly amazing applications. In the heart of all of those is the principle of hybridisation. It is not surprising at all witnessing a lot of activities within this realm including a number of international conferences, workshops, symposia, and tutorials.

It is my genuine pleasure to introduce a new treatise on the subject of CI and its recent trends co-edited by two well-known researchers, Professors Mircea Gh. Negoita of Wellington Institute of Technology, Wellington, New Zealand, and Bernd Reusch of Dortmund University, Dortmund, Germany. Both of them have a successful track record in this area and are highly qualified for this job. It is needless to say that the book has immensely benefited from their own research, professional insights and practical experience.

The volume is the tangible result of the KES 2004 conference held in New Zealand and more specifically conference tutorials grouped under the banner of the First NZ-German School of Computational Intelligence. The editors are to be congratulated on the careful selection of the material that very well reflects the breadth of the discipline covering a range of highly relevant and

VIII Foreword

practical design principles governing the development of intelligent systems in data mining, robotics, bioinformatics, intelligent tutoring systems and alike. Given this amazingly wide scope of the areas, the reader will be pleased by the depth and clarity of exposure of the material. A newcomer will be pleased by the comprehensive and well-organized material of the volume. A practitioner will gain a down-to-the earth view at the design principles useful in the design, implementation, and validation of intelligent systems. This volume supplements very well the previous book by Negoita et al. entitled "*Computational Intelligence. Engineering of Hybrid Systems*" being already published in the same series in 2004.

The editors and contributing authors deserve our congratulations on their outstanding job. Lucid presentations, coherent organization, breadth and the authoritative coverage of the area – those features make the book highly attractive. An excellent reading for everybody interested in the design and analysis of intelligent systems.

University of Alberta, Edmonton, Alberta, Canada
January 11, 2005

Witold Pedrycz

Preface

With great pleasure, we would like to welcome you to this Edited Book on Computational Intelligence. The motivation of its selected topic – *REAL-WORLD APPLICATIONS OF COMPUTATIONAL INTELLIGENCE* – is deeply justified. The Institute of Electrical and Electronics Engineering (IEEE) has recently created a new Society, The Computational Intelligence Society. The technical frame of this new IEEE Society views Computational Intelligence on a first glance as the area of different computer technologies, mainly the Neural Networks, the FUZZY Systems, the Evolutionary Computation, the DNA Computing, the Artificial Immune Systems, the Evolvable Hardware for and some Symbolic Knowledge Based techniques, that have a large application area both by their variants and hybrids. These intelligent technologies play a major role in real-world applications, so the integration (hybridization) of Computational Intelligence with other technologies such as data bases and data mining are leading to some of the most effective Hybrid *Intelligent Systems (HIS)* actually available on the market of intelligent information and engineering systems.

The scientific cooperation between prof. Bernd Reusch from University of Dortmund, Germany and prof. Mircea Negoita from Wellington Institute of Technology, New Zealand runs over more than ten years and was pointed out by common projects, achievements and events. One of them is the NZ-German School of Computational Intelligence at KES'2004, the Eighth International Conference on Knowledge-Based Intelligent Information and Engineering Systems, together with some other invited tutorial at the same conference.

The tutorial lectures presented to KES'2004 NZ-German School of Computational Intelligence are the component chapter of the first part of this volume, whilst some other KES'2004 selected tutorials are introduced in the second part.

In the first chapter, Negoita is focused on the concomitant increase in dialogue and interconnection between the *Intelligent Technologies* that has led to *Computational Intelligence* and its practical implementation – the **Hybrid Intelligent Systems (HIS)**. Also an engineering design perspective is given of

some *GA* simultaneous implications on both the structure and the performances of the complex *HIS*. Two applications of *VLGGA* (**V**ariable **L**ength **G**enotype **G**enetic **A**lgorithm) – a special non-standard *GA* – are introduced in the form of two hybrid intelligent optimization methods, which have wide applicability. Specialized *GA* applications in the area of crucial high tech *HIS* fields, as **E**volvable **H**ardware (*EHW*), bioinformatics and fault location/diagnosis are the topic of the second and the third part of this chapter. Another promising area of *HIS* applications is introduced, namely the intelligent tutoring systems.

The second chapter by Kecman gives a condensed (but systematic) presentation of a novel learning paradigm embodied in the **S**upport **V**ector **M**achines (*SVMs*). This work focuses on the constructive learning algorithms for both the classification (pattern recognition) and regression (function approximation) problems. Same main *SVMs* strength are emphasized. As the neural networks (or similarly to them), *SVMs* possess the well-known ability of being universal approximators of any multivariate function to any desired degree of accuracy. Consequently, they are of particular interest for modeling the unknown, or partially known, highly nonlinear, complex systems, plants or processes. Also it is clearly stated that there is no need for an application of *SVMs*' model-building techniques. In short, whenever there exists an analytical closed-form model (or it is possible to devise one).

Weerasinghe and Mitrovic deal, in the third chapter, with some investigations regarding the **I**ntelligent **T**utoring **S**ystems (*ITS*), namely whether self-explanation can be used to facilitate deep learning in an open-ended, ill-structured domain. Engagement of students in tutorial dialogues enhanced KERMIT, an already existing *ITS* that teaches conceptual database design, to support self-explanation when their solutions are erroneous. The resulting system, KERMIT-SE, is using self-explanation that leads to improved performance in both conceptual and procedural knowledge.

The difficult task of extracting the models for complex systems from numerical data of behavior is treated by Moraga in the fourth chapter, namely refinements are studied regarding systems representable as sets of fuzzy if-then rules where the premises are not connected by *t*-norms, but by a compensating aggregation. A method is presented to extract this kind of fuzzy rules with support of neural networks. The original method consists of a one to one relation between neural networks using a sigmoide activation function and rules combining premises with a symmetric summation. The method allows not only a convenient way of neuro-fuzzy modeling compensating systems, but also to extract (compensating) rules from existing neural networks. A NN hidden layer optimization, either by means of a pruning strategy or an evolutionary algorithm, leads to a correspondingly reduced number of reasonably understandable rules.

In the fifth chapter, Hildebrand presents industrial applications of *HISs* that are based on international projects funded by the European Community (EC) or the German Research Society (DFG). The results in this projects

prove that the combination of more than one method from computational intelligence together with methods from other disciplines like statistics offer the chance to solve even very difficult problems. Examples are the combination of fuzzy logic and evolutionary algorithms to allow a linguistic processing of color information in a rule-based way. The adaptation of the related fuzzy sets can be achieved by using frequency distributions and evolution strategies. Another example is the combination of digital image processing, digital filters and evolutionary algorithms. **Glow Discharge Optical Emission Spectrometry (GD-OES)** show also the benefits of using computational intelligence methods in combination with methods from statistics. The prediction of the layer thickness by using just the *GD-OES* depth profiles is a challenging improvement of the cost and speed of industrial quality assessment.

Jesse, in the sixth chapter, underlines how far the development of autonomous, intelligent robots has progressed and what problems lie at the centre of research and development in this field. The author proves robots evolution from rigid automatons in the true sense of the word, towards really flexible protagonists capable of carrying out tasks autonomously. The idea is emphasized that even if the service robots currently on the market are only capable of simple tasks like vacuum- or window-cleaning, the coming robot generations will be able to carry out more versatile actions, with characteristics that are in many respects more human. At the same time, and with all due respect to the visions of scientists in robot laboratories a realistic remark is presented with regard to the long way to go before beings equal to humans will be created. The projects that focus on humanoid robots and robot soccer show the way, but only the future will tell where this will lead.

In the seventh chapter, Grzymala-Busse presents introductory aspects of rough set theory. Consistent data are discussed, including blocks of attribute-value pairs, reducts of information tables, indiscernibility relation, decision tables, and global and local coverings. Rule induction algorithms LEM1 and LEM2 are presented. Then the rough set approach to inconsistent data is introduced, with lower and upper approximations and certain and possible rule sets. Some successful examples of rough set theory application to data mining are presented as follows with emphasis on the machine learning/data mining system LERS that has proven its applicability having been used for years by NASA Johnson Space Center (Automation and Robotics Division), as a tool to develop expert systems of the type most likely to be used in medical decision – making on board the International Space Station. LERS was also used to enhance facility compliance as required in a project funded by the U.S. Environmental Protection Agency. LERS was used in other areas as the diagnosis of melanoma, the prediction of behavior under mental retardation, analysis of animal models for prediction of self-injurious behavior, global warming, natural language and data transmission.

In the eighth paper, Howard presents an extremely useful material for *AI* and computer science people trying to get into Bioinformatics. The chapter is an erudite miniencyclopedia covering most areas of Bioinformatics. Essential

cells biology and the complexity of life are discussed. Also is emphasized the relevance of Evolutionary Computation and Genetic Programming in the simulation and understanding of these highly non-linear systems that are the cell community and the organism. Cell Biology and some new Bioinformatics research fields are overviewed as: Genomics, Transcriptomics, Proteomics, Metabolomics, Metabonomics, following a fruitful consultation of an impressive number of relevant works in the literature. The central idea governing this paper is that “a serious attempt at specifying a definition of the problem must come first (the homework), before any attempt to apply the computational intelligence algorithm”.

Paper nine, by Kusiak and Shah, focuses on refined data mining approaches that are using a large bunch of Intelligent Technologies with direct application in development of robust predictive models. Three different data transformation schemes (moving average, Haar and Daubechies wavelets) were successfully applied as data-mining approaches to maximize combustion efficiency by predicting boiler performance. Also the fuzzy logic was applied for better handling the uncertain nature of the parameters and the decision (efficiency). The combustion improvement by minimizing emissions that is of importance to electric power industry was achieved by introduction of meta-controllers based on data mining algorithms.

The book was organized as having a first part comprising the papers of KES'2004 NZ-German School of Computational Intelligence (3 papers from NZ and three papers from Germany) and the second part comprising selected papers from other KES'2004 invited tutorials. The diversity of writing style featuring each of the authors was strictly respected, but this led to the unity and cohesion of the book as a whole. We are indebted to the authors for their efforts, outstanding contributions and assistance in the preparation of this volume. The common feature of all the chapters consists in emphasizing the practical engineering design aspects of complex applications of the Computational Intelligence, including high technology and defense, some of them being the result of international cooperation projects. We would like to express our sincere gratitude to the Royal Society of New Zealand that included the NZ-German School of Computational Intelligence as an official event of its calendar. Finally we would like to express our deepest appreciation to the KES organization, both to its Chief Executive, dr. Robert J. Howlett and to its Honorary Founder prof. Lakhmi C. Jain for their kind agreement of including the NZ-German School of Computational Intelligence in the frame of KES'2004 Conference. This decision was an essential contribution to the final success of the School by providing a large crowd of interested people for attending it.

Wellington
Dortmund
March 2005

*Prof. Mircea Gh. Negoita
Prof. Bernd Reusch*

List of Contributors

M.Gh. Negoita

Director, Centre for Computational Intelligence, Wellington Institute of Technology, Wellington New Zealand
mircea.negoita@weltec.ac.nz
<http://www.weltec.ac.nz/schools/it/cci/index.html>

V. Kecman

Department of Mechanical Engineering
The University of Auckland
Auckland, New Zealand
v.kecman@auckland.ac.nz

A. Weerasinge and A. Mitrovic

Intelligent Computer Tutoring Group Department of Computer Science, University of Canterbury Private Bag 4800, Christchurch New Zealand
t.mitrovic@canterbury.ac.nz

C. Moraga

Dept. Computer Science and Computer Engineering
University of Dortmund
Germany
Claudio.Moraga@udo.edu

L. Hildebrand

University of Dortmund
Department Computer Science I
Otto-Hahn-Strasse 16
44227 Dortmund, Germany
lars.hildebrand@udo.edu

N. Jesse

University of Dortmund
Department Computer Science I
Otto-Hahn-Strasse
44227 Dortmund, Germany
jesse@ls1.cs.uni-dortmund.de

QuinScape GmbH

Thomassstraße 1
44135 Dortmund
<http://www.QuinScape.de>
Norbert.Jesse@QuinScape.de

J.W. Grzymala-Busse

Department of Electrical Engineering and Computer Science
University of Kansas
Lawrence, KS 66045, USA
and
Institute of Computer Science
Polish Academy of Sciences
01-237 Warsaw, Poland
Jerzy@ku.edu

XIV List of Contributors

D. Howard

QinetiQ, Malvern, Worcestershire
United Kingdom
dhoward@qinetiq.com

A. Kusiak

Intelligent Systems Laboratory
Mechanical and Industrial
Engineering
3131 Seamans Center
The University of Iowa
Iowa City, IA 52242–1527, USA
andrew-kusiak@uiowa.edu

S. Shah

Intelligent Systems Laboratory
Mechanical and Industrial
Engineering
3131 Seamans Center
The University of Iowa
Iowa City, IA 52242–1527, USA
andrew-kusiak@uiowa.edu

Contents

Part I NZ–German School of Computational Intelligence at KES2004

| | |
|--|-----|
| Basics of Engineering the Hybrid Intelligent Systems – Not Only Industrial Applications | |
| <i>M. Gh. Negoita</i> | 3 |
| Basics of Machine Learning by Support Vector Machines | |
| <i>V. Kezman</i> | 49 |
| Supporting Deep Learning in an Open-ended Domain | |
| <i>A. Weerasinghe and A. Mitrovic</i> | 105 |
| Data Driven Fuzzy Modelling with Neural Networks | |
| <i>C. Moraga</i> | 153 |
| Hybrid Computational Intelligence Systems for Real World Applications | |
| <i>L. Hildebrand</i> | 165 |
| Autonomous Mobile Robots – From Science Fiction to Reality | |
| <i>N. Jesse</i> | 197 |

Part II Selected KES2004 Conference Tutorials

| | |
|--|-----|
| Rough Set Theory with Applications to Data Mining | |
| <i>J. W. Grzymala-Busse</i> | 223 |
| Bioinformatics with Evolutionary Computation | |
| <i>D. Howard</i> | 245 |

XVI Contents

| | |
|--|-----|
| Maximization of Combustion Efficiency: A Data Mining Approach | |
| <i>A. Kusiak and S. Shah</i> | 283 |

Part I

**NZ–German School
of Computational Intelligence at KES2004**

Basics of Engineering the Hybrid Intelligent Systems – Not Only Industrial Applications

M.Gh. Negoita

Abstract. Computational Intelligence (*CI*) is the methodological framework fitting the highly interdisciplinary requirements featuring most real-world applications, no bridge exists between the different stand alone *Intelligent Technologies*, namely **Fuzzy Systems** (*FS*), **Neural Networks** (*NN*), **Evolutionary Algorithms** (*EA*) **DNA Computing**, **Artificial Immune Systems** (*AIS*) and **Knowledge Based Systems** (*KBS*). The concomitant increase in dialogue and interconnection between the *Intelligent Technologies* has led to *Computational Intelligence* and its practical implementation – the **Hybrid intelligent systems** (*HIS*).

This paper begins with drawing up the framework of *CI* and by introducing the *HIS*. The second section gives an engineering design perspective of some *GA* simultaneous implications on both the structure and the performances of the complex *HIS*. Two applications of **VLGGA** (**V**ariable **L**ength **G**enotype **G**enetic **A**lgorithm) – a special non-standard *GA* – are introduced in the form of two hybrid intelligent optimization methods which have wide applicability. Specialized *GA* applications in the area of crucial high tech *HIS* fields as **Evolvable Hardware** (*EHW*), bioinformatics and fault location/diagnosis are the topic of the second and the third part of this work.

Another recent but promising area of *HIS* applications is focused on intelligent tutoring systems. They model instructional and teaching strategies, empowering educational programs with the ability to decide on what and how to teach students. Usually, learning systems require that the student change to fit the system but with the advent of electronic learning systems, an added flexibility gives the opportunity for *the learning system to change to fit the student's needs*. The key role for accomplishing this task inside the ultimate learning systems is played by *HIS* technologies as described in Chap. 5. Also, **WITNeSS** (**W**ellington **I**nstitute of **T**echnology **N**ovel **E**xpert **S**tudent **S**upport) is an intelligent tutoring system introduced in Chap. 5.

1 Intelligent Techniques in the Framework of Computational Intelligence (CI)

Intelligent Technologies (*IT*) or **Computational Intelligence** (*CI*) are increasingly improving any area of our social-economic life. Nowadays we are

confronted with a lot of complex real-world applications such as different forms of pattern recognition (image, speech or handwriting), robotics, forecast and different kind of decision-making in uncertainty conditions, etc. These kind of applications is relied on a new concept and at the same time a new framework that is named **Soft Computing (SC)** by L.A. Zadeh, the father of fuzzy systems. The basic ideas underlying *SC* belong to prof. Zadeh. His vision on different techniques in interaction is strongly influencing the development of the new generation of intelligent (perception-based) systems toward a **Computational Intelligence (CI)** [75, 76].

1.1 Key Features of the Hybrid Intelligent Systems

A large bunch of methods have been developed as useful tools of information processing, machine learning and knowledge representation. These methods are either an outcome of “**Artificial Intelligence**” (*AI*) framework, mainly relied on symbolic **Knowledge Based Expert Systems (KBES)**, or, an outcome of “**Intelligent Technologies**” (*IT*) framework, mainly relied on **Fuzzy Systems (FS)**, **Neural Networks (NN)**, **Evolutionary Computation (EC)** and even on some other unconventional methods as **Artificial Immune Systems (AIS)** or *DNA* computing for example. *DNA* computing framework was enlarged by *DNA* hybridisation: *FS* were implemented in *DNA* bio molecules, *DNA NNs* became available and *DNA* computing began to be applied in **Genetic Programming (GP)** implementation. *AIS* are actually widely used in main in conjunction with *NN* and *EC*, but also with other *CI* paradigms, such as **Case-Based Reasoning (CBR)**, **Classifier Systems (CS)** and *FS*. The integrative benefits of the current most influential biological motivated *CI* techniques were already proved by a wide range of *AIS* applications.

IT and *AI* are distinct frameworks; they must be distinguished with the aim of an improved exploitation in applications (see Table 1). Each *IT* as well as each *AI* method has its particular *strengths* and also its *weaknesses* that make it suitable for particular applications and not for other ones. These specific limitations both to *AI* and to *IT* led to creation of **Hybrid intelligent systems (HIS)**, where two or *more techniques are combined* to “co-work” for overcoming the limitations of individual techniques. An overview of the main *IT* and *AI* techniques as working solely, will be useful with regard to what they are dealing with, or which own advantages and limitations are featuring the individual effectiveness of the method.

FS are dealing with handling the imprecision in data, finally performing an intelligent decision made by *fuzzy concepts* and *human like approximate reasoning*. *FS* have knowledge bases that are easy to examine and understand, as well as easy to update and maintain, because they are based on fuzzy IF-THEN rules. *FS* are flexible when dealing with incomplete and inconsistent data by aggregating the hypothesis of all the rules. But the heavy *reliance on human experts* in building the fuzzy IF THEN rules, the *manually specification*

Table 1. A Comparison between *AI* and *CI*

| | AI | IT |
|---|--|--|
| <i>Main purpose</i> | Conceiving and design of some <i>intelligent non-living systems</i> | Increasing the machine intelligence quotient of non-living systems (making the machines able of <i>thinking</i> and acting as close as possible to the behaviour of human beings) |
| <i>Methods</i> | Classical ones, most based on <i>first-order bivalent predicate logic</i> | Solutions of the problems are satisfactory in form of <i>global approximations</i> and in most of cases <i>not by all means optimal</i> |
| <i>Information processing</i> | By handling <i>symbols</i> | By <i>numerical computation</i> |
| <i>Some typical successful applications</i> | <ul style="list-style-type: none"> • Knowledge-based engineering • Fault diagnosis, etc... | <ul style="list-style-type: none"> • Different forms of pattern recognition (image, speech, handwriting) • Robotics • Forecast applications • Decision-making under uncertain conditions, etc... |

of membership functions and fuzzy rules, as well as strong restrictions in *automatic adaptive learning* capabilities of changing in the external environment, are real *FS* limitations.

NN are dealing with inherent *parallelism* in data flow. Their capacity of learning, namely learning patterns in data that are noisy, incomplete and even that contain contradictory examples, confer them two distinct advantages in information processing: not only the ability of *processing incomplete information*, but also to *generalize the conclusions*. *NN* effectiveness consists of an easy way of modeling and forecasting non-linear systems as well as a well suited method for tasks when dealing with incomplete data, unavailable experts or where no rules can be formulated. *NN* still are *unable to explain the reason of their conclusions* as well as unable to interact with conventional data bases. The *lack of structured knowledge representation* is another *NN* drawback. *NN* difficulties in training and in generalization capabilities for large and complex problems are real *scalability problems*.

EC (Genetic Algorithms – *GA*, especially) are also dealing with *parallelism* in data flow, they are efficiently searching and optimization tools for very large data sets, largely unconstrained by local minima problems. They

don't require any mathematical modeling. *EC* are optimization methods of either symbolic or *NN* or *FS* systems. Strictly referring to *GA*, their effectiveness consists of: highly suitability for parallel computer implementation; particular success in large search and optimization problems; ability of learning complex relationships in incomplete data sets; availability of being used as "data mining" tools for discovering previously unknown pattern; ability of providing explanations of the decisions that they produce, in a format that human understand; they can adapt to changes in their operating environment. *GA* limitations are as follows: they are *computational expensive* methods involving an *application dependent setting* of *GA* parameters (for crossover, mutation) that could be a time-consuming trial and error process.

DNA computing is certainly a promising implementation method of information-processing capabilities of organic molecules, used with the aim of replacing digital switching primitives in the computers [60]. This kind of computing is strongly relied on the availability of proper working environment (technological support devices): a jump from microchip electronic devices to *DNA* molecules is finally expected, but actually the technology is still struggling at the level of combining electronic computing with *DNA* computing, where most of *DNA* operation in the test tubes are carried out without the intervention of the user. *DNA* computing is featured by *higher parallel data structures* than *NN* and *GA*, namely the case is now of a massive parallelism conferred by *DNA* strands. The second strength of *DNA* computing derives from the *complementarity* between two *DNA* strands when bonding takes place. *DNA* effectiveness is mainly as a powerful tool for computing, relied on handling suitable encoded information of a very high density that leads to far-reaching conclusions when bonding takes place. The main limitation of *DNA* computing consists of a *higher than admissible error rate of operations* because the actual lack of a proper technological environment.

AIS extend the concept of organization of multicellular organisms for the information systems in context of an external environment that is dynamic and involves diverse and unpredictable events, where faults and events are not only complicated faults, such as intermittent faults, but they are malicious faults potentially exploiting the weak points or holes in a system. *AIS* are the emergent intelligent technique that develops a crucial strategy of systems survival – the defence of the information systems against malicious faults.

The computational and practical issues of real world-applications illustrating the strengths and weaknesses of *CI* and *AI* methods led to evolution and development of *HIS*. The evolution of *HIS* is a consequence of modelling the human information processing, but the application engineering was in main the way of hybridising the *CI* and *AI* methods, and this was made under an interdisciplinary vision of solving a large range of real-world applications. A key factor of success for an effective use of *CI* and *AI* methods is a real understanding of the strengths and applicability of *HIS*, in a very large area of industry and finance/business environment.

There are some *key features* of most *HIS*, which make them particularly useful for problem solving, but certainly no *CI/AI* technique exhibits *all these features*. The *key (application required) features* making *HIS* development suitable for one or other application as the case, are mentioned as to be the following: *learning, adaptation, flexibility, explanation* and *discovery* [12]. Accurate quantitative methods would be ideal for assessing the potential availability of the main *CI/AI* techniques in providing the five main desired *HIS* properties. No such tools are available, but a qualitative balance using the remarks accumulated and proved as a result of application development might perform the analysis.

An *HIS* must be able of performing just a simple mining through a huge amount of previously acquired records of input data, so as a model of application practices to be get finally. This *HIS* feature is called *learning* and means the capability of learning the tasks/decision that *HIS* have to perform, directly from collected data.

A typical intelligence to *HIS* must be expressed also by the availability of decision-making procedures, which run so as to be understood by humans. Also decision-making procedures have to be featured by transparency, this allowing a reasoning process that is understood so as the *HIS* itself to be possibly improved. This *HIS* feature is known under the name of *explanation*.

But learning is a process that must continuously feature the activity of any *HIS* in any conditions (learning just an initial amount of knowledge is not at all enough for making an *HIS* able of performing a particularly task). *HIS* must be able of monitoring the system tasks constantly (including knowledge revision according to any change in operating environment). This *HIS* feature is called *adaptation*.

Decision making capability of an *HIS* must feature the system even when the case is of imprecise and incomplete input data, even in conditions that *HIS* not encountered before. This basic *HIS* feature is called *flexibility*.

Discovery is *HIS* data mining ability – the capability of mining through a huge amount of collected input data and not only finding relationships that were previously unknown, but checking (both on data integrity and sample size) so that a validation is performed whether the discovered relationship is not just a statistical fluke.

No accurate quantitative comparison methods are available for assessing the potential availability of main *CI/AI* techniques in providing the five main desired *HIS* properties. But the comparison might be performed using the remarks accumulated and proved as a result of application development (see Table 2).

Some comments must be made on the compatibility of main *CI/AI* methods to an ideal implementation, namely for conferring to *HIS* the typical features to real-time systems.

Table 2. Potential availability of main *CI/AI* techniques in providing the five main desired *HIS* properties

| CI/AI Technique | Learning | Explanation | Adaptation | Discovery | Flexibility |
|-----------------|------------|-------------|------------|------------|-------------|
| <i>KBS</i> | inadequate | excellent | inadequate | inadequate | inadequate |
| <i>FS</i> | inadequate | moderate | inadequate | inadequate | excellent |
| <i>NN</i> | excellent | inadequate | excellent | adequate | excellent |
| <i>GA</i> | excellent | moderate | good | excellent | good |

1.2 Classification of the Hybrid Intelligent Systems

Section 1.1 has mentioned the *key (application required) features* making *HIS* development suitable for one or other application, as the case. The summarized quantitative comparison, as in Table 2, is to be used by practitioners as the starting point in building *HIS* that are effective in applications. Table 2 suggests two practical strategies of building *HIS* and much more, *HIS* are classified according to these design strategies [12]. Such a classification might divide HIS in two classes: *function-replacing HIS* and *intercommunicating HIS*.

Function-replacing HIS are systems that were built by combining a *CI/AI* method that is inadequate with respect to one key *HIS* feature with a method that is good to excellent with respect to other key feature. A lot of function-replacing *HIS* is already traditional. See for example: *GA-NN* relied *HIS* that are used for *NN* optimisation (*NN* weights and/or structure adjustments); *GA-FS* relied *HIS* that are used for optimising both the rules structure (number of fuzzy labels) and the FS parameters (fuzzy shape, position of fuzzy variables).

Intercommunicating HIS are systems relied on a *technique-to-task allocation*: a complex application problem is sub-divided in specialised component problems (tasks), and suitable *CI/AI* techniques are allocated to each of these tasks. So different component problems, each of which may require different type of information processing, are solved by different specific *CI/AI* techniques and the final (global) results of the *HIS* are performed by communicating the specific results to each *CI/AI* techniques among themselves. See for example all the aspects with respect to the problem of manufacture launching in case of a new product: a *forecasting task* is solved by using a *NN* technique, a *multi-objective optimisation* task is solved by using *GA* techniques, some reasoning tasks are by using *FS/KBES* techniques as the case.

The first kind of *HIS* classification is relied on real-world application requirements and groups the hybrid systems in two classes as above mentioned. An engineering point of view rather than a “systemic” one might be kept even if we are looking to theoretical (structural) aspects of information processing in *HIS*. But four classes are grouping the *HIS* in this case: *fusion* systems, *transformation* systems, *combination* systems and *associative* systems [23].

Fusion HIS are featured by a new *CI/AI* technique, X , whose procedure of information processing and/or its representation features are melted into the representation structure of a host *CI/AI* technique, Y . The practical aspects of melting a new *CI/AI* technique in a fusion *HIS* means that the host technique Y is diminishing its own weakness and exploits its strength more effective for solving the real-world application under these new circumstances. Let us mention some applications of *NN* based hybrid systems in which *FS* information processing features are fused: fuzzification of input data, optimisation of fuzzy systems, modelling fuzzy inference in *NN* structures, as well as implementation of *FS* logic operations in *NN*. A well known *GA* based hybrid system in which *FS* information processing features are fused is Lee/Takagi *GA* based system of intelligent integrated *FS* design system (both the membership functions and the number of useful fuzzy rules, as well as rule-consequent parameters are designed automatically and optimised involving *GA*), see [27].

Transformation HIS are featured by the transformation of one form of information representation into another one, so these hybrid systems are used for applications where the required knowledge for task achievement is not available and where one *CI/AI* method depends upon a different *CI/AI* method for its reasoning and processing purposes. Let us mention some example of transformation *HIS*. Some typical applications solved by neuro-fuzzy transformation systems are as follows: learning fuzzy *IF-THEN* rules, learning fuzzy clusters and learning membership functions. Here *NN* learning feature is integrated as a method operating on input-output data pairs or even on simply input data. Another type of transformation *HIS* is the *NN-KBES* ones that are performing concept hierarchies or *KBES* rules from a *NN*.

A special case of transformation *HIS* occurs in special application circumstances where transformation is required to be in form of a complex optimisation: first – an un-optimised technique X is melted in to representation Y ; second – an extraction of the optimised prior representation X is performed from optimised representation Y . This is an intermediated class of *HIS* that is usually called as *fusion and transformation HIS*. Such systems are *GA-NN* or *GA-FS/NN* systems performing a complex *GA* relied optimisation of *NN* or *fuzzy NN*, namely envisaging the *NN* connection weights, *NN* structure, *NN* topology and *NN* input data, see [2, 53]). During the last decade, all *CI/AI* methods have got an increased audience from behalf of practitioners, so these techniques were largely involved in a large variety of real-world applications. The approach of using these technologies by themselves alone has exposed real limitations. These limitations are nothing else than the result of the expression aspects of complexities associated with real problems in the real-world work environment. So another class of hybrid systems grows up, *intelligent combination HIS* namely. Combination *HIS* involves two or more *CI/AI* technologies. The two or more *CI/AI* technologies are globally featured by developing a more effective problem solving strategy, relied on the distinct role played by each of these methods in the process of problem solving.

Combination *HIS* approach has an explicit hybridisation essence that is relied on a refined matching of the *CI/AI* methods to the particular components of a modular structured model. Most common combination hybrid systems are *FS-NN-KBES* complex (fault) diagnosis systems in industry and medicine; *FS-NN-GA* hybrid systems for developing intelligent *GA* learning based adaptive control applications of *FS-NN* controllers; combination systems performing a large variety of *KDD* Applications in finance/banking area (*KDD* stands for **K**nowledge **D**iscovery in **D**atabases); general purpose *GA-KBES* combination scheduling systems; general purpose *FS – GA-KBES* customer advisor systems advising people how and what to buy; combination systems performing a large variety of *KDD* specialised forecast/prediction tasks ; combination systems for complex tasks in robotics, etc.

The four *HIS* classes, namely *fusion HIS*, *transformation HIS*, *fusion-transformation HIS* and *combination HIS* are deeply application motivated. All of these *HIS* categories led to spectacular results both from the point of view of the quality in task achievement and from the point of view of the range of covered task in case of a suitable application matching.

But no *HIS* of these categories was able of being general purpose available any was the application. Some limitation are quite featuring these four classes of *HIS*, as follows: sometimes they are not able of capturing all aspects of human cognition related to application solving (typical drawback to fusion and transformation systems); even the combination *HIS* which seems to be the most complete ones express a lack of minimal knowledge transfer among modules despite of their system flexibility; the range of covered tasks by fusion systems is restricted by loosing on the declarative aspects of application solving due to conversion of explicit knowledge into implicit knowledge.

Drawbacks of fusion, transformation, fusion-transformation and combination systems favoured implementation of *associative HIS* a class of more powerful systems in applications. Associative *HIS* may incorporate fusion, transformation and combination architectural/concept elements as well as integrate standalone *CI/AI* techniques. The results of these strategies confer much-improved *HIS* quality features with respect to quality of task achievement and to tasks range. See [23] for more details on associative *HIS* as well as for a wonderful description of a lot of *HIS* applications.

As a conclusion, hybridisation was application required and strong related to real-world applications. Hybridisation might be considered as an hierarchical process: the first hybridisation level consists of just combining some *CI/AI* techniques among themselves in form of *HIS* and this is matching applications of a low to moderate complexity; the second hybridisation level is typical to applications of high complexity where some *HIS* are combined on their turn among themselves, together or not, with some stand alone *CI/AI* techniques. These two level of hybridisation were nothing else than the result in evolution of design strategies but just inside the “silicon environment” if we are thinking on actual *HIS* technological support. Despite of still a pioneering stage, quite promising steps were made in a possible transition from

the silicon environment (microchips hosted) to the so-called “carbon environment” (hosted by *DNA* molecules). *DNA* molecules have intrinsic huge resources of information-processing abilities featuring the organic molecules. So the way is paved for a third hybridisation level performing hybridisation of some stand alone *CI/AI* techniques or even of some *HIS*. The third level of hybridisation means a bio-molecular implementation of soft computing, so the uncertain and inexact nature of chemical reactions inspiring *DNA* computation will lead to implementation of a new generation of *HIS*, the so-called **Robust (Soft Computing) Hybrid intelligent systems – RHIS**.

RHIS are systems of biological intelligence radically different from any kind of previous intelligent systems by three main features:

- *robustness* – conferred by the “carbon” technological environment hosting them
- *miniaturization* of their technological components at a molecular level
- the highest (*biological*) *intelligence* level possible to be implemented in non living systems
- dealing with world knowledge in a manner of *high similarity to human beings*, mainly as the result of embedding *FL*-based methods of **Computing with Words** and **Perceptions (CWP)** featured by the understanding that perceptions are described in a natural language.

In last years, interest has been growing in the use of a large *HIS* variety. Any is the hybridisation level, the use of biologically inspired *CI* techniques was decisive for *HIS* features and performances. An emerging and promising biologically inspired technique is the field of **Artificial Immune Systems (AIS)**. These are adaptive systems inspired by theoretical immunology – from the vertebrate immune system, especially – and by observed immune functions and principles, which are applied to typical problems to be solved by *HIS* such as: pattern recognition, data analysis (clustering), function approximation and multi-modal optimisation [9]. Results got in some applications regarding function optimisation show that AIS algorithms are more effective than GA. AIS perform significantly fewer evaluations than a GA and this is not at all altering the quality of the final solution [22].

2 Aspects of a Design Perspective Regarding Real-World Application of the Hybrid Intelligent Systems

This chapter is focussed on design aspects of *HIS* engineering. The work is limited to just introduce some special design elements in connection with original applications. On the other hand, this chapter will also be focussed on some combination approaches which involve the main *CI/AI* techniques, but in the same time are both fundamental and with a very large applicability.

A set of main remarks is to be made regarding the combination approaches with large applicability, namely with respect to the optimisation techniques in *HIS*: *GA* can be seen as a better alternative optimisation method to *NN*; *GA* based fusion and transformation systems are increasingly used in most applications. *SoGA-FS* relied *HIS* are used for automating and optimisation tasks in *FS* design. *GA-FS* relied or even *GA-NN-FS* relied *HIS* are used for simultaneously optimising the connection weights as well as the *NN* structure and topology. Two applications of *VLGGA* (**V**ariable **L**ength **G**enotype **G**enetic **A**lgorithm) – a special non-standard *GA* – are introduced in form of two *hybrid intelligent optimisation methods* of large applicability:

- a *VLGGA* based method of learning (from examples) the parameters featuring the fuzzy inference systems [10];
- a *VLGGA* relied optimisation method of parameters featuring the fuzzy recurrent neural networks [2]

2.1 GA with Variable Length Chromosomes as an Optimisation Tool in Complex Hybrid Intelligent Systems

Some features of traditional *GA* are leading to a recombination operator having a simple implementation: they use genotypes of predetermined length, that is homologous features are coded for on homologous positions within the genotype.

Complex organisms in nature present a kind of *gene's specialization: structural genes (qualitative)* and *adjusting genes (quantitative)*. The adjusting genes affect the structural genes globally [78]. Another interesting evolution aspect is that a correspondence for a recombination operator that involves individuals with significant different genotypes is not possible to be found at any level of beings in natural environment.

These two last aspects of evolution in nature made possible implementation of optimization methods in form of *VLGGA*, where:

- absolute position of the symbols usually does not provide information about the related feature;
- the recombination operator should be implemented in a different manner and, most implementations are application-dependent.

VLGGA are successfully used in getting better performances for systems with complex structure or, at the same performances, a less complex structure of the system [10]. The flow-chart procedure of our *VLGGA* is as from [10], see Fig. 1:

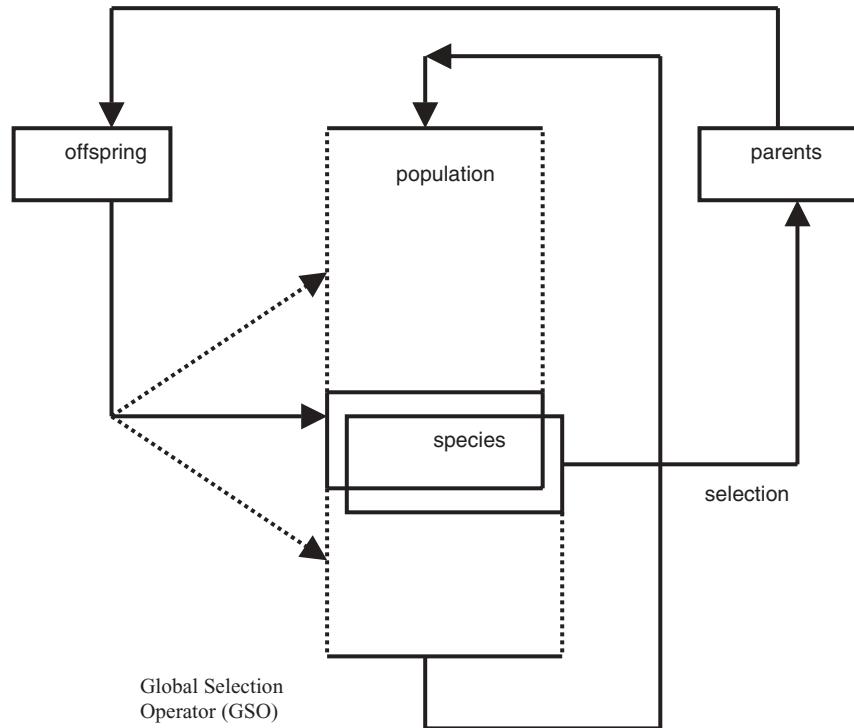


Fig. 1. The flow chart of a *GA with variable length chromosomes*

STEP1 – *Generate initial population* individual by individual (two fields for each of the individuals: *structural genes* generation, followed by *adjustable genes* generation);
 STEP2 – *Group* the individuals *in species*;
 STEP3 – *Evaluate* the initial population
 STEP4 – **repeat**
 for each *species*
 advance (*selection, crossover, mutation*)
 end
 STEP5 – *global selection on the whole population*
 STEP6 – **until** *STOP conditions*

2.2 VLGGA Based Learning the Parameters of a Fuzzy Inference System from Examples

Fuzzy Inference Systems (*FIS*) provide models for approximating continuous, real valued functions. The successful application of fuzzy reasoning models depends on a number of parameters, which are usually decided by a subjective

methodology (traditionally, fuzzy rule bases was constructed by knowledge acquisition from human experts).

Both the fuzzy partitions of the input and output spaces and the rule base were formerly decided in a subjective manner, by knowledge acquisition from human experts. However it is important that a change in a structure of a rule may locally alter the performances of a *FIS*. Moreover, a change in the partition of the input or output spaces may alter the performances in a more global and significant way.

A systematic procedure for the design and optimisation of a *FIS* is required. The interest for this topic was reflected in the number of papers concerning methods for learning/optimising the parameters of a *FIS*. However, most of them treat the different optimisation tasks in a separate manner. It has been shown [57], that the performances of a fuzzy controller may be improved if the fuzzy reasoning model is supplemented by a genetic-based learning mechanism. The results have indicated that the best performance of the fuzzy control system was got when the spaces of both fuzzy membership functions and the fuzzy relation matrix was searched by *GA*.

It still remains a source of subjectivity in deciding the number of fuzzy labels for both input and output universes of discourse. It is obvious that a smaller number of fuzzy labels determines a smaller number of fuzzy rules and, consequently it is possible to obtain a less complex fuzzy model and a smaller computational time for the inference.

The aim of this section is to describe the main aspects involved in developing a genetic-based flexible method able of learning the parameters of a *FIS* using a set of given examples. These parameters are: a minimal number of fuzzy labels, the shape and position of the corresponding membership functions, and the structure of a minimal number of fuzzy rules such as the inference error related to the set of examples to be below a given threshold.

It is proved in [10], that a non-linear continuous function $f: [0, 1] \rightarrow [0, 1]$ can be better approximated by a number of line segments with variable sizes (Triangular Partition – *TP* case) than by the same number of segments but with uniform projections on the horizontal axis (Equidistant Triangular Partition – *TPE* case). A much better approximation may be got in case of hyperbolic segments (Triangular Partition with Variable Length – *TPh* case). It means that it is also possible to get the same approximation error but with a smaller number of segments i.e. a smaller number of fuzzy labels. Furthermore, changes in the shape of the transfer function can be got significantly by modifying the inference method [73] or the defuzzification method [24]. A small error in a certain application can be got by a *TPE*-based system if complex enough. But the *HIS* described in this section uses a learning algorithm intending to achieve a certain level of performances with a system as simple as possible.

The task of this *HIS* is to simultaneously learn the parameters of a *FIS* (see for example, the number of fuzzy labels for inputs and output, the shape and position of the membership functions and the structure of the rule base. The

chromosome code structures of different complexities so the length of these chromosomes is subject to change during the algorithm, see also [15]. Therefore the genetic operators and the strategy of evolution are adapted to these circumstances. The main features of the genetic-based learning method are as follows:

- the *chromosome* is structured by three fields: *FL*-field that codes the number of fuzzy label for inputs/outputs; *MF*-field that codes the shape and position for the fuzzy membership functions; *FR*-field that codes the structure of the rule base
- the *initial population* randomly generated as follows: first, the *FL*-field is randomly generated for each chromosome; then, according to the number of fuzzy labels, the next two segments (fields) are generated; the phenotype corresponding to the *FL*-field is used to compute the length of each chromosome; the *FL*-field gives the complexity of each structure and the *MF* and *FR* fields give the structure itself
- the *fitness function* is computed in order to combine two different optimization problems, i.e. a minimal error related to a set of given examples and a minimal structure (meaning a minimal number of fuzzy labels for input and output and a minimal number of rules in the rule base);

The fitness function comprises two main terms:

$$\text{Fitness (chromosome)} = \alpha E1 + \beta E2$$

where

$$E1 = E1 \text{ (error, threshold)}$$

$$E2 = E3 \text{ (number of labels, number of rules)}$$

α, β – parameters that give variable weights to each optimization task according to the application goals

The most important weight has to be for *E1* but, when the threshold is reached, the complex structures are penalized in order to minimize the number of label and rules.

The *GA* runs as described in Sect. 2.1.

Each species contains structures of same complexity and advance independently. New species can be created when a mutation occurs in the *FL*-field, so the population size varies during the algorithm. When *GSO* – the global selection operator is applied, only the fittest chromosomes survive. Some species, then, may grow and other disappears. The evolution is similar to that one in natural environment where different competitive species share the same (limited) survival resources.

The *VLGGA* was tested on a two inputs/one output *FIS* that had to approximate a non-linear, real-valued function:

$$f : [0, 1] \rightarrow [0, 1]$$

$$f(x, y) = 1/(1 + 3^*x - 1.5)^6 + (10^*y - 5)^6$$

A set of examples were generated by using $f(x, y)$ and these examples (values) were used for trying to find *FIS*'s parameters so as to get a good approximation of $f(x, y)$. The *VLGGA* was set up in two ways:

- first, the best *FIS* approximation was searched for a *FIS* featured by *TPE* for the input space, equidistant crisp partition for the output space – 6 labels for each of them – and both *FL* and *ML* fields unchanged
- then, the modification of *FL* and *MF* fields was enabled by using the best error as the threshold for the fitness function.

The evolution of the fittest chromosome in the population and its corresponding total number of labels versus the number of *GA* iterations, led to the conclusion that in case of a *TP*-based system – variable length genotypes – the number of fuzzy labels first increases (complex structures get better approximations). But the number of fuzzy labels decreases below the total number of labels in *TPE* case if the fitness function of the *TP*-based system comes near the given (penalty) threshold, and this happens due to the penalty function for complex structure that become significant. The mean square fitness function errors related to the same set of examples remain comparable both in case of *TPE*-based system and in case of *TP*-based system. A concluding remark may be mentioned that it is possible to reduce the complexity of a *FIS* – the number of fuzzy labels and the number of fuzzy rules – without altering the performance, if simultaneously both the fuzzy input/output partitions and the structure of the rule base are optimized. The proposed *HIS* also tries to avoid the difficult problem of recombination between parents with variable complexities.

2.3 VLGGA for Optimization of Fuzzy Recurrent NN

Various fuzzification approaches of neural networks (*NN*) have been proposed by using fuzzy numbers [68]. A special class of intelligent hybrid (neuro-fuzzy) systems, based on *NNs* and fuzzy algebraic systems was considered, for different *NN* topologies – multilayer perceptron, Radial Basis Function [20] and fuzzy recurrent *NN* [2]. The application area of fuzzy *NN* is very large, from prediction of time series to modeling of complex chaotic systems.

The general learning methods for neuro-fuzzy *HIS* are based on gradient techniques, developed for working with fuzzy numbers. The well-known typical problems to the gradient learning techniques (many local minima, learning factor determined experimentally, e.g.) are naturally present in these structures too, but their effect is more amplified due to constraint problems with regard to definition of the membership function of fuzzy numbers.

A recently proposed *NN* structure is a Recurrent Artificial Neural Network with Fuzzy Numbers (*RAFNN*), see [2]. The recurrent topology is a fully

connected one that uses symmetric triangular fuzzy numbers in the fuzzy algebraic framework. A computational expansive gradient-based algorithm was proposed and used in order to compute the matrix of fuzzy weights, but the structure (number of neurons) is supposed to be known, or discovered by simple incremental trials starting from minimal number of neurons set to two. *This can be very costly (excessive time consumption) in case of modeling complex systems that could require large number of neurons.*

The gradient-based learning algorithm as mentioned above, was used to prove that a *RAFNN* could learn to solve an *XOR* type problem in the frame of the fuzzy numbers, continuously with two simultaneously changing inputs. This is not at all the case of a simple static *XOR* (with 4 possible input values and giving a typical *XOR* output to a *TTL* circuit). But the case is of a *RAFNN* learning the dynamics of a string of 0, 1 that is randomly, continuously generated producing an *XOR* output that is delayed by 2–3 steps (in each update cycle, the teacher is delayed by $q = 2$ cycles relative to the input that is used for *XOR*, [72]). The structure of *RAFNN* (see Fig. 2), has n units and m inputs – an architecture that is similar to the crisp one in [72].

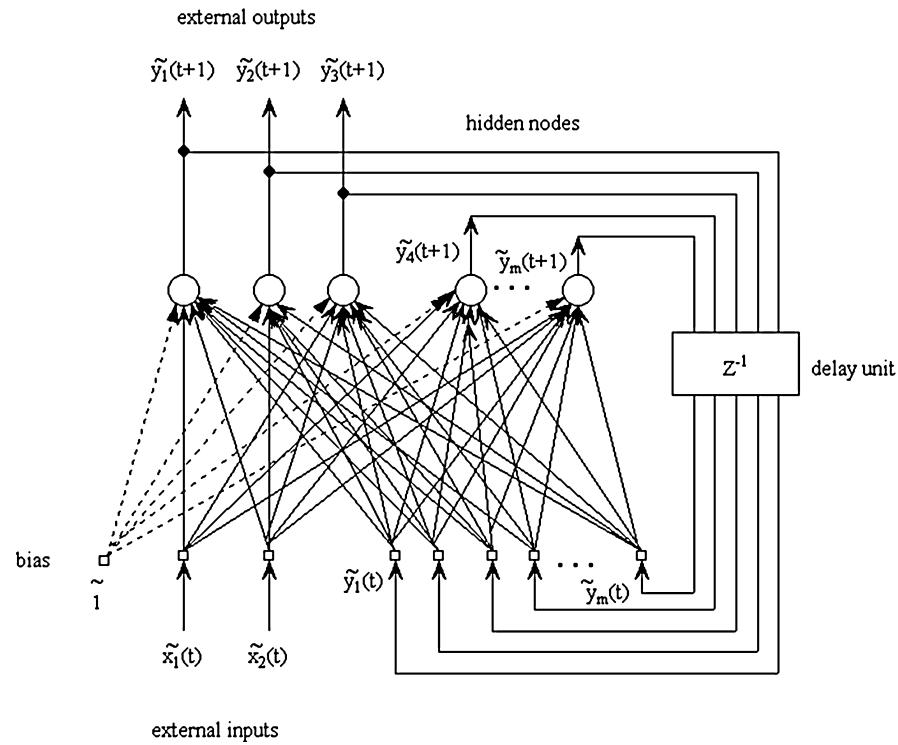


Fig. 2. The *RAFNN* Structure

Each bias allocation in Fig. 2 will be seen as an input line whose value is always $(1, 1, 1)$. The set of indices for input units, output units and target units are denoted by (I, U, T) . The inputs are denoted by $\tilde{x}(t)$, the outputs are denoted by $\tilde{y}(t)$ and $\tilde{d}(t)$ are the targets (if exist) for each of *RAFNN* unit at time t . A generalized $\tilde{z}(t)$ was denoted similar to [72]:

$$\tilde{z}_k = \begin{cases} \tilde{x}_k(t) & \text{if } k \in I \\ \tilde{y}_k(t) & \text{if } k \in U - T(t) \end{cases}$$

The *HIS* introduced in this section made use of a *VLGGA* implementing a *GA* learning algorithm both for the fuzzy weights of *RAFNN* and for *RAFNN* recurrent structure (number of neurons). The genetic-based learning method used for simultaneously learning the *RAFNN* structure and parameters has the steps as described below.

STEP 1. The initial population is randomly generated in two steps for each of the individuals, as follows:

- first, the *structural genes* (representing n – the number of neurons) are generated
- these are random generated numbers, in a mapping format of 15 binary bits (0 or 1) corresponding to a number of neurons in a range $n \in [2, 15]$
- second, the *adjusting gene* is generated in form of a matrix with $n \times (m + n)$ elements
- m is the number of *NN* inputs, incremented by 1 – the bias input ($m = 2$ in case of an *XOR* problem)
- the $n \times (m + n)$ matrix is a linear representation of fuzzy symmetric triangular numbers (see Fig. 3)

| STRUCTURAL GENE | ADJUSTING GENE |
|--|--|
| n genes (number of neurons in <i>NN</i>) | <i>the fuzzy weight matrix</i> $n(m+n)$ genes |

Fig. 3. The chromosome code structure for a *VLGGA* used in *RAFNN* optimization

STEP 2. The individuals are grouped in *species* taking into account the genotype- n – of the structural field in each chromosome, following an ascending order from the lowest to largest number n . A *species* means all the chromosomes with the *same length* (really, in a larger meaning: the *same length* and the *same structure*). This means more than the same length, because different individuals in *VLGGA* may usually have the same length despite of having different structures. But *no special precautions are to be made with regard to this aspect in this application because*

of the suitable coding structure. This is another advantage of particularly using a *VLGGA* in this application.

STEP 3. Each individual of each species is evaluated during this step. The fitness function is defined in order to combine two different optimization problems, i.e. a minimal *NN* error related to a set of desired outputs and a minimal *NN* structure. The formula of fitness function g (*chromosome*) is as follows:

$$\begin{aligned} g(\text{chromosome}) &= \lambda \cdot E_1 + \delta \cdot E_2 \\ E_1 &= |1.0 - J_{\text{total}}| \\ E_2 &= 1/(1+n) \end{aligned}$$

The experimentally determined parameters $-\lambda$ and δ – show criteria weights in the composed fitness function. E_1 is the most important parameter and is given by total error during the k -steps of tests (k is usually 1000). A penalty function [7] is included in the fitness formula above.

The complex structure is penalized after p steps if the error J_{total} has not a decreasing with a threshold δ (see [46]):

$$\sigma = \sigma(k, \rho(J_{\text{total}}), \text{generation})$$

STEP 4. Each species contains structures of the same complexity (see *STEP 2*) and advance independently under *common GA operators (selection, crossover, mutation)* as *GA* are running in parallel. For species containing just one individual, only mutation is applied. The resulted offspring is appended to the species if it is better than the worst fitted chromosome in the whole species. In this manner each species grows as far as it produces well-fitted offspring.

New species can be created when a mutation occurs in the structural field, the chromosome lengths are modified, the individual goes to another species or another species is created. The number of species is subject to change during the algorithm.

STEP 5. After a predetermined number of iterations, *global selection operator – GSO* – is applied, doesn't matter for it the species. The number of fittest chromosomes is fixed no matter what species they belong to.

GSO frequency application is by choice (application-dependent), not necessarily at each iteration: species with short chromosomes are favored by a high frequency of *GSO*, species with long chromosomes are favored by a low frequency of *GSO*. A *Boltzmann selection* was applied in this case with a probability as follows:

$$p = e^{-\frac{E_i(t)}{kT}}$$

where T depends on iteration number iter and k on chromosome length, $E_i(t)$

$$\begin{aligned}
k &= 1/(1 + l_{\text{chrom}}) \\
T(\text{iter} + 1) &= 0.9 \cdot T(\text{iter}), \quad T(0) = T_0 \text{ if } \text{iter} \text{ is multiple of predefined } k_{\text{gen}} \\
E_i(t) &= -J_{\text{total}}
\end{aligned}$$

Only the fittest chromosomes survive when *GSO* is applied: some species may then grow, others disappear; the evolution is similar with the natural environment where different competitive species share the limited resources.

In the work described in [46], $T_0 = 17.5$. The initial population of 200 NNs, was split in 16 species; the number of members in each species ranged between 5 and 20. The label of one species is given by $-n$ – the number of neurons in the *RAFNN* and it belongs to integer numbers between 2 and 15. The population is generated using a flip coin probability. A very good error for an *XOR* problem extended to fuzzy triangular numbers $E_{\text{max}} < 0.23$ – was got just after 4 iterations; this means the *RAFNN* dynamics was already stabilized. And this stabilization was kept during the whole test, for 1000 iterations namely.

In case of *VLGGA* optimized learning algorithm, the optimal solution appeared just only after 4 iterations and the constant behavior appeared after a magnitude order of 10^2 , meanwhile in case of gradient method that was used in [2], the solution appeared after around 5×10^4 iterations. The improvement by more than one order of magnitude proved that a *VLGGA* relied learning solution is significant better than the gradient solution.

Usual, the test stage is considered successful as in [72], if the output of selected neuron (in our case the last neuron of the *RAFNN*) produces a correct extended *XOR* delayed during a sufficient large number of steps. In our case, the number of steps is 1000 and the delay is set to 2 steps.

Another interesting aspect – namely, a typical one to *VLGGA* – was the evolution of number of species in the population, along the training period of time: the *VLGGA* moderates the pressure over variety of individuals in order to avoid excessive elitism in species.

The *VLGGA* introduced in this chapter is different from [15] that consider chromosomes of variable length, but the number of members of population is constant and the crossover is applied for members of the species indifferent of the lengths of chromosomes.

VLGGA application area may be extended to other *HIS* envisaging other *NN* architectures using non-symmetric triangular fuzzy number (three values coded for each fuzzy parameter). Another possible further extension could be made to other architectures that use crisp number but also fuzzy numbers (i.e. multiplayer perceptron and *RBF*). The structure of connection of neurons can be considered as a matrix of 1 and 0 values. A further application may include this parameter as the third encoding field in the space of solutions (the first two are the same as in this actual *VLGGA* – the number of *NN* neurons and the matrix of fuzzy *NN* weights), namely the connection gene field might be supplementary considered in form of the matrix of connections (a matrix of

0s and 1s in the typical sense to a classical notation in graph representation by matrices of connection [8]).

3 Genetic Algorithms Based Hybrid Intelligent Systems

The oldest branch of Evolutionary Computation – the *GA* namely – is in the same time the most spread one in real-world applications of *HIS*. Chapter 3 of this work tries to introduce some of these applications. The introduced applications were so selected as to illustrate the efficiency of *GA* used in combination with other intelligent technologies under the requested circumstances by the case.

3.1 GA Based Hybrid Intelligent Systems in Evolvable Hardware Implementation

Evolvable Hardware (EHW) has a central position in the large frame of *EC* applications because the hardware implementation of both genetic coding and artificial evolution methods led to a radical structural change of technology as a result. It means coming up of a new generation of machines. These machines evolve to attain a desired behavior, namely they have a *behavioral computational intelligence*. One of the main *EHW* implications on engineering design and automation is a conceptual/procedural one: no more difference exists between adaptation and design concerning these machines, these two concepts representing no longer opposite concepts [45].

EHW Implications on Engineering Design and Automation

A definition of *EHW* may be as follows: a sub-domain of artificial evolution represented by a design methodology (consortium of methods) involving the application of *EA* to the synthesis of digital and analog electronic circuits and systems [21]. A more agreed definition among the practitioners might be: *EHW* is programmable hardware that can be evolved [71].

This new design methodology for electronic circuits and systems is not a fashion. It is suitable to special uncertain, imprecise or incomplete defined real-world problems, claiming a continuous adaptation and evolution too. An increased efficiency of the methodology may be obtained by its application in the *HIS* framework that means in aggregation with other intelligent technologies such as *FS*, *NN* and *AIS*. The reason of using *EHW* in the above mentioned type of applications is its main advantage over the traditional engineering techniques for the electronic circuit design, namely the fact that the designer's job is very much simplified following an algorithm with a step sequence as below:

STEP 1 – problem specification

- 1.1 – requirements specification of the circuit to be designed
- 1.2 – specification of the basic (structural) elements of the circuit

STEP 2 – genome codification

- an adequate (genotypic) encoding of the basic elements to properly achieve the structural circuit description

STEP 3 – fitness calculation

- specification of the testing scheme used to calculate the genome fitness

STEP 4 – evolution (automatically generation of the required circuit)

- generation of the desired circuit

The designer himself is involved by acting directly during the first three steps, while the fourth step is automatically generation of the circuit. The flow modality of both step 3 and step 4 leads to same categorizing classes criteria for *EHW*.

The idea of applying evolution to artificial systems has surfaced some decades ago, but the technology available at the time was not proper to implement this methodology in hardware. Both computing technique development with increasing computational power and the appearance of programmable integrated circuits, make possible for most companies to evolve circuits as the case would be.

The second main *EHW* implication on engineering design and automation is consisting of the good circumstance for electronics engineers, this profession being deeply changed: evolutionary approaches applied by *advanced reconfigurable hardware* to electronics make hardware architectures as malleable as software, evolution don't care about complexity as the evolving system works, the only limits to electronics design are the limits imposed by our own understanding.

EHW Architectures – The Ideal Environment for Implementing the Hardware Hybridization of Different Intelligent Techniques

Some examples of *EHW* systems are applied nowadays in well known areas such as: analog and digital electronic circuits, cellular machines, controllers for autonomous mobile robots, pattern recognition and special *NNs* with dynamic topologies.

The common framework of behavioral *AI* and of *SC* made possible the implementation of hardware circuitry with intrinsic logic elements specific to *EC*. This was called by real-world applications as those typical in environments where human capacity of intervention is limited – nuclear plants, space applications, etc. [16, 37, 40, 65]. This chapter will refer to silicon *EHW*, but

we underline that much reported research began to focus on wetware and nanotechnology *EHW* too.

The main types of hardware architectures with intrinsic *EC* logic elements are:

- embryological architectures
- emergent functionality architectures
- evolvable fault tolerant systems
- parallel evolvable architectures (of Higuchi type)

The specialized **F***ield P***rogrammable G***ate A***r***ray (FPGA)* for embryological architectures is in essence a lattice substrate of *multiplexer cells* with associated *RAM*. This architecture is suitable for hardware implementation of both combinatorial and sequential complex logic circuits. Circuits are grown in silicon by architectures that are *RAM*-reprogrammable *FPGA* – a two-dimensional array consisting of dedicated (specialized cells) that can be interconnected by general interconnection resources [28].

Emergent functionality architectures allow on-line evolution by real time development of a new functionality and new behavioral complexity for some autonomous agents typically in robotics or pattern recognition. Two main types of such architectures are known: co-operation architectures and subsumption architectures [65].

An evolvable fault tolerant systems was described in [69] that was evolved to control a real robot. It is an evolvable finite state machine held in *RAM*.

Parallel evolvable architectures (of Higuchi type) are a type of architecture with intrinsic *EC* elements – a real time reconfigurable hardware (**E***volvable H***ardW***are – EHW*) as proposed by Higuchi [16, 17, 18]. We will call this architecture in this paragraph of the book just as *EHW*. It has the greatest applicability nowadays: real time adaptivity of typical control systems in robotics or pattern recognition for example.

EHW is an ideal environment for implementing the hardware hybridization of different intelligent techniques (**F***uzzy L***ogic – FL**, **N***eural N***etworks – NN**, **G***enetic A***lgorithms – GA**) with **SC** learning methods (typically reinforcement learning or genetic learning), on a massive parallel architecture. This *HIS* technique confers very interesting and unique properties on *EHW* architectures: real time adaptation as a reaction to the work of the external environment (by real time modifying the architecture's structure) and robustness (slow decrease of the performances due to the environmental perturbations or hardware faults).

The concept of *EHW* architectures includes *three main component blocks* built in the same massive parallel manner – each block composed by the same number of specific sub-blocks (see Fig. 4, where just one of the *n* channels of sub-blocks was – for simplicity reasons – represented):

- the evolutionary component (*GA*) – a general structure, independent of application (a block of identical sub-blocks parallel implementing *GA*, with

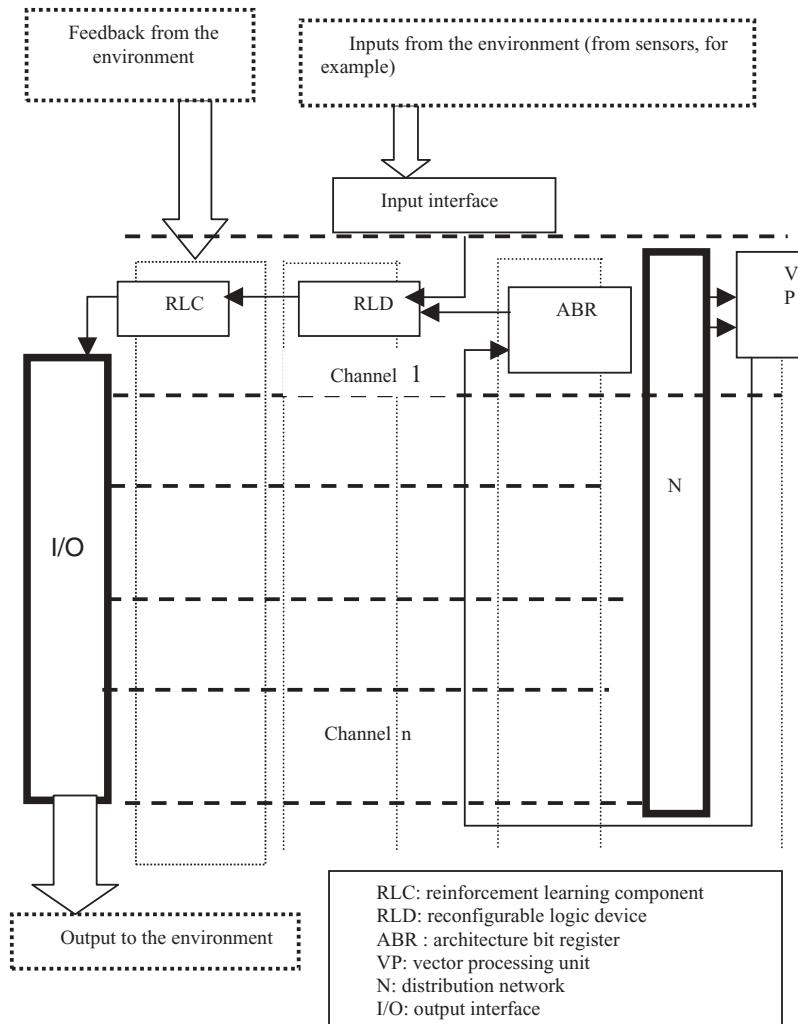


Fig. 4. The parallel evolvable architecture (after Higuchi)

GA operators acting bit wise, that means bit by bit action in parallel, on different chromosomes); the *GA* block computes by genetic operations the architecture bits for *RLD* block.

- the hardware reconfigurable component (*RLD*) – a general *FPGA*-relied structure, independent of application; this is a block of identical sub-blocks allowing real time change of configuration and functionality in hardware, as a response related to the behavior in the external environment; the *RLD* outputs depend on inputs and the architectural bits as provided from *GA*; the outputs of *RLD* are learned by the *RLC* component

- the learning component (*RLC*); this block has an application dependent architectural structure and usually is externally implemented; the *RLC* computes the reinforcement signal (reward or penalty) as an environment reaction for adaptively modifying the fitness function of *GA* chromosomes

Higuchi *EHW* architecture typically performs learning of the table representing the relation between the inputs and outputs, an usual application both in complex control systems of robots and in pattern recognition.

The hardware implementation of *GA*, especially of those devoted to *EHW* architectures, arises some problems because of the required compromise between complexity of efficient *GA* and decreased architectural complexity that can solve the optimization of hardware devices [31]. Good solutions of this problem seem to be tackling for the further research a simplified hardware implementation of efficient, but complex *GA* such as *Nagoya GA* or even *VLGGA*.

Real-time reconfigurability in *EHW* systems means changes in their architectures, as a reaction to the variable external environmental conditions and as a privilege conferred by the properties of the technological support of these circuits. This approach is a step further than the standard approaches so far, where adaptation only affects software structures.

RLD is a principal component part of Higuchi *EHW* type of architectures. As previously mentioned, the general structure of *RLD* block is based on the technological support of *FPGA*. *RLD* related simulations were made considering one of the most used *FPGA* – a *PLD* structure that is composed of a lot of identical macro-cells. A *GAL16V8 PLD* circuit, like those delivered by Lattice Semiconductors, was simulated in [41, 42, 43].

The tool selected for this application was a *GA with local improvement of chromosomes*. The main reason of local improvements in chromosomes is just the application of *genetic operators at the level of each group gene*, see [11, 74].

Experimental results were obtained [41, 43], as a comparison of the solutions found to this application problem by using both a classical Goldberg's *GA* (*SGA*) and a *GA* of Nagoya type (*NGA*) as thought in the two above mentioned works. The improved *NGA* used in the work had $m = 5$ – the copies number for a group gene, and the number of group genes (segments) was $N = 8$, namely the same as the number of columns in the connection matrix of *GAL16V8* circuit. In contrast with the Nagoya algorithm used in [11, 74], after applying the Nagoya mutation on all chromosomes, *Step 3* of the improved *NGA* includes the mutation too, not the reproduction and crossover only, the parents selection type and the replacement of the population for the new generation being kept.

3.2 GA Based Hybrid Intelligent Systems in Bioinformatics

A *GA-NN HIS* was presented in [62] that performs the identification of *Escherichia Coli* (*E.Coli*) promoter sequences in strings of *DNA*. As each gene in

DNA is preceded by a promoter sequence, the successful location of an *E.Coli* promoter leads to the identification of the *E.Coli* gene in the *DNA* sequence. A promoter is a region in the *DNA* where the transcription initiation takes place.

The combination (aggregation) function used in *MultiNNProm* combines the classification results of the individual classifiers by calculating the entropy of each output and then aggregating them through a weighted sum. The values of the aggregating weights were obtained by initializing a random population of weights.

Then a GA runs in order to obtain an optimal set of weights, by using the classification accuracy of the combined classifier as fitness value. Genetic algorithms are able to search a large space of candidate hypotheses to identify the best hypothesis. They have been proven to be useful in situations where an evaluation method to calculate the fitness of a candidate hypothesis is available.

An overview of this *GA-NN HIS* – as from Fig. 5 – shows that the system is a *NN* based multi-classifier system.

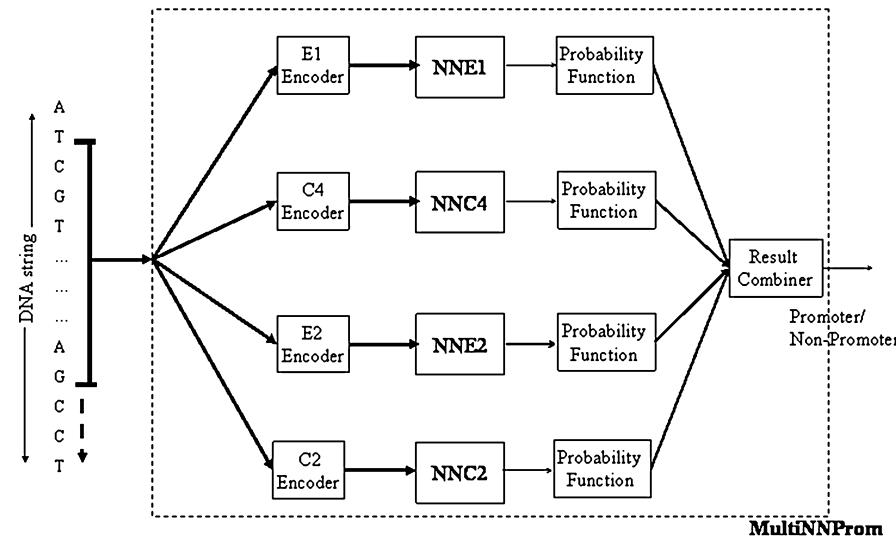


Fig. 5. The *GA-NN HIS* architecture of *MultiNNProm*

Four *NNs* were developed, that are called *NNE1*, *NNE2*, *NNC2* and *NNC4*, into which the same *DNA* sequence is inputted using four different encoding methods: *E1*, *E2*, *C2* and *C4* respectively. The outputs of the individual *NNs* are then passed onto a probability builder function, which assigns probabilities as to whether the presented sequence is a *E.Coli* promoter or not. Finally, the outputs of the probability functions are aggregated by a result combiner, which combines the results and produces a final result as to whether the given

sequence is an *E.Coli* promoter or not. The final output is of the “yes” or “no” form.

The combination of results obtained through multiple *NN* classifications was made by using an aggregation method called *LOP2*. This proposed method is a modification of the **L**ogarithmic **O**nion **P**ool (*LOP*) [14] – which proved to be more adequate and produce better results when using GA. More regarding the methods for combining classifiers and on *LOP2* especially are to be found in [62].

3.3 HS for NN Compiling into Fuzzy Rules using GA

NN lack of human comprehensibility and therefore lack of explanation ability, make them to act as a black box to the users, as well known. A *GA* based HIS method as proposed in [53] is useful therefore for *NN* compiling (compiling of captured knowledge in the *NN* weight matrix) into a set of fuzzy rules (an equivalent *FS* to the *NN*). The method improves *NN* based systems (see the neural controllers) conferring them explanation facilities: a taken decision in a neural controller can be presented to the user in form of fuzzy rules as a consequence of this method. The *GA* is used to solve two distinct tasks of this *NN* compiling method:

- first, the *GA* is used for finding the best *FS* hierarchical structure (the fuzzy rules) [54], that are equivalent to the *NN*
- second, after the first task was achieved, the *GA* acts for tuning the shapes and the number of linguistic terms (membership functions) of the hierarchical structure found at step 1

The optimization goal of *GA* based *NN* compiling into a *FS* is to minimize the difference between the *NN* and its equivalent *FS* that was got after the first two tasks are achieved

A step-by-step description of the *GA* used in [53] for mapping a *NN* into its equivalent *FS* looks as followings:

STEP 1 – equidistant partition of input space by suitably setting a number of certain membership functions, so that they divide the input space equally

STEP 2 – calculate – by *VIA* method [53] or better by method of interval propagation – [54] – the corresponding intervals of the output to the upper bases of the membership functions of the inputs. The method of interval propagation is an alternative to *VIA* method [70]

STEP 3 – an initial population of solutions is built by randomly initializing the parameters of the lower bases of trapezoidal membership functions, based on the crisp rules got during *STEP 2*

STEP 4 – tune the shape of membership functions until the best fitness value becomes less than the target one, or searching time reaches the pre-established number of generations. In case of a stop criteria is due to reaching the limited number of generations, go to *STEP 5*; stop otherwise, because the minimal and optimal fuzzy model is reached

STEP 5 – let mutation operator to prune a membership function in one of the inputs or in the output; go to *STEP 2*

The chromosome evaluation is performed by a fitness function in form of the sum of squared errors between the output value of the *NN* and the output value of the *FS*, with the parameters taken from the current chromosome in the population.

Some of the structural elements of *GA* based *NN* compiling into a *FS* are detailed as follows: *population size* was of 80–100 members; the *chromosome* was 72 genes *real-encoded*; the crossover operator is a two-point one, one point is along the first 7 genes and the second point along the genes corresponds to the output. The mutation operator can alter any parameter of the gene. When the mutation operator alters the upper bases of a trapezoidal membership function of the input, automatically the upper bases of the corresponding membership functions are calculated (by VIA method or by interval propagation).

But a main disadvantage of above-presented method is the exponential increasing of the fuzzy rules number with the increase of input space dimension (the fuzzy rules number is given by the product of membership functions number for each input). A second *GA* was applied in such circumstances to find an *optimal hierarchical structure of the FS* in order to reduce the number of membership functions and in the same time the number of fuzzy rules.

The building strategy of this hierarchical structure consists of collecting all the inputs having closer relationship in the same fuzzy unit.

In a particular example, the total number of fuzzy rules for a *NN* having 4 inputs, 1 output and 3 membership functions on each input, is $3 \times 3 \times 3 \times 3 = 81$ fuzzy rules. If the case is of an existing relationships between input 1 and input 2 for example, and also between inputs 3 and 4, these input are grouped in a hierarchical structure as in Fig. 6. Every fuzzy unit is described by 3×3 fuzzy rules, which mean an important diminishing of the total number of equivalent fuzzy rules.

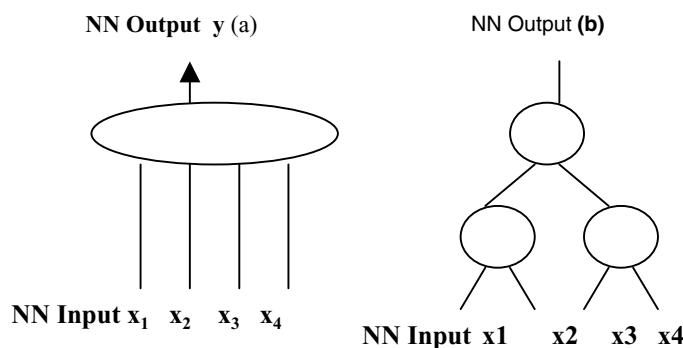


Fig. 6. An example of a black box **NN** (a), and its hierarchical structure (b)

A tree structure is to build with 2^{n-2} node units having 1, 2, 4, 8, ... units on each level in case of a *NN* that has n inputs, every *NN* input being assigned to a fuzzy unit in the tree. The fuzzy input is passed to the upper fuzzy unit in the tree if the fuzzy input has only one input.

An application was considered for *NN* with inputs $n = 5$, the fuzzy units arrangement is as in Fig. 6a and an assigning of the three units is in Fig. 6b.

The developed *GA* for such a *NN* of large sizes is composed of two sub-procedures:

STEP 1 – find the best hierarchical structure of the equivalent *FS* to the *NN*
STEP 2 – tune the shapes and the number of membership functions of the

hierarchical structure found during *STEP 1*

The chromosome is so encoded [64] that each gene locus corresponds to a *NN* input and the gene allele is the number of the fuzzy unit which is assigned to that input, see Fig. 7.

The decoded corresponding *NN* hierarchical structure to example from Fig. 7 is as in Fig. 8:

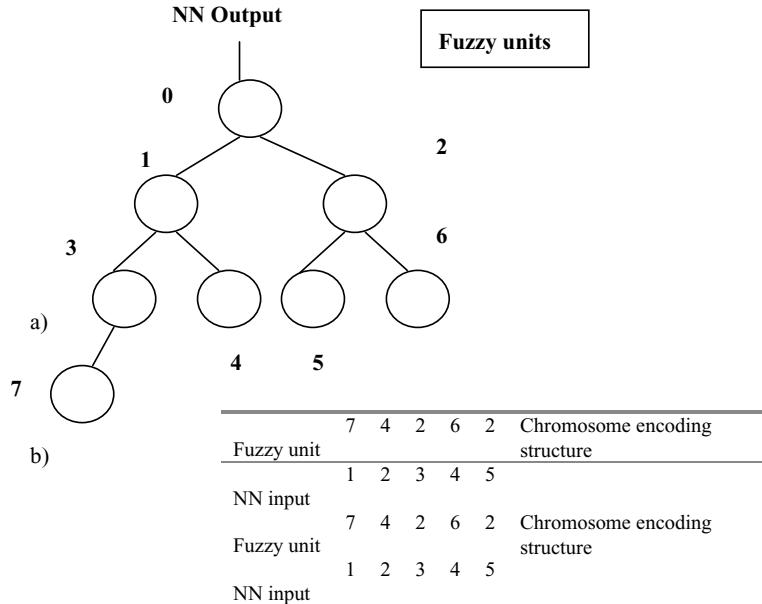


Fig. 7. (a) The fuzzy units tree corresponding to a *NN* with n inputs (b) The associated chromosome structure – the sequence of assigned (fuzzy unit) numbers to *NN* inputs

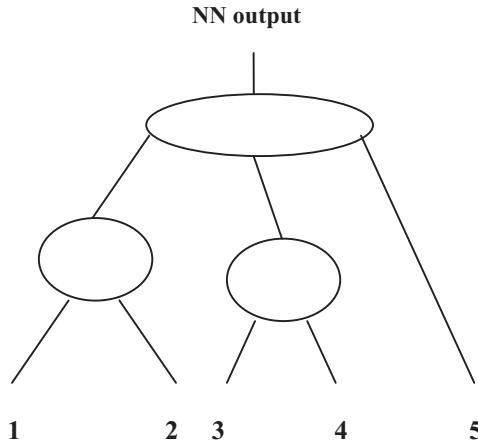


Fig. 8. Decoded corresponding NN hierarchical structure to coding in Fig. 3

4 Neuro-Fuzzy Based Hybrid Intelligent Systems for Fault Diagnosis

The conceptual diagram for a fault diagnosis system consists of two sequential steps: residual generation and residual evaluation. In the first step a number of residual signals are generated in order to determine the state of the process. The objective of fault isolation is to determine if a fault has occurred and also the location of the fault, by an analysis on the residual vector.

Many authors have focused on the use of *NN* in **Fault Detection and Isolation (FDI)** applications [25, 29] for solving the specific tasks in *FDI*, such as fault isolation but mainly fault detection. Other authors [26] used *FS* for fault diagnosis, especially for fault isolation, but some of them even for fault detection, using for example *TSK* fuzzy models. In the last few years there is also an increasing number of authors [5, 59] who try to integrate *NN* and *FS* in form of *NN-FS HIS* in order to benefit of the advantages of both techniques for fault diagnosis applications.

NN have been successfully applied to fault diagnosis problems due to their capabilities to cope with non-linearity, complexity, uncertainty, noisy or corrupted data. *NN* are very good modeling tools for highly non-linear processes. Generally, it is easier to develop a non-linear *NN* based model for a range of operating than to develop many linear models, each one for a particular operating point. Due to these modeling abilities, *NN* are ideal tools for generating residuals. *NN* can also be seen as universal approximators.

The drawback of using *NN* for classification of faults is their lack of transparency in human understandable terms. *FS* techniques are more appropriate for fault isolation as it allows the integration in a natural way of human operator knowledge into the fault diagnosis process. The formulation of the

decisions taken for fault isolation is done in a human understandable way such as linguistic rules.

The main drawback of *NN* is represented by their “black box” nature, while the disadvantage of *FS* is represented by the difficult and time-consuming process of knowledge acquisition. On the other hand the advantage of *NN* over *FS* is learning and adaptation capabilities, while the advantage of *FS* is the human understandable form of knowledge representation. *NN* use an implicit way of knowledge representation while *FS* and *NN-FS HIS* represent knowledge in an explicit form, such as rules.

4.1 Methods of NN-FS Hybridization in Fault Diagnosis

The *NN-FS* hybridization can be done in two main ways.

- (a) *NN* are the basic methodology and *FS* is the second. These *NN-FS HIS* are mainly *NN* featured but the *NN* are equipped with abilities of processing fuzzy information. These *HIS* are also termed *FS-NN Networks* and they are systems where the inputs and/or the outputs and/or the weights are fuzzy sets, and they usually consist of a special type of neurons, called fuzzy neurons. Fuzzy neurons are neurons with inputs and/or outputs and weights represented by fuzzy sets, the operation performed by the neuron being a fuzzy operation.
- (b) *FS* is the basic methodology and *NN* the subsequent. These *HIS* systems can be viewed as *FS* augmented with *NN* facilities, such as learning, adaptation, and parallelism. These *HIS* systems are also called *NN-FS HIS*. Most authors in the field of neuro-fuzzy computation understand neuro-fuzzy systems as a special way to learn *FS* from data *using NN* type learning algorithms. Some authors [63] term these *NN-FS HIS* systems also *fuzzy NN*, but most of them like to term them as *NN-FS HIS*. *NN-FS HIS* can be always interpreted as a set of fuzzy rules and can be represented as a feed-forward network architecture.

These two previous ways of *NN-FS* fuzzy hybridization can be viewed as a type of fusion *HIS*, as it is difficult to see a clear separation between the two methodologies. One methodology is fused into the other methodology, and it is assumed that one technique is the basic technique and the other is fused into it and augments the capabilities of information processing of the first methodology. Besides these fusion *NN-FS HIS* systems, there is another way of hybridization of *NN* and *FS*, where each methodology maintains its own identity and the hybrid *NN-FS* system consists of a modules structure which cooperate in solving the problem. These kind of *NN-FS HIS* are called combination *HIS* systems. The *NN* based modules can work in parallel or serial configuration with *FS* based modules and augments each other. In some approaches, a *NN* – such as a self-organising map – can preprocess input data for a *FS*, performing for example data clustering or filtering noise. But, especially in *FDI* applications, many authors use a *FS* as a pre-processor for a

NN. In [1] the residuals signals are fuzzified first and then fed into a recurrent *NN* for evaluation, in order to perform fault isolation.

The most often used *NN-FS HIS* systems are fusion *HIS* and the most common understanding for such a system is the following. A *NN-FS* fusion *HIS* is a *NN* which is topologically equivalent to the structure of a *FS*. The network inputs/outputs and weights are real numbers, but the network nodes implement operations specific to *FS*: fuzzification : fuzzy operators – conjunction, disjunction-; defuzzification. In other words, a *NN-FS* fusion *HIS* can be viewed as a *FS*, with its operations implemented in a parallel manner by a *NN*, and that's why it is easy to establish a one-to-one correspondence between the *NN* and the equivalent *FS*. *NN-FS* fusion *HIS* can be used to identify *FS* models directly from input-output relationships, but they can be also used to optimize (refine/tune) an initial *FS* model acquired from human expert, using additional data.

4.2 Residual Generation Using Neuro-Fuzzy Models

The purpose of this sub chapter is to present a *NN-FS HIS* application to detect and isolate mainly actuator faults, but also other types of faults such as components or sensor faults, occurred in an industrial gas turbine. In this turbine model, air flows via an inlet duct to the compressor and the high pressure air from the compressor is heated in combustion chambers and expands through a single stage compressor turbine. A *Butterfly valve* provides a means of generating a back pressure on the compressor turbine (there is no power turbine present in the model). Cooling air is bled from the compressor outlet to cool the turbine stator and rotor. A *Governor* regulates the combustor fuel flow to maintain the compressor speed at a set-point value. For simulation purposes we used a *Simulink* prototype model of such an industrial gas turbine as presented in [58] and developed at ABB-Alstom Power, United Kingdom. The *Simulink* prototype simulates the real measurements taken from the gas turbine with a sampling rate of 0.08 s. The model has two inputs and 28 output measurements which can be used for generating residuals. The *Simulink* model was validated in steady state conditions against the real measurements and all the model variables were found to be within 5% accuracy. Different *NN-FS HIS* models were developed for generating residuals purposes, but all of them are driven by two inputs: valve angle (*va*) and the fuel flow (*ff*) which is also a control variable.

One common fault in the gas turbine is the fuel actuator friction wear fault. Other faults considered were compressor contamination fault, thermocouple sensor fault and high-pressure turbine seal damage. Usually these faults in the industrial gas turbine develop slowly over a long period of time. During the simulations, was tried to detect the actuator fault and to isolate it from other faults in the turbine. For simplicity reasons, this sub chapter presents only just the results for the first two faults – fuel actuator friction wear fault and compressor contamination fault.

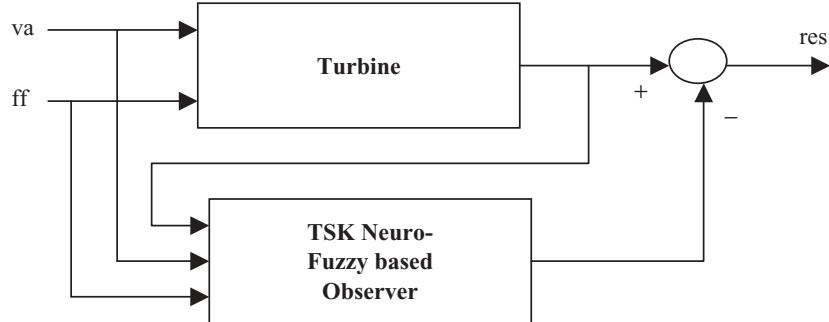


Fig. 9. *NN-FS HIS* based observer scheme for generating residuals

The residual signals are given by the difference between estimated signal given by observer and the actual value of the signal, see Fig. 9. The residuals are generated using *TSK NN-FS HIS* networks. A main concern was focused on how to find accurate *NN-FS HIS* models for generating residuals, but was tried to have as much as possible a certain degree of transparency of the models. That's why it had to be found a good structure of the *NN-FS HIS* models and therefore a good partition of the input space, using clustering techniques for that. There is a compromise between the interpretability and the precision of the *NN-FS HIS* model.

First a 3 inputs and 1 output *TSK NN-FS HIS* network structure was developed for the output measurement which is most affected by the actuator fault (*ao*). The input of the network uses the present value for valve angle (*va*) and fuel flow (*ff*), and the previous value is used as the output affected by the fault. Three linguistic values were used for each input variable and grid partition of the input space. See [55] for more details regarding the performance of the model, the generated residual and the difference between the system output and the model output. Unfortunately, due to the control loop action, this kind of fault can not be seen in steady state regime and can not be described as a ramp function in order to show what's happened in case of a gradual developing fault. But this actuator fault can be seen in dynamic regime, for different values of the valve angle. For isolating purposes the absolute value of this residual signal was taken and the pass through a filter for obtaining a persistent residual signal. In order to see a gradually developing fault, a *NN-FS HIS* based observer was built for the output most affected by the compressor contamination fault.

In a similar way was also used a *NN-FS HIS* with 3 inputs and 3 membership functions per input and first order linear models in the consequent of the rules. The output (*co*) most sensitive to a compressor ramp fault was depicted and the residual generated too. The compressor fault affected also the output (*ao*) – most affected output by the actuator fault – but not in the

same magnitude as the output (co) was affected. Then, the residual designed for output (ao) is sensitive to both faults. See [55] for more details.

Several other types of models were developed for the residual sensitive to the actuator fault, in order to see a comparison between the accuracy and the transparency of the model.

4.3 Neuro-Fuzzy Based Residual Evaluation

In the residual generation part of a diagnosis system the user should be more concerned on the accuracy of *NN-FS HIS* models, even desirable to have interpretable models also for residual generation, such as *TSK NN-FS HIS* models.

For the evaluation part it is more important the transparency or the interpretability of the fault classifier, in human understandable terms, such as classification rules.

The main problem in *NN-FS HIS* fault classification is how to obtain an interpretable *FS* classifier, which should have few meaningful fuzzy rules with few meaningful linguistic rules for input/output variables. *NN-FS HIS* structured networks for Mamdani *FS* models are appropriate tools to evaluate residuals and perform fault isolation, as the consequence of the rules contains linguistic values which are more readable than linear models in case of using *TSK FS* models. As mentioned in the previous section, a price was paid for the interpretability of the fault classifier: the loss of the precision of the classification task.

More details regarding the experiments and their concluding remarks regarding based residual evaluation are in [55]. For training the *NN-FS HIS* network in order to isolate these faults, 150 patterns for each fault were used. The *NN-FS HIS* network decisions for the residual values were assigned in relation with the known faulty behavior. In order to obtain a readable fault classifier, the NEFCLASS – a *NN-FS HIS* classifier was used [32]. This *NN-FS HIS* has a slightly different structure than the *NN-FS HIS* for Mamdani models presented in the previous section, but it allows to the user to obtain in an interactive manner and very easy an interpretable *FS* fault classifier, at the desired level and compromise accuracy/transparency. Conjunctive fuzzy rules for fault classification (both residual inputs in the antecedent), were considered.

Table 3 summarizes the results of the simulation experiments featuring transparency/accuracy of *NN-FS HIS* fault classifiers.

Case Example: Fault Detection and Isolation of an Electro-Pneumatic Valve

The early detection of faults – just beginning and still developing – can help avoid system shutdown, breakdown and even catastrophes involving human fatalities and material damage. The mathematical model used in the traditional **Fault Detection and Isolation (FDI)** methods is sensitive to modeling

Table 3.

| Transparency (no. of rules) | 2 | 4 | 8 | 12 |
|---|------|------|------|------|
| Accuracy (no. of patterns correctly classified – in %) | 88.7 | 91.2 | 96.4 | 99.6 |

errors, parameter variation, noise and disturbance. Process modeling has limitations, especially when the system is complex and uncertain and the data are ambiguous i.e. not information rich. Different **Computational Intelligence (CI)** methods – **Neural Networks (NN)**, **Fuzzy Systems (FS)**, **Evolutionary Algorithms (EA)** are known to overcome some of the above mentioned problems (Patton et al. 2000).

The *NN-FS HIS* models combine, in a single framework, both numerical and symbolic knowledge about the process. Automatic linguistic rule extraction is a useful aspect of *NN-FS HIS* especially when little or no prior knowledge about the process is available [4, 19]. For example, a *NN-FS HIS* model of a non-linear dynamical system can be identified from the empirical data. This model can give some insight about the nonlinearity and dynamics properties of the system. But *NN-FS HIS* networks by intrinsic nature can handle just a limited number of inputs. When the system to be identified is complex and has large number of inputs, the fuzzy rule base becomes large. The *NN-FS HIS* models usually identified from empirical data are also not very transparent. Transparency accounts a more meaningful description of the process i.e. less rules with appropriate membership functions. Hierarchical *NN-FS HIS* networks can be used to overcome the dimensionality problem by decomposing the system into a series of *MISO* and/or *SISO* systems called hierarchical systems [67]. The local rules use subsets of input spaces and are activated by higher level rules.

The function of the *NN-FS HIS* model in the *FDI* scheme is also important i.e.: Preprocessing data, Identification (Residual generation) or classification (Decision Making/Fault Isolation). For example a *NN-FS HIS* model with high approximation capability and disturbance rejection is needed for identification so that the residuals are more accurate. Whereas in the classification stage, a *NN-FS HIS* network with more transparency is required.

The following characteristics of *NN-FS HIS* models are important: Approximation/Generalization capabilities; Transparency – Reasoning/use of prior knowledge/rules; Training Speed/Processing speed; Complexity; Transformability – To be able to convert in other forms of *NN-FS HIS* models in order to provide different levels of transparency and approximation power; Adaptive learning. Two most important characteristics are the generalizing and reasoning capabilities. Depending on the application requirement, usually a compromise is made between the above two.

The valve considered for *FDI* in this case example is an electro-pneumatic flow controller in the evaporation stage of a sugar factory. For this application was built a non-linear mathematical model of the valve using *Simulink* and *MATLAB*. The model is then used to generate faulty/fault-free data to evaluate different *NN-FS HIS* based fault isolation schemes. The whole valve assembly consists of 3 main parts as in Fig. 10:

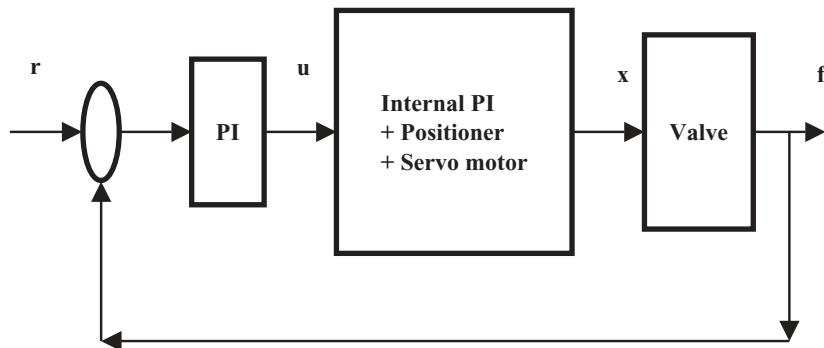


Fig. 10. Main parts of the valve assembly

The PI controller – controls the *Positioner & Actuator* output to regulate the flow through the valve.

Positioner and Actuator – Pneumatic pressure is applied to the servomotor diaphragm to control the stem position that changes the flow. The *Positioner* adjusts this pressure input to the servomotor to obtain the correct stem position of the *actuator*.

The Valve – is the final element in the assembly which alters the flow according to the stem position.

The following list of faults is considered in the valve actuator assembly:

- f_1 – External *PI* controller proportional gain fault
- f_2 – External *PI* controller integral gain fault
- f_3 – Increased friction of the servomotor
- f_4 – Decreased elasticity of the servomotor
- f_5 – Decrease of pneumatic pressure
- f_6 – Internal *PI* controller fault
- f_7 – Internal position sensor fault
- f_8 – Valve clogging
- f_9 – Valve leakage
- f_{10} – Chocked flow

Two *NN-FS HIS* models were used here with transparent structure. A *TSK* structure with linear dynamic models as consequents is used to approximate the internal *PI* controller, the *Positioner* and *Servomotor*. The identified *TSK*

model has three locally linear models as consequents. The time constants of these local models are 18 sec, 12 sec and 8 sec, respectively, which show that the *NN-FS HIS* system is faster at high values of flow and slower at the low values. More details regarding the performance of the *TSK* model in closed-loop, parallel to the system are to be found in [56].

The changes in physical parameters e.g. time constant (r_{TC}), static gain (r_{SG}), static offset (r_{SO}) and settling time (r_{ST}) are computed by use of the local models, and $RLSE$. These changes are the residuals which can be used for fault isolation.

A Linguistic/Mamdani *NN-FS HIS* model is identified to approximate the valve. The model input is the stem position x and the output is volumetric flow rate f . The control input u is predicted from input set-point flow and measured flow, by integrating and using *RLSE*. *GK*-clustering algorithm [13] is used to partition the input space, where clusters are projected onto the I/O space to find membership functions. A gradient-based optimization method is used to fine-tune the membership functions. More details on both clustered valve data and on the tuned *I/O* membership functions for Mamdani model are to be found in [56]. The predicted values u, x, f and the measured values are used to generate the residuals r_u, r_x, r_f . Fault isolation table given in Table 4 shows that some faults could only be detected during the time when the valve is being opened and closed. Moreover, choked flow could only be detected at high values of flow.

Table 4.

| | f ₁ | f ₂ | f ₃ | f ₄ | f ₅ | f ₆ | f ₇ | f ₈ | f ₉ | f ₁₀ |
|-----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|-----------------|
| r _u | Op | Op | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
| | + | - | | | | | | | | |
| | Cl | Cl | | | | | | | | |
| | - | + | | | | | | | | |
| r _x | ~ | ~ | ~ | ~ | ~ | ~ | Ch | ~ | ~ | ~ |
| r _f | ~ | ~ | ~ | ~ | ~ | ~ | ~ | Op | Op | - |
| | | | | | | | | + | - | |
| | | | | | | | | Cl | Cl | |
| | | | | | | | | - | + | |
| r _{ST} | ~ | ~ | + | + | 0 | - | ~ | ~ | ~ | ~ |
| r _{TC} | ~ | ~ | + | - | + | - | ~ | ~ | ~ | ~ |
| r _{SO} | ~ | ~ | 0 | 0 | 0 | + | ~ | ~ | ~ | ~ |

5 Hybrid Intelligent Systems for Educational Purposes: WITNEeSS (Wellington Institute of Technology Novel Expert Student Support) – An Intelligent Tutoring System (ITS)

HIS by its main *CI/AI* techniques will be the main field of knowledge involved in development of ultimate learning systems, but other disciplines will be embedded too: psychology, human-computer interface technologies, knowledge representation, databases, system analysis and design and not at last, advanced computer programming technologies.

Actually electronic learning systems would have the student change to fit the learning system, but their flexibility requires for *the learning systems to be able to change to fit the student needs*. So the ultimate learning systems must interact with a student in a way that is a little bit, but enough different. The key role for accomplishing this task inside the ultimate learning systems is played by *HIS* technologies.

A good description of the basic structure of an *ITS* was made in [30]. Here are shown the main typical components to an *ITS*. When these components work together, the framework of a system is created that can recognise patterns of learn behaviour and react to them. Let us just mention the main components of an *ITS* that are the followings, as from [30]: the Expert Model [52], the Student model, the Instruction Model [3] and the Interface Model [30, 52].

The interaction of all above-mentioned components must be implemented so as providing the student with a face-to-face encounter with the knowledge of a subject domain. It is crucial that this internal *ITS* interaction to act properly so as the student being able to successfully assimilate new knowledge into his current mental schemata. There are different types of *ITSs*. It is beyond the aim of this book to make an overview on them, the book is focussed on their common fundamental features only.

Also the main problems and drawbacks that are typical to different kind of *ITSs* will be presented as a useful selection criterion of different *CI/AI* techniques aimed to improve the *ITSs* in form of hybrid intelligent systems.

Some of the main problems within the field of *ITS* will be mentioned in the following paragraph, see also [48].

The description boxes with regard to Fig. 11 are as follows:

1. When student logs on all information – long term profile, starting session profile, fuzzy decision rule profile – is loaded into the *Student Database*
2. Although the student starts with standardised profiles and rule sets, as the student interacts with the system, the system, using hybrid intelligent technologies, will fine tune these profiles and rules
3. What happens is that the system will learn how to provide the best tutoring service for a particular student. The students will end up with their own individualised profile and rule sets

Note: the numbers in brackets on the flow arrows refer to the description box of the same number.

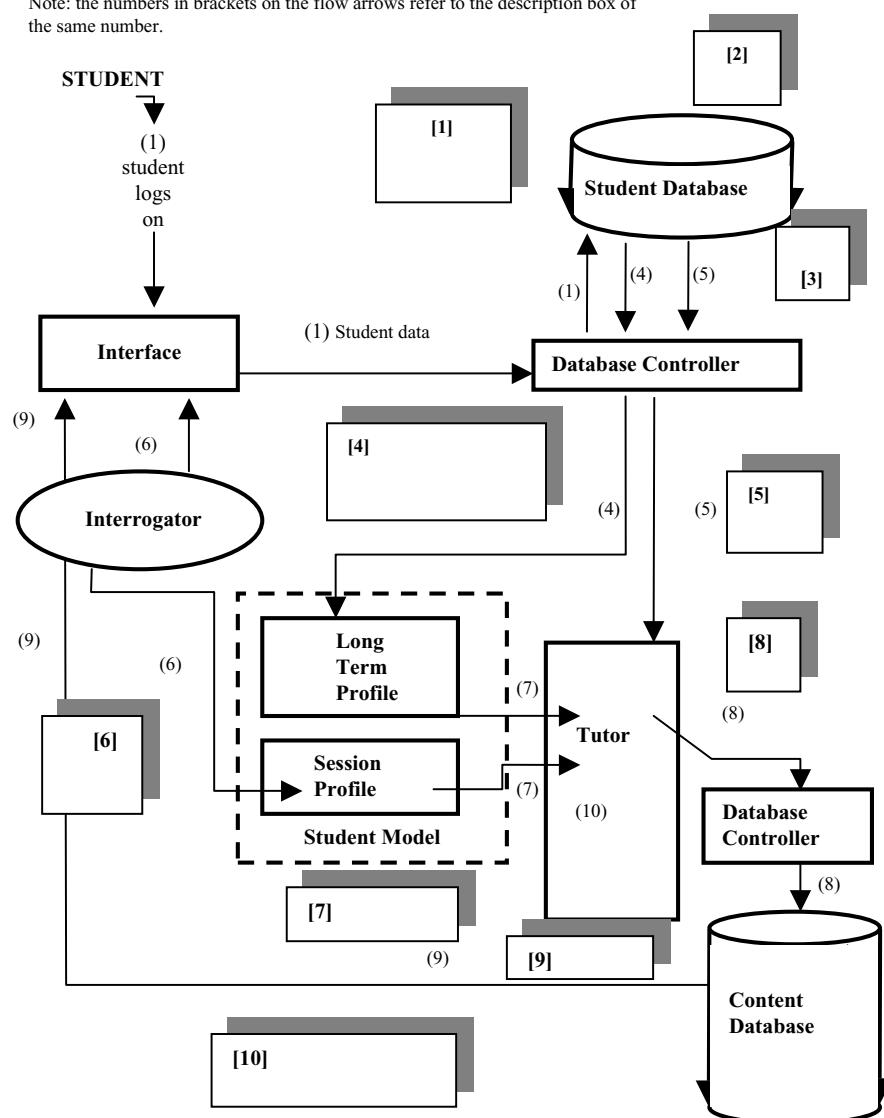


Fig. 11. WITNeSS working

4. Current information about the student plus their long-term profile is loaded into the *Student Model*
5. Student's profile of fuzzy rules for determining the next content and presentation mode is loaded into the *Tutor* module
6. The *Interrogator* starts to gather information from the interface about the student's behaviour. This is feed into the *Session Profile* module of the student model
7. Student current information, *Long Term Profile*, current session data is feed into the *Tutor* module
8. Using the fuzzy decision rule structure, *Tutor* determines what content modules to present to the student next
9. Content *Database Controller* arranges for the appropriate modules to be sent to the student interface
10. When the final data about the session comes back from the *Interface* via the *Interrogator* and *Student Model*, the *Tutor* module using its fuzzy-neural component – fine tunes the fuzzy rules in an effort to find a better what of making decisions on the student's behalf. The process now repeats from step [8] through to [10]

The *Student Model* is sited in into a typical *ITS* system configuration [30], and an agent we call the “*Optimizer*” was supplementary added. The system, as a whole, works in the way described above. The *Student Model* is used with an optimizer agent to “fine-tune” the linguistic variables of a fuzzy rule decision structure that is used by a *Tutor* model to decide on “*what*” should be presented next to a student and “*how*” it should be presented. There is a detailed description of how the concept of an “*optimiser*” works, see [61].

The *Optimizer* sends its “shape refined” linguistic variables back to the main *Tutor*, replacing its fuzzy rule structure with a more efficient one. The *Optimizer* then starts over, taking the latest information about how the student is performing, and works to find an even better “shape” to the linguistic variables.

What the *Optimizer* is doing step by step for performing its role, it looks as follows:

STEP 1 – The systems creates the *Optimizer*, which in turn, creates copies of the *Student*, *Tutor* and *Knowledge* agents from the main system (see type 1 arrows in Fig. 12). It also copies the current fuzzy rule decision structure (see type 2 arrow in the same figure). It then creates an initial *GA* population of 20 chromosomes. Each chromosome represents changes that can be made – (*possibility be made*), to the shape of the basic linguistic variables (see type 3 arrows in Fig. 12).

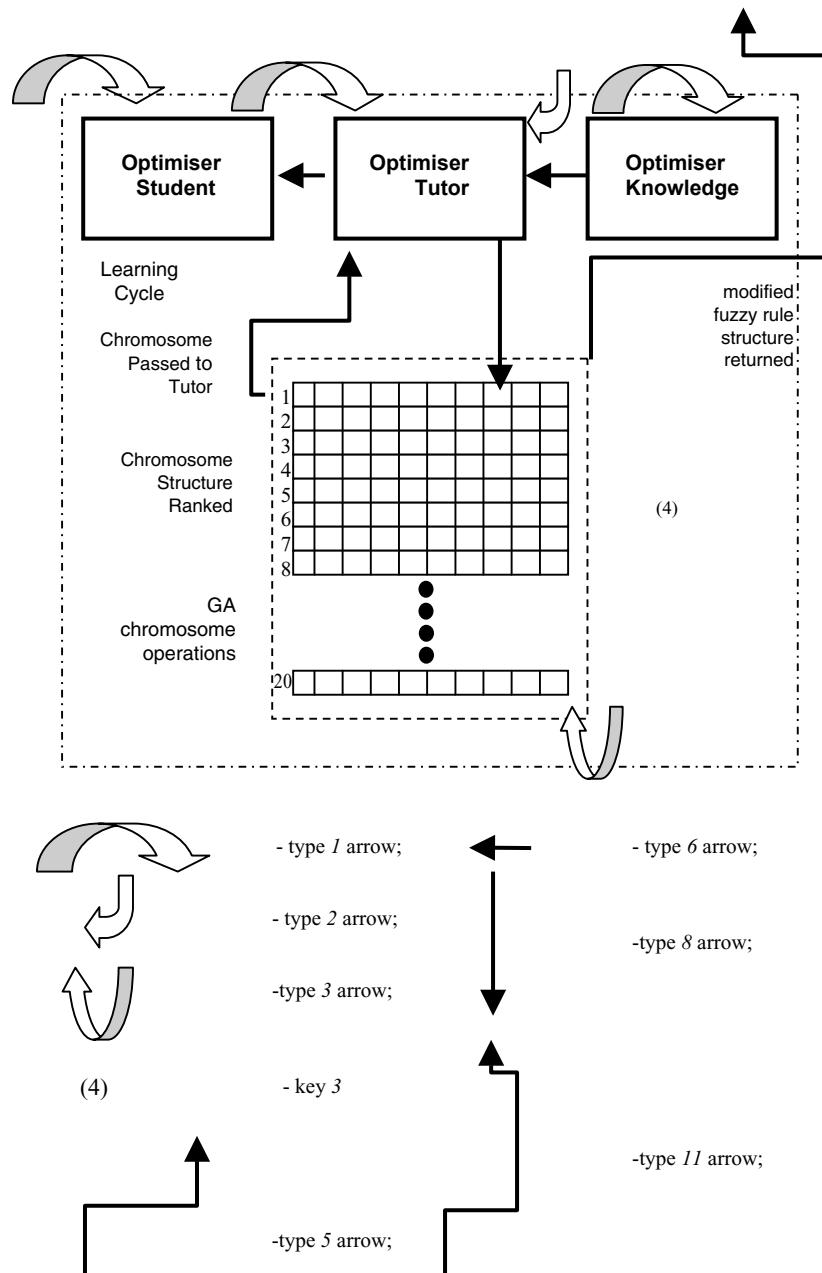


Fig. 12. How the Optimizer agent works

STEP 2 – Each chromosome can have up to 10 genes in it. Each activated gene represents a change that *can be made* (possibly *be made*) to the shape of any of the linguistic variables (see key 4 in Fig. 12).

STEP 3 – A chromosome is passed to the *Tutor* and used to modify the “shapes” of the linguistic variables resulting in a different variation of the current fuzzy rule structure (see type 5 arrows in Fig. 12).

STEP 4 – The *Tutor* uses this modified version of its fuzzy rules to take learning activities from the knowledge agent and also to present them to the student (see type 6 arrows in Fig. 12).

STEP 5 – Both keys 6 operations are iteratively repeated until the student has learnt the whole knowledge structure (see key 7 in Fig. 12).

STEP 6 – The student is tested to see how well the knowledge has been learnt. The chromosome is evaluated using a Fitness Function of the following form (also see type 8 arrows in Fig. 6.6):

$$f(t, q) = t \times q$$

where:

$f(t, q)$ – fitness value of the chromosome.

t – number of steps taken to learn content.

Q – number of errors student makes in the test

The population of chromosomes is ranked in a descending order by fitness value, the top 20 individuals are kept and the rest of them are discarded (see Fig. 12). Each new generation is operated on using *GA* operators of selection, crossover and mutation, to arrive at approximately 40 chromosomes. The whole process then goes back to *STEP 5 – Chromosome passed to Tutor*. A number of generations are produced until the stop condition has been met. The best chromosome is passed back to the main *Tutor* agent where it is used to modify the “shapes” of the linguistic variables. This gives a more efficient fuzzy rule decision structure for this particular student (see type 11 arrow in Fig. 12).

The *HIS* framework offers a new line of thinking in developing intelligent tutor systems or learning environments that can dynamically adapt its scheduling of teaching to result in quicker, more efficient tutoring of an individual student. The results of the experiments are supportive to above-mentioned remarks.

However there is work still to do in the future before *WITNeSS* comes close to a real-life intelligent tutoring system. The work must be more focused on details regarding the cost analysis of the system in terms of processing resources.

Experiments were conducted to test *WITNeSS* in its ability to optimally decide on “which” learning material to present to students and “how”, so that the students experience maximum learning. Each time *WITNeSS* was tried

out on students, the learning results recorded and analysed, and a comparison against other learning systems made.

The problem arose whereby it wouldn't be possible to organize "*human*" *students* for the experiments. This was due to time constraints and the time of the year. The question became, to create "*virtual*" *students* that could simulate certain human learning behaviour, so the *Student Object* has been created. The concept of "*virtual*" *student* is a key concept for *ITS* testing and its further development.

Of high priority will be the improvement of how the system performs when confronted with a real-life situation – real students and a more complex knowledge structure. Some work has already started in developing a student model that represents more accurately the particular features to any student who works with the system. These improved knowledge structures and student models will, for sure, add greater efficiency to the system.

While *WITNeSS*, at the moment, is relied on a *FS-GA* hybrid method, we still need to look at the possibility of other hybrid intelligent techniques to be used in making this *ITS* more efficient. For example the efficiency of pattern recognition tasks may certainly be improved by using Artificial Immune Systems.

References

1. Alexandru D, Combastel C, Gentil S (2000) Diagnostic decision using recurrent neural network. In: Proc. of the 4th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes, pp 410-415, Budapest **32**
2. Arotaritei D, Negoita Gh M (2002) Optimisation of Recurrent NN by GA with Variable Length Genotype. In: McKay B, Slaney J (eds) AI2002: Advances in Artificial Intelligence. Springer-Verlag, Berlin Heidelberg New York, pp 691–692 **9, 12, 16, 20**
3. Buiu C (1999) Artificial Intelligence in education – the State of the Art and Perspectives (ZIFF Papiere 111) Germany, Fern University, Institute for Research into Distance Education. (Eric Reproduction Service No ED434903) **38**
4. Brown M, Harris C J (1994) Neuro-fuzzy adaptive Modelling and Control. Prentice-Hall **35**
5. Calado J M F, Sa da Costa J M G. (1999) An expert system coupled with a hierarchical structure of fuzzy neural networks for fault diagnosis. J Applied Mathematics and Computer Science 3 (9) : 667–688 **30**
6. Chen W (2003) Supporting Collaborative Knowledge Building with Intelligent Agents. In: Palade V, Howlett J R, Jain L (eds) Knowledge-Based Intelligent Information and Engineering Systems. Springer-Verlag, Berlin Heidelberg New York, Part II, pp 238–244
7. Coello C A (1999) A Survey of Constraint Handling Techniques used with Evolutionary Algorithms. Lania-RI-99-04, Laboratorio Nacional de Informática Avanzada **19**
8. Cormen T H, Leiserson C E, Rivest R L (1990) Introduction to Algorithms. The MIT Press **21**

9. de Castro L N, Timmis J I (2002) Artificial Immune Systems: A New Computational Approach. Springer Verlag, London 11
10. Fagharasan F, Negoita Gh M (1995) A Genetic-Based Method for Learning the Parameter of a Fuzzy Inference System. In: Kasabov N, Coghill G (eds) Artificial Neural Networks and Expert Systems, IEEE Computer Society Press, Los Alamitos California, pp 223–226 12, 14
11. Furuhashi T, Nakaoka K, Uchikawa Y (1994) A New Approach to Genetic Based Machine Learning and an Efficient Finding of Fuzzy Rules. In: Proceedings of The IEEE/Nagoya University WWW on Fuzzy Logic and Neural Networks/Genetic Algorithms. Nagoya, Japan, pp 114–122 25
12. Goonatilake S, Treleaven P (1996) Intelligent Systems for Finance and Business. John Wiley & Sons, Chichester New York Brisbane Toronto Singapore 7, 8
13. Gustafson D E, Kessel W C (1979) Fuzzy clustering with a fuzzy covariance matrix, In: *Proc IEEE Conf Decision Contr.* San Diego, CA, pp 761–766 37
14. Hansen JV, Krogh A (1995) A General Method for Combining in Predictors Tested on Protein Secondary Structure Prediction, citeseer.ist.psu.edu/324992.html 27
15. Harvey I (1992) The SAGA Cross: The mechanism of Recombination for Species with Variable-Length Genotypes. *J Parallel Problem Solving from Nature* 2. Elsevier Science Publisher B V 15, 20
16. Higuchi T, Iba H, Manderick B (1994) Evolvable Hardware. In: Kitano H, Hendler J (eds) Massively Parallel Artificial Intelligence, AAAI Press/MIT Press, pp 339–420 22, 23
17. Higuchi T (1996) Evolvable Hardware with Genetic Learning. In: Proceedings of IEEE International Symposium on Circuits and Systems, ISCAS96. May 13, Atlanta, USA 23
18. Higuchi T, Iwata M, Kajtani I, Iba H, Hirao Y, Furuya T, Manderick, B (1996) Evolvable Hardware and Its Applications to Pattern Recognition and Fault-Tolerant Systems. In: Sanchez, E, Tomassini M (eds) Towards Evolvable Hardware. Springer Verlag, Heidelberg, pp 118–136 23
19. Jang J S, Sun R (1995) Neuro-fuzzy modeling and control. *J Proc IEEE* 83(3): 378–405 35
20. Kandel A, Teodorescu H N, Arotaritei D (1998) Fuzzy RBF Neural Network. In: Proceedings of the 17th Annual Meeting of the North American Fuzzy Information Processing Society NAFIPS'98, Pensacola Beach, Florida, USA 16
21. Kelly M (1996) Fit for Future? Evolutionary Computation in Industry. *J EvoNews* 2 : 1–3 21
22. Kelsey J, Timmis J I (2003) Immune Inspired Somatic Contiguous Hypermutation for Function Optimization. In: Cantu-Paz E, Foster J, Kalaynmooy D, Lawrence D D, Rajkumar R, O'Reilly U-M, Beyer H-G, Standish R, Kendall G, Wilson S, Harman M, Wegner J, Dasgupta D, Potter M, Schultz A, Dowsland K, Jonoska N, Miller J (eds) Genetic and Evolutionary Computation – GECCO 2003, Springer Verlag Berlin Heidelberg New York, Part I, pp 207–218 11
23. Khosla R, Dillon T (1997) Engineering Intelligent Hybrid Multi-Agent Systems. Kluwer Academic Publishers, Boston Dodrecht London 8, 10
24. Kiendl H (1994) The Inference Filter. In: Proceedings of 2nd European Congress on Intelligent Techniques and Soft Computing EUFIT'94, pp 443–447 14
25. Korbicz J, Patan K, Obuchowitcz O (1999) Dynamic neural networks for process modeling in fault detection and isolation systems. *J Applied Mathematics and Computer Science* 3 (9): 519–546 30

26. Koscieny J M, Syfert M, Bartys M (1999) Fuzzy-logic fault diagnosis of Industrial process actuators. *J Applied Mathematics and Computer Science* 3 (9): 637–652 **30**
27. Lee M A, Takagi H (1993) Integrating Design Stages of Fuzzy Systems Using Genetic Algorithms. In: Proc IEEE Int Conf on Fuzzy Systems (FUZZ-IEEE '93), San Francisco California, pp 612–617 **9**
28. Marchal P, Nussbaum P, Piguet C, Durand S, Mange D, Sanchez E, Stauffer A, Tempesti G (1996) Embryonics: The Births of Synthetic Life. In: Sanchez, E, Tomassini M (eds) Towards Evolvable Hardware. Springer Verlag, Heidelberg, pp 166–195 **23**
29. Marcu T, Mirea L, Frank P M (1999) Development of dynamic neural networks with application to observer-based fault detection and isolation. *J of Applied Mathematics and Computer Science* 3 (9): 547–570 **30**
30. Mc Taggart J (2001) Intelligent Tutoring Systems and Education for the Future. In: 512X Literature Review April 30, pp 2. <http://www.drake.edu/mathcs/mctaggart/C1512X/LitReview.pdf> **38, 40**
31. Mihaila D, Fagarasan F, Negoita M Gh (1996) Architectural Implications of Genetic Algorithms Complexity in Evolvable Hardware Implementation. In: Proceedings of the European Congress EUFIT'96. Vol. 1. September, Aachen, Germany, pp 400–404 **25**
32. Nauck D, Kruse R (1998) Nefclass – a soft computing tool to build readable fuzzy classifiers. *J BT Technol* 16 (3): 89–103 **34**
33. Negoita M Gh (1994) Genetic Algorithms in Soft-Computing Environment. Theory and Applications. A Tutorial Presentation, Gerhardt Mercator University, Duisburg, Germany, October
34. Negoita M Gh, Fagarasan F, Agapie A (1994a) Soft-Computing: Fusion Examples of Intelligent Techniques. In: Proceedings of International Conference OPTIM'94, Transilvania University of Brasov, Romania, May, pp 335–340
35. Negoita M Gh, Agapie A, Fagarasan F (1994b) The Fusion of Genetic Algorithms and Fuzzy Logic. Application in Expert Systems and Intelligent Control. In : Proceedings of IEEE/Nagoya University WWW Conference on Fuzzy Logic and Neural Net works/Genetic Algorithms, August, Nagoya, Japan, pp 130–133
36. Negoita M Gh, Fagarasan F, Agapie A (1994c) Application of Genetic Algorithms in Solving Fuzzy Relational Equations. In: Proceedings of EUFIT'94, Aachen, Germany, September, pp 1126–1129
37. Negoita M Gh (1995) Evolutionary Computation in the Soft Computing Framework. In: Zimmermann H-Z, Negoita M Gh, Dascalu D (eds) Real World Applications of Intelligent Technologies. Editura Academiei Romane, Bucharest, Romania, pp 113–139 **22**
38. Negoita M Gh, Giuclea M, Dediu A-H (1995) GA to Optimize Approximate Solutions of Fuzzy Relational Equations for Fuzzy Systems or Controllers. In: Proceedings of ANNES'95, Dunedin, Otago, New Zealand, pp 124–127
39. Negoita M Gh, Mihaila D (1995) Intelligent Techniques Based on Genetic Evolution with Applications to Neural Networks Weights Optimization. In: Proceedings of 14-th International Congress on Cybernetics. Namur, Belgium, August
40. Negoita M Gh (1996) Methods Based on Evolutionary Computation Techniques for Implementation of Evolvable Hardware. In: Proceedings of the International Conference of Technical Informatics – vol. 1- Computer Science and Engineering. Timisoara, Romania, pp 37–45 **22**

41. Negoita M Gh (1997) A Modern Solution for the Technology of Preventing and Alarm Systems: Evolutionary Computation in Evolvable Hardware Implementation. In: Proceedings of The Second Dunav Preving International Conference on Preventing and Alarm Systems. Belgrade, November, pp 201–209 **25**
42. Negoita M Gh (1997) Evolutionary Computation in Evolvable Hardware Implementation, An Advanced Topic Lecture. The University of Auckland, Auckland, New Zealand, December **25**
43. Negoita M Gh, Dediu A-H, Mihaela D (1997) Application of GA with Local Improvement of the Chromosomes for the Design of EHW Architectures. In: Proceedings of EUFIT'97. Aachen, Germany, pp 814–818 **25**
44. Negoita M Gh (2002) Intelligent Multi-Agent Hybrid Systems (IMAHS). A Course for Internal Use of Students Only, Wellington Institute of Technology, Wellington, New Zealand
45. Negoita M Gh (2002a) Evolutionary Computation in Evolvable Hardware Implementation – Implication on Engineering Design and Automation. A Tutorial Presentation at AI'02, Canberra, Australia, December **21**
46. Negoita M Gh, Arotaritei D (2003) A GA with Variable-Length Chromosomes for Optimization Objectives of Fuzzy Recurrent Neural Networks. In: A. Barry (ed) 2003 Genetic and Evolutionary Computation Conference Workshop Program, University of Bath, pp 208–214 **19, 20**
47. Negoita M Gh, Pritchard D (2003a) Testing Intelligent Tutoring Systems by Virtual Students. In: Arif Wani M (ed) Proc Int Conf on Machine-Learning and Applications (ICMLA'03), Los Angeles, USA, pp 98–104
48. Negoita M Gh, Pritchard D (2003b) Some Test Problems Regarding Intelligent Tutoring Systems. In: Palade V, Howlett J R, Jain L (eds) Knowledge-Based Intelligent Information and Engineering Systems. Springer-Verlag, Berlin Heidelberg New York, Part II, pp 98–992 **38**
49. Negoita M Gh, Pritchard D (2004a) Using Virtual Student Model for Testing Intelligent Tutoring Systems. *J Interactive Technology & Smart Education* 1: 3–10
50. Negoita M Gh, Pritchard D (2004b) A “Virtual Student” Leads to the Possibility of Optimizer Agents in an ITS. In: Kantardzic M (ed) Proc. ICMLA'04, Louisville, KY, USA, in press
51. Nasraoui O, Gonzales F, Cardona C, Rojas C, Dasgupta D (2003) A Scalable Artificial Immune System Model for Dynamic Unsupervised Learning. In: Cantu-Paz E, Foster J, Kalaynmo D, Lawrence D D, Rajkumar R, O'Reilly U-M, Beyer H-G, Standish R, Kendall G, Wilson S, Harman M, Wegner J, Dasgupta D, Potter M, Schultz A, Dowsland K, Jonoska N, Miller J (eds) Genetic and Evolutionary Computation – GECCO 2003, Springer Verlag Berlin Heidelberg New York, Part I, pp 219–230
52. Orey M A, Nelson W A (1993) Development Principles for Intelligent Tutoring Systems: Integrating Cognitive Theory into the Development of Computer-based Instruction. *J Educational Technology Research and Development* 41(1): 59–72 **38**
53. Palade V, Bumabaru S, Negoita M (1998) A Method for Compiling Neural Networks into Fuzzy Rules Using Genetic Algorithms and Hierarchical Approach. In: Proc KES Int Conf on Knowledge-Based Intelligent Electronic Systems, Adelaide, Australia, pp 353–358 **9, 27**
54. Palade V, Negoita M Gh, Ariton V (1999) Genetic Algorithms Optimization of Knowledge Extraction from Neural Networks. In: Proceedings of ICONIP'99, November, Perth, Australia, pp 752–758 **27**

55. Palade V, Patton R J, Uppal F J, Quevedo J, Daley S (2002a) *Fault diagnosis of an industrial gas turbine using neuro-fuzzy methods*. In: Proceedings of the 15th IFAC World Congress. Barcelona-Spain, 21–26 July, pp 2477–2482 **33, 34**
56. Uppal F J, Patton R J, Palade V (2002) *Neuro-fuzzy based fault diagnosis applied to an electro-pneumatic valve*. In: Proceedings of the 15th IFAC World Congress, Barcelona-Spain, 21–26 July, pp 2483–2488 **37**
57. Park D, Kandel A, Langholz G (1994) Genetic Based New Fuzzy Reasoning Method with Application to Fuzzy Control. IEEE Trans. on SMC 1 : 39–47 **14**
58. Patton R J, Simani S (1999) Identification and fault diagnosis of a simulated model of an industrial gas turbine. Technical Research Report. The University of Hull-UK, Department of Engineering **32**
59. Patton R J, Lopez-Toribio C J, Uppal F J (1999) Artificial Intelligence Aproaches to fault diagnosis for dynamic systems. J Applied Mathematics and Computer Science 3 (9) : 471–518 **30**
60. Paun Gh, Rozenberg G, Salomaa A (1998) DNA Computing – New Computing Paradigms. Springer-Verlag, Berlin Heidelberg New York **6**
61. Pritchard D, Negoita Gh M (2004) A Fuzzy – GA Hybrid Technique for Optimisation of Teaching Sequences Presented in ITSs. In: B. Reusch (ed) Proc 8-th Fuzzy Days. LNCS, Springer Verlag, Berlin Heidelberg New York, in press **40**
62. Ranawana R, Palade V (2004) A Neural Network Based Multi-Classifier System for Gene Identification in DNA Sequences. J Neural Computing, in press **25, 27**
63. Shann J J, Fu H C (1995) A fuzzy neural network for rule acquiring on fuzzy control systems. J Fuzzy Sets and Systems 71: 345–357 **31**
64. Shimojima K, Fukuda T, Hasegawa I (1995) Self-tuning Fuzzy Modeling with Adaptive Membership Function, Rules, and Hierarchical Structure Based on Genetic Algorithm. J Fuzzy Sets and Systems 71: 294–309 **29**
65. Shimonara K (1994) Evolutionary Systems for Brain Communications – Towards an Artificial Intelligence. In: Brooks A R, Maes P (eds) Artificial Life IV. The MIT Press, pp 4–7 **22, 23**
66. Smyth B (2003) Intelligent Navigation on the Mobile Internet. In: Palade V, Howlett J R, Jain L (eds) Knowledge-Based Intelligent Information and Engineering Systems. Springer-Verlag, Berlin, Heidelberg, New York, Part I, pp 17–19
67. Tachibana K, Furuhashi T (1994) A hierarchical fuzzy modelling method using genetic algorithm for identification of concise submodels. In: Proc of 2nd Int Conference on Knowledge-Based Intelligent Electronic Systems. April, Ade laide, Australia. **35**
68. Teodorescu H-N (1994) Non-Linear Systems, Fuzzy Systems, and Neural Networks. In: Proceedings of 3rd Conference on Fuzzy Logic, Neural Networks and Soft Computing, pp 17–28 **16**
69. Thompson A (1995) Evolving Fault Tolerant Systems CSRP 385. In: Proceedings of 1st IEE/IEEE IC on GA's in Eng Sys GALESIA'95, pp 524–529 **23**
70. Thrun S B (1994) Extracting Symbolic Knowledge from Artificial Neural Networks. Revised version of *Technical Report TR-IAI-93-5*, Institut fuer Informatik III – Universitaet Bonn **27**
71. Torresen J (1997) Evolvable Hardware – The Coming Hardware Design Method? In: Kasabov N (Ed) Neuro-fuzzy Tools and Techniques, Springer Verlag **21**
72. Williams R J, Zipser D (1989) Experimental Analysis of the Real-Time Recurrent Learning Algorithm. J Connection Science 1: 87–111 **17, 18, 20**

73. Yager R R, Filev D P, Sadeghi T (1994) Analysis of Flexible Structured Fuzzy Logic Controllers. *IEEE Trans on SMC* 7:1035–1043 **14**
74. Yang X, Furuhashi T, Obata K, Uchikawa Y (1995) Constructing a High Performance Signature Verification System Using A GA Method. In: Proceedings of ANNES'95, Dunedin, Otago, New Zealand, pp 170–173 **25**
75. Zadeh L A (2003) From Search Engines to Question-Answering Systems – The Need For New Tools. URL: <http://www-bisc.cs.berkeley.edu> Computer Science Division Department of EECS UC Berkeley **4**
76. Zadeh L A, Nikravesh M (2002) Perception-Based Intelligent Decision Systems. ONR Summer 2002 Program Review, URL: <http://www-bisc.cs.berkeley.edu> Computer Science Division Department of EECS UC Berkeley **4**
77. Zimmermann H-J (1991) Fuzzy Set Theory and Its Applications (Second Revised Edition). Kluwer Academic Publishers, Boston, Dodrecht, London.
78. Zimmermann H-J, Negoita Gh M, Dascalu D (Eds) (1996) Real World Application of Intelligent Technologies. Editura Academiei Romane, Bucharest **12**

Basics of Machine Learning by Support Vector Machines

V. Kecman

Abstract. Here, we talk about the (machine) learning from empirical data (i.e., examples, samples, measurements, records, patterns or observations) by applying support vector machines (SVMs) a.k.a. kernel machines¹. The basic aim of this chapter is to give, as far as possible, a condensed (but systematic) presentation of a novel learning paradigm embodied in SVMs. Our focus will be on the constructive learning algorithms for both the classification (pattern recognition) and regression (function approximation) problems. Consequently, we will not go into all the subtleties and details of the statistical learning theory (SLT) and structural risk minimization (SRM) which are theoretical foundations for the learning algorithms presented below. This seems more appropriate for the application oriented readers. The theoretically minded and interested reader may find an extensive presentation of both the SLT and SRM in [4, 15, 23, 31, 33]. Instead of diving into a theory, a quadratic programming based learning leading to parsimonious SVMs will be presented in a gentle way - starting with linear separable problems, through the classification tasks having overlapped classes but still a linear separation boundary, beyond the linearity assumptions to the nonlinear separation boundary, and finally to the linear and nonlinear regression problems. Here, the adjective “parsimonious” denotes a SVM with a small number of support vectors (“hidden layer neurons”). The scarcity of the model results from a sophisticated, QP based, learning that matches the model capacity to the data complexity ensuring a good generalization, i.e., a good performance of SVM on the future, previously, during the training unseen, data.

Same as the neural networks (or similarly to them), SVMs possess the well-known ability of being universal approximators of any multivariate function to any desired degree of accuracy. Consequently, they are of particular interest for modeling the unknown, or partially known, highly nonlinear, complex systems, plants or processes. Also, at the very beginning, and just to be sure what the whole booklet is about, we should state clearly when there is no need for an application of SVMs’ model-building techniques. In short, whenever there exists an analytical closed-form model

¹The chapter is an extended tutorial presented at KES 2004 in Wellington, NZ, and it strictly follows the School of Engineering of The University of Auckland Report 616. The right to use the material from this report is received with gratitude.

(or it is possible to devise one) there is no need to resort to learning from empirical data by SVMs (or by any other type of a learning machine).

1 Basics of Learning From Data

SVMs have been developed in the reverse order to the development of neural networks (NNs). SVMs evolved from the sound theory to the implementation and experiments, while the NNs followed more heuristic path, from applications and extensive experimentation to the theory. It is interesting to note that the very strong theoretical background of SVMs did not make them widely appreciated at the beginning. The publication of the first papers by Vapnik, Chervonenkis [28] and co-workers went largely unnoticed till 1992. This was due to a widespread belief in the statistical and/or machine learning community that, despite being theoretically appealing, SVMs are neither suitable nor relevant for practical applications. They were taken seriously only when excellent results on practical learning benchmarks were achieved (in numeral recognition, computer vision and text categorization). Today, SVMs show better results than (or comparable outcomes to) NNs and other statistical models, on the most popular benchmark problems.

The learning problem setting for SVMs is as follows: there is some unknown and nonlinear dependency (mapping, function) $y = f(\mathbf{x})$ between some high-dimensional input vector \mathbf{x} and scalar output y (or the vector output \mathbf{y} as in the case of multiclass SVMs). There is no information about the underlying joint probability functions here. Thus, one must perform a *distribution-free learning*. The only information available is a training data set $D = \{(\mathbf{x}_i, y_i) \in X \times Y\}, i = 1, l$, where l stands for the number of the training data pairs and is therefore equal to the size of the training data set D . Often, y_i is denoted as d_i , where d stands for a desired (target) value. Hence, SVMs belong to the supervised learning techniques.

Note that this problem is similar to the classic statistical inference. However, there are several very important differences between the approaches and assumptions in training SVMs and the ones in classic statistics and/or NNs modeling. Classic statistical inference is based on the following three fundamental assumptions:

1. Data can be modeled by a set of linear in parameter functions; this is a foundation of a parametric paradigm in learning from experimental data.
2. In the most of real-life problems, a stochastic component of data is the normal probability distribution law, that is, the underlying joint probability distribution is a Gaussian distribution.
3. Because of the second assumption, the induction paradigm for parameter estimation is the maximum likelihood method, which is reduced to the minimization of the sum-of-errors-squares cost function in most engineering applications.

All three assumptions on which the classic statistical paradigm relied turned out to be inappropriate for many contemporary real-life problems [33] because of the following facts:

1. Modern problems are high-dimensional, and if the underlying mapping is not very smooth the linear paradigm needs an exponentially increasing number of terms with an increasing dimensionality of the input space X (an increasing number of independent variables). This is known as “the curse of dimensionality”.
2. The underlying real-life data generation laws may typically be very far from the normal distribution and a model-builder must consider this difference in order to construct an effective learning algorithm.
3. From the first two points it follows that the maximum likelihood estimator (and consequently the sum-of-error-squares cost function) should be replaced by a new induction paradigm that is uniformly better, in order to model non-Gaussian distributions.

In addition to the three basic objectives above, the novel SVMs’ problem setting and inductive principle have been developed for standard contemporary data sets which are typically high-dimensional and sparse (meaning, the data sets contain small number of the training data pairs).

SVMs are the so-called “nonparametric” models. “Nonparametric” does not mean that the SVMs’ models do not have parameters at all. On the contrary, their “learning” (selection, identification, estimation, training or tuning) is the crucial issue here. However, unlike in classic statistical inference, the parameters are not predefined and their number depends on the training data used. In other words, parameters that define the capacity of the model are data-driven in such a way as to match the model capacity to data complexity. This is a basic paradigm of the structural risk minimization (SRM) introduced by Vapnik and Chervonenkis and their coworkers that led to the new learning algorithm. Namely, there are two basic constructive approaches possible in designing a model that will have a good generalization property [31, 33]:

1. choose an appropriate structure of the model (order of polynomials, number of HL neurons, number of rules in the fuzzy logic model) and, keeping the estimation error (a.k.a. confidence interval, a.k.a. variance of the model) fixed in this way, minimize the training error (i.e., empirical risk), or
2. keep the value of the training error (a.k.a. an approximation error, a.k.a. an empirical risk) fixed (equal to zero or equal to some acceptable level), and minimize the confidence interval.

Classic NNs implement the first approach (or some of its sophisticated variants) and SVMs implement the second strategy. In both cases the resulting model should resolve the trade-off between under-fitting and over-fitting the training data. The final model structure (its order) should ideally *match the learning machines capacity with training data complexity*. This important difference in two learning approaches comes from the minimization of different

Table 1. Basic Models and Their Error (Risk) Functionals

| Multilayer Perceptron (NN) | Regularization Network (Radial Basis Functions Network) | Support Vector Machine |
|--|---|--|
| $R = \sum_{i=1}^l \underbrace{(d_i - f(\mathbf{x}_i, \mathbf{w}))^2}_{\text{Closeness to data}}$ | $R = \sum_{i=1}^l \underbrace{(d_i - f(\mathbf{x}_i, \mathbf{w}))^2}_{\text{Closeness to data}} + \lambda \underbrace{\ \mathbf{P}f\ ^2}_{\text{Smoothness}}$ | $R = \sum_{i=1}^l \underbrace{L\varepsilon}_{\text{Closeness to data}} + \underbrace{\Omega(l, h)}_{\text{capacity of a machine}}$ |

Closeness to data = training error, a.k.a. empirical risk

cost (error, loss) functionals. Table 1 tabulates the basic risk functionals applied in developing the three contemporary statistical models.

d_i stands for desired values, \mathbf{w} is the weight vector subject to training, λ is a regularization parameter, \mathbf{P} is a smoothness operator, L_ε is a SVMs' loss function, h is a VC dimension and Ω is a function bounding the capacity of the learning machine. In classification problems L_ε is typically 0–1 loss function, and in regression problems L_ε is the so-called Vapnik's ε -insensitivity loss (error) function

$$L_\varepsilon = |y - f(\mathbf{x}, \mathbf{w})|_\varepsilon = \begin{cases} 0, & \text{if } |y - f(\mathbf{x}, \mathbf{w})| \leq \varepsilon \\ |y - f(\mathbf{x}, \mathbf{w})| - \varepsilon, & \text{otherwise.} \end{cases} \quad (1)$$

where ε is a radius of a tube within which the regression function must lie, after the successful learning. (Note that for $\varepsilon = 0$, the interpolation of training data will be performed). It is interesting to note that [11] has shown that under some constraints the SV machine can also be derived from the framework of regularization theory rather than SLT and SRM. Thus, *unlike the classic adaptation algorithms that work in the L_2 norm*. *SV machines represent novel learning techniques which perform SRM*. In this way, the SV machine creates a model with minimized VC dimension and when the VC dimension of the model is low, the expected probability of error is low as well. This means good performance on previously unseen data, i.e. a good generalization. This property is of particular interest because the model that generalizes well is a good model and not the model that performs well on training data pairs. Too good a performance on training data is also known as an extremely undesirable overfitting.

As it will be shown below, in the “simplest” pattern recognition tasks, support vector machines use a linear separating hyperplane to create a *classifier with a maximal margin*. In order to do that, the learning problem for the SV machine will be cast as a *constrained nonlinear optimization* problem. In this setting the cost function will be quadratic and the constraints linear (i.e., one will have to solve a classic *quadratic programming problem*).

In cases when given classes cannot be linearly separated in the original input space, the SV machine first (non-linearly) transforms the original input space into a higher dimensional *feature space*. This transformation can be achieved by using various nonlinear mappings; polynomial, sigmoidal as in multilayer perceptrons, RBF mappings having as the basis functions radially

symmetric functions such as Gaussians, or multiquadratics or different spline functions. After this nonlinear transformation step, the task of a SV machine in finding the linear optimal separating hyperplane in this feature space is “relatively trivial”. Namely, the optimization problem to solve in a feature space will be of the same kind as the calculation of a maximal margin separating hyperplane in original input space for linearly separable classes. How, after the specific nonlinear transformation, nonlinearly separable problems in input space can become linearly separable problems in a feature space will be shown later.

In a probabilistic setting, there are three basic components in all learning from data tasks: a *generator* of random inputs \mathbf{x} , a *system* whose *training responses* y (i.e., d) are used for training the learning machine, and a *learning machine* which, by using inputs \mathbf{x}_i and system’s responses y_i , should learn (estimate, model) the unknown dependency between these two sets of variables defined by the weight vector \mathbf{w} (Fig. 1).

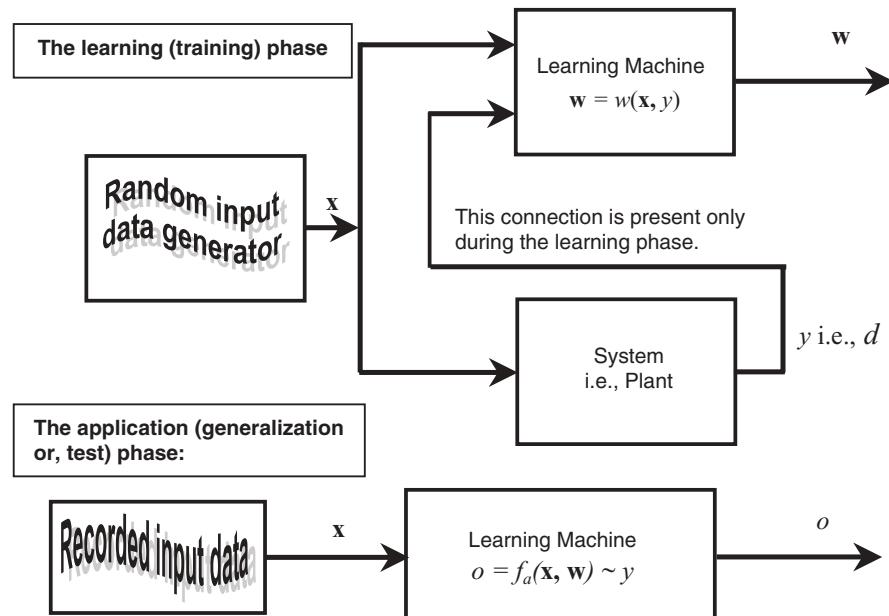


Fig. 1. A model of a learning machine (*top*) $w = w(x, y)$ that during *the training phase* (by observing inputs x_i to, and outputs y_i from, the system) estimates (learns, adjusts, trains, tunes) its parameters (weights) w , and in this way learns mapping $y = f(\mathbf{x}, \mathbf{w})$ performed by the system. The use of $f_a(\mathbf{x}, \mathbf{w}) \sim y$ denotes that we will rarely try to *interpolate* training data pairs. We would rather seek an *approximating function* that can generalize well. After the training, at the *generalization* or *test phase*, the output from a machine $o = f_a(\mathbf{x}, \mathbf{w})$ is expected to be “a good” estimate of a system’s true response y

The figure shows the most common learning setting that some readers may have already seen in various other fields - notably in statistics, NNs, control system identification and/or in signal processing. During the (successful) training phase a learning machine should be able to find the relationship between an input space X and an output space Y , by using data D in regression tasks (or to find a function that separates data within the input space, in classification ones). The result of a learning process is an “approximating function” $f_a(\mathbf{x}, \mathbf{w})$, which in statistical literature is also known as, a *hypothesis* $f_a(\mathbf{x}, \mathbf{w})$. This function approximates the underlying (or true) dependency between the input and output in the case of regression, and the decision boundary, i.e., separation function, in a classification. The chosen hypothesis $f_a(\mathbf{x}, \mathbf{w})$ belongs to a *hypothesis space of functions* $H(f_a \in H)$, and it is a function that minimizes some *risk functional* $R(\mathbf{w})$.

It may be practical to remind the reader that under the general name “approximating function” we understand any mathematical structure that maps inputs \mathbf{x} into outputs y . Hence, an “approximating function” may be: a multilayer perceptron NN, RBF network, SV machine, fuzzy model, Fourier truncated series or polynomial approximating function. Here we discuss SVMs. A set of parameters \mathbf{w} is the very subject of learning and generally these parameters are called *weights*. These parameters may have different geometrical and/or physical meanings. Depending upon the hypothesis space of functions H we are working with the parameters \mathbf{w} are usually:

- the hidden and the output layer weights in multilayer perceptrons,
- the rules and the parameters (for the positions and shapes) of fuzzy subsets,
- the coefficients of a polynomial or Fourier series,
- the centers and (co)variances of Gaussian basis functions as well as the output layer weights of this RBF network,
- the support vector weights in SVMs.

There is another important class of functions in learning from examples tasks. A learning machine tries to capture an unknown *target function* $f_0(\mathbf{x})$ that is believed to belong to some target space T , or to a class T , that is also called a *concept class*. Note that we rarely know the target space T and that our learning machine generally does not belong to the same class of functions as an unknown target function $f_0(\mathbf{x})$. Typical examples of target spaces are continuous functions with s continuous derivatives in n variables; Sobolev spaces (comprising square integrable functions in n variables with s square integrable derivatives), band-limited functions, functions with integrable Fourier transforms, Boolean functions, etc. In the following, we will assume that the target space T is a space of differentiable functions. The basic problem we are facing stems from the fact that we know very little about the possible underlying function between the input and the output variables. All we have at our disposal is a training data set of labeled examples drawn by independently sampling a $(X \times Y)$ space according to some unknown probability distribution.

The learning-from-data problem is ill-posed. (This will be shown on Figs. 2 and 3 for a regression and classification examples respectively). The basic source of the ill-posedness of the problem is due to the infinite number of possible solutions to the learning problem. At this point, just for the sake of illustration, it is useful to remember that all functions that interpolate data points will result in a zero value for training error (empirical risk) as shown (in the case of regression) in Fig. 2. The figure shows a simple example of three-out-of-infinitely-many different interpolating functions of training data pairs sampled from a noiseless function $y = \sin(x)$.

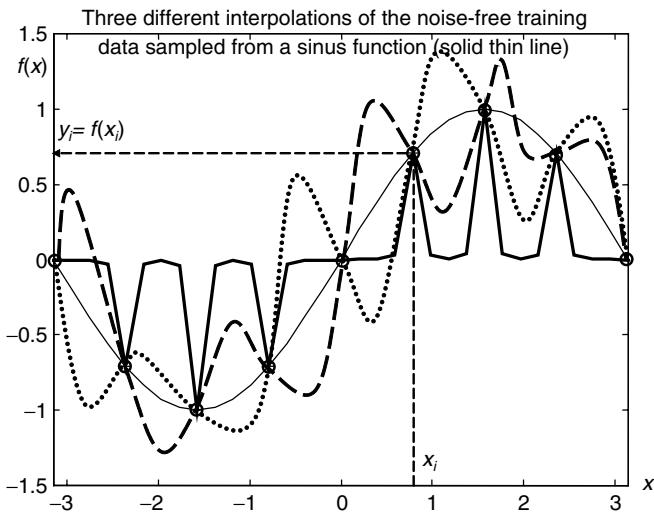


Fig. 2. Three-out-of-infinitely-many interpolating functions resulting in a training error equal to 0. However, a *thick solid, dashed and dotted lines* are bad models of a true function $y = \sin(x)$ (*thin dashed line*)

In Fig. 2, each interpolant results in a training error equal to zero, but at the same time, each one is a very bad model of the true underlying dependency between x and y , because all three functions perform very poorly outside the training inputs. In other words, none of these three particular interpolants can generalize well. However, not only interpolating functions can mislead. There are many other approximating functions (learning machines) that will minimize the empirical risk (approximation or training error) but not necessarily the generalization error (true, expected or guaranteed risk). This follows from the fact that a learning machine is trained by using some particular sample of the true underlying function and consequently it always produces biased approximating functions. These approximants depend necessarily on the specific training data pairs (i.e., the training sample) used.

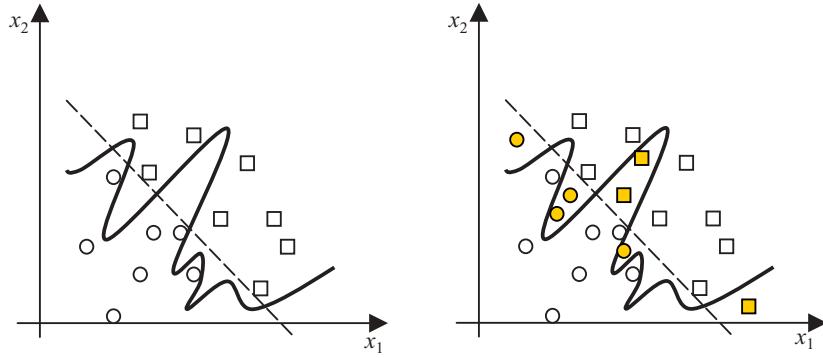


Fig. 3. Overfitting in the case of linearly separable classification problem. *Left:* The perfect classification of the training data (*empty circles and squares*) by both low order linear model (*dashed line*) and high order nonlinear one (*solid wiggly curve*). *Right:* Wrong classification of all the test data shown (*filled circles and squares*) by a high capacity model, but correct one by the simple linear separation boundary

Figure 3 shows an extremely simple classification example where the classes (represented by the empty training circles and squares) are linearly separable. However, in addition to a linear separation (dashed line) the learning was also performed by using a model of a high capacity (say, the one with Gaussian basis functions, or the one created by a high order polynomial, over the 2-dimensional input space) that produced a perfect separation boundary (empirical risk equals zero) too. However, such a model is overfitting the data and it will definitely perform very badly on, during the training unseen, test examples. Filled circles and squares in the right hand graph are all wrongly classified by the nonlinear model. Note that a simple linear separation boundary correctly classifies both the training and the test data.

A solution to this problem proposed in the framework of the SLT is restricting the hypothesis space H of approximating functions to a set smaller than that of the target function T while simultaneously controlling the flexibility (complexity) of these approximating functions. This is ensured by an introduction of a novel induction principle of the SRM and its algorithmic realization through the SV machine. The Structural Risk Minimization principle [28] tries to minimize an expected risk (the cost function) R comprising two terms as given in Table 1 for the SVMs $R = \Omega(l, h) + \sum_{i=1}^l L_\varepsilon = \Omega(l, h) + R_{\text{emp}}$ and it is based on the fact that for the classification learning problem with a probability of at least $1 - \eta$ the bound

$$R(\mathbf{w}_n) \leq \Omega\left(\frac{h}{l}, \frac{\ln(\eta)}{l}\right) + R_{\text{emp}}(\mathbf{w}_n), \quad (2a)$$

holds. The first term on the right hand side is named a VC confidence (confidence term or confidence interval) that is defined as

$$\Omega\left(\frac{h}{l}, \frac{\ln(\eta)}{l}\right) = \sqrt{\frac{h [\ln(\frac{2l}{h}) + 1] - \ln(\frac{\eta}{4})}{l}}. \quad (2b)$$

The parameter h is called the VC [28, 30] dimension of a set of functions. It describes the capacity of a set of functions implemented in a learning machine. For a binary classification h is the maximal number of points which can be separated (shattered) into two classes in all possible 2^h ways by using the functions of the learning machine.

A SV (learning) machine can be thought of as

- a set of functions implemented in a SVM,
- an induction principle and,
- an algorithmic procedure for implementing the induction principle on the given set of functions.

The notation for risks given above by using $R(\mathbf{w}_n)$ denotes that an expected risk is calculated over a set of functions $f_{an}(\mathbf{x}, \mathbf{w}_n)$ of increasing complexity. Different bounds can also be formulated in terms of other concepts such as *growth function* or *annealed VC entropy*. Bounds also differ for regression tasks. More detail can be found in ([31], as well as in [4]). However, the general characteristics of the dependence of the confidence interval on the number of training data l and on the VC dimension h is similar and given in Fig. 4.

Equations (2a) show that when the number of training data increases, i.e., for $l \rightarrow \infty$ (with other parameters fixed), an expected (true) risk $R(\mathbf{w}_n)$ is very close to empirical risk $R_{\text{emp}}(\mathbf{w}_n)$ because $\Omega \rightarrow 0$. On the other hand,

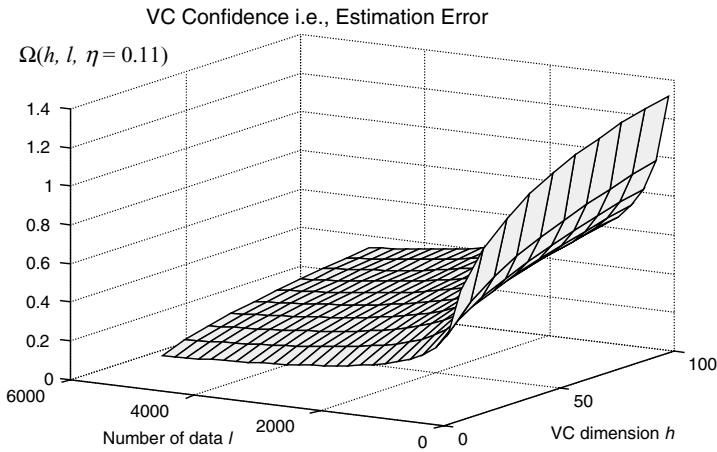


Fig. 4. The dependency of VC confidence $\Omega(h, l, \eta)$ on the number of training data l and the VC dimension h ($h < l$) for a fixed confidence level $1 - \eta = 1 - 0.11 = 0.89$

when the probability $1 - \eta$ (also called a confidence level which should not be confused with the confidence term Ω) approaches 1, the generalization bound grows large, because in the case when $\eta \rightarrow 0$ (meaning that the confidence level $1 - \eta \rightarrow 1$), the value of $\Omega \rightarrow \infty$. This has an obvious intuitive interpretation [4] in that any learning machine (model, estimates) obtained from a finite number of training data cannot have an arbitrarily high confidence level. There is always a trade-off between the accuracy provided by bounds and the degree of confidence (in these bounds). Figure 4 also shows that the VC confidence interval increases with an increase in a VC dimension h for a fixed number of the training data pairs l .

The SRM is a novel inductive principle for learning from finite training data sets. It proved to be very useful when dealing with *small samples*. The basic idea of the SRM is to choose (from a large number of possibly candidate learning machines), a model of the right capacity to describe *the given training data pairs*. As mentioned, this can be done by restricting the hypothesis space H of approximating functions and simultaneously by controlling their flexibility (complexity). Thus, learning machines will be those parameterized models that, by increasing the number of parameters (typically called weights w_i here), form a nested structure in the following sense

$$H_1 \subset H_2 \subset H_3 \subset \dots H_{n-1} \subset H_n \subset \dots \subset H \quad (3)$$

In such a nested set of functions, every function always contains a previous, less complex, function. Typically, H_n may be: a set of polynomials in one variable of degree n ; fuzzy logic model having n rules; multilayer perceptrons, or RBF network having n HL neurons, SVM structured over n support vectors. The goal of learning is one of a *subset selection* that matches training data complexity with approximating model capacity. In other words, a learning algorithm chooses an optimal polynomial degree or, an optimal number of HL neurons or, an optimal number of FL model rules, for a polynomial model or NN or FL model respectively. For learning machines linear in parameters, this complexity (expressed by the VC dimension) is given by the number of weights, i.e., by the number of “free parameters”. For approximating models nonlinear in parameters, the calculation of the VC dimension is often not an easy task. Nevertheless, even for these networks, by using simulation experiments, one can find a model of appropriate complexity.

2 Support Vector Machines in Classification and Regression

Below, we focus on the algorithm for implementing the SRM induction principle on the given set of functions. It implements the strategy mentioned previously – it keeps the training error fixed and minimizes the confidence interval. We first consider a “simple” example of linear decision rules (i.e., the separating functions will be hyperplanes) for binary classification (dichotomization)

of linearly separable data. In such a problem, we are able to perfectly classify data pairs, meaning that an empirical risk can be set to zero. It is the easiest classification problem and yet an excellent introduction of all relevant and important ideas underlying the SLT, SRM and SVM.

Our presentation will gradually increase in complexity. It will begin with a *Linear Maximal Margin Classifier for Linearly Separable Data* where there is no sample overlapping. Afterwards, we will allow some degree of overlapping of training data pairs. However, we will still try to separate classes by using linear hyperplanes. This will lead to the *Linear Soft Margin Classifier for Overlapping Classes*. In problems when linear decision hyperplanes are no longer feasible, the mapping of an input space into the so-called feature space (that “corresponds” to the HL in NN models) will take place resulting in the *Nonlinear Classifier*. Finally, in the subsection on *Regression by SV Machines* we introduce same approaches and techniques for solving regression (i.e., function approximation) problems.

2.1 Linear Maximal Margin Classifier for Linearly Separable Data

Consider the problem of binary classification or dichotomization. Training data are given as

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n), \mathbf{x} \in \Re^n, y \in \{+1, -1\} \quad (4)$$

For reasons of visualization only, we will consider the case of a two-dimensional input space, i.e., $\mathbf{x} \in \Re^2$. Data are linearly separable and there are many different hyperplanes that can perform separation (Fig. 5). (Actually, for $\mathbf{x} \in \Re^2$, the separation is performed by “planes” $w_1x_1 + w_2x_2 + b = o$. In other words, the decision boundary, i.e., the separation line in input space is defined by the equation $w_1x_1 + w_2x_2 + b = 0$.) How to find “the best” one? The difficult part is that all we have at our disposal are sparse training data. Thus, we want to find the optimal separating function without knowing the underlying probability distribution $P(\mathbf{x}, y)$. There are many functions that can solve given pattern recognition (or functional approximation) tasks. In such a problem setting, the SLT (developed in the early 1960s by Vapnik and Chervonenkis) shows that it is crucial to restrict the class of functions implemented by a learning machine to one with a complexity that is suitable for the amount of available training data.

In the case of a classification of linearly separable data, this idea is transformed into the following approach – among all the hyperplanes that minimize the training error (i.e., empirical risk) find the one with the largest margin. This is an intuitively acceptable approach. Just by looking at Fig. 5 we will find that the dashed separation line shown in the *right graph* seems to promise *probably* good classification while facing previously unseen data (meaning, in the generalization, i.e. test, phase). Or, at least, it seems to probably be better

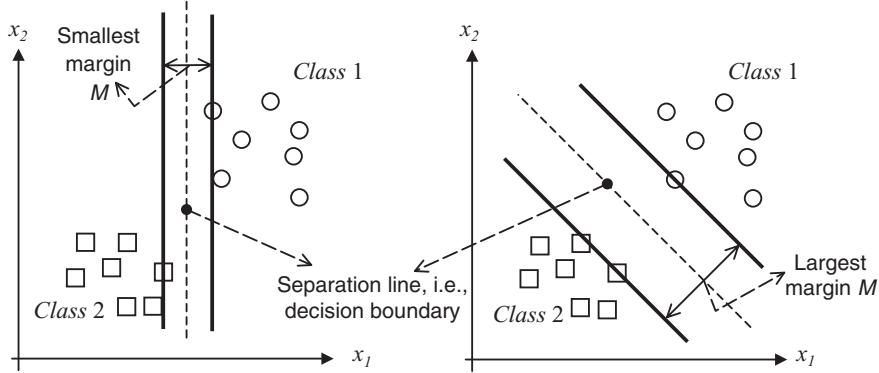


Fig. 5. Two-out-of-many separating lines: a good one with a large margin (*right*) and a less acceptable separating line with a small margin, (*left*)

in generalization than the dashed decision boundary having smaller margin shown in the left graph. This can also be expressed as that a classifier with smaller margin will have higher expected risk.

By using given training examples, during the learning stage, our machine finds parameters $\mathbf{w} = [w_1 w_2 \dots w_n]^T$ and b of a discriminant or decision function $d(\mathbf{x}, \mathbf{w}, b)$ given as

$$d(\mathbf{x}, \mathbf{w}, b) = \mathbf{w}^T \mathbf{x} + b = \sum_{i=1}^n w_i x_i + b, \quad (5)$$

where $\mathbf{x}, \mathbf{w} \in \Re^n$, and the scalar b is called *a bias*. (Note that the dashed separation lines in Fig. 5 represent the line that follows from $d(\mathbf{x}, \mathbf{w}, b) = 0$). After the successful training stage, by using the weights obtained, the learning machine, given previously unseen pattern \mathbf{x}_p , produces output o according to an *indicator function* given as

$$i_F = o = \text{sign}(d(\mathbf{x}_p, \mathbf{w}, b)), \quad (6)$$

where o is the standard notation for the *output* from the learning machine. In other words, *the decision rule is*:

- if $d(\mathbf{x}_p, \mathbf{w}, b) > o$, the pattern \mathbf{x}_p belongs to a class 1 (i.e., $o = y_1 = +1$),
- and
- if $d(\mathbf{x}_p, \mathbf{w}, b) < o$ the pattern \mathbf{x}_p belongs to a class 2 (i.e., $o = y_2 = -1$).

The *indicator function* i_F given by (6) is a step-wise (i.e., a stairs-wise) function (see Figs. 6 and 7). At the same time, the decision (or discriminant) function $d(\mathbf{x}, \mathbf{w}, b)$ is a hyperplane. Note also that both a decision hyperplane d and the indicator function i_F live in an $n + 1$ -dimensional space

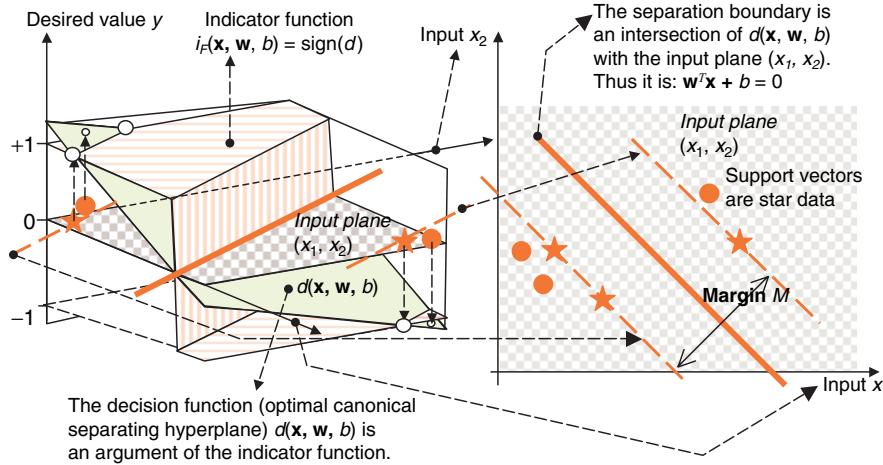


Fig. 6. The definition of a decision (discriminant) *function* or hyperplane $d(\mathbf{x}, \mathbf{w}, b)$, a decision (separating) *boundary* $d(\mathbf{x}, \mathbf{w}, b) = 0$ and an indicator function $i_F = \text{sign}(d(\mathbf{x}, \mathbf{w}, b))$ whose value represents a learning, or SV, machine's output o

or they lie “over” a training pattern’s n -dimensional input space. There is one more mathematical object in classification problems called a *separation boundary* that lives in the same n -dimensional space of input vectors \mathbf{x} . Separation boundary separates vectors \mathbf{x} into two classes. Here, in cases of linearly separable data, the boundary is also a (separating) hyperplane but of a lower order than $d(\mathbf{x}, \mathbf{w}, b)$. The decision (separation) *boundary* is an intersection of a decision *function* $d(\mathbf{x}, \mathbf{w}, b)$ and a space of input features. It is given by

$$d(\mathbf{x}, \mathbf{w}, b) = 0 . \quad (7)$$

All these functions and relationships can be followed, for two-dimensional inputs \mathbf{x} , in Fig. 6. In this particular case, the decision boundary i.e., separating (hyper)plane is actually a separating line in a $x_1 - x_2$ plane and, a decision function $d(\mathbf{x}, \mathbf{w}, b)$ is a plane over the 2-dimensional space of features, i.e., over a $x_1 - x_2$ plane.

In the case of 1-dimensional training patterns x (i.e., for 1-dimensional inputs x to the learning machine), decision function $d(x, \mathbf{w}, b)$ is a straight line in an $x - y$ plane. An intersection of this line with an x -axis defines a point that is a separation boundary between two classes. This can be followed in Fig. 7. Before attempting to find an optimal separating hyperplane having the largest margin, we introduce the concept of the *canonical hyperplane*. We depict this concept with the help of the 1-dimensional example shown in Fig. 7. Not quite incidentally, the decision plane $d(\mathbf{x}, \mathbf{w}, b)$ shown in Fig. 6 is also a *canonical* plane. Namely, the values of d and of i_F are the same and both are equal to $|1|$ for the support vectors depicted by stars. At the same time, for all other training patterns $|d| > |i_F|$. In order to present a

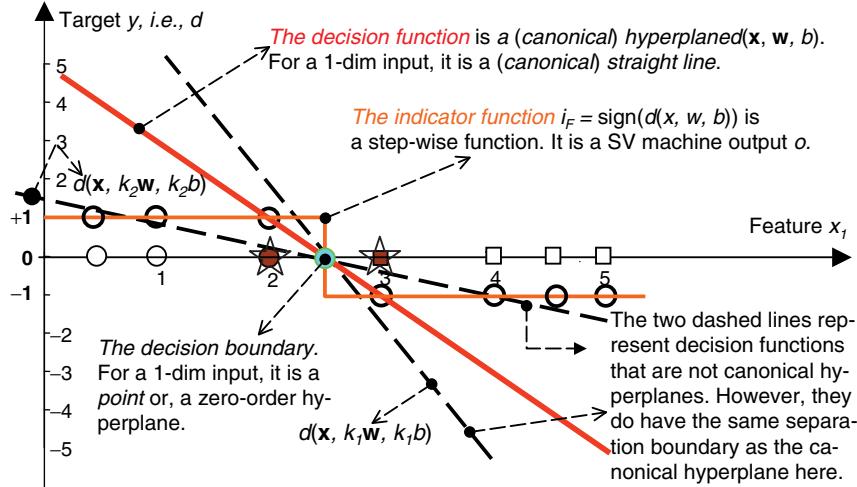


Fig. 7. SV classification for 1-dimensional inputs by the linear decision function. Graphical presentation of a *canonical hyperplane*. For 1-dimensional inputs, it is actually a canonical straight line (depicted as a *thick straight solid line*) that passes through points $(+2, +1)$ and $(+3, -1)$ defined as the support vectors (*stars*). The two *dashed lines* are the two other decision hyperplanes (i.e., *straight lines*). The training input patterns $\{x_1 = 0.5, x_2 = 1, x_3 = 2\} \in \text{Class 1}$ have a desired or target value (label) $y_1 = +1$. The inputs $\{x_4 = 3, x_5 = 4, x_6 = 4.5, x_7 = 5\} \in \text{Class 2}$ have the label $y_2 = -1$

notion of this new concept of the canonical plane, first note that there are many hyperplanes that can correctly separate data. In Fig. 7 three different decision functions $d(\mathbf{x}, \mathbf{w}, b)$ are shown. There are infinitely many more. In fact, given $d(\mathbf{x}, \mathbf{w}, b)$, all functions $d(\mathbf{x}, k\mathbf{w}, kb)$, where k is a positive scalar, are correct decision functions too. Because parameters (\mathbf{w}, b) describe the same separation hyperplane as parameters $(k\mathbf{w}, kb)$ there is a need to introduce the notion of a *canonical hyperplane*:

A hyperplane is in the canonical form with respect to training data $\mathbf{x} \in X$ if

$$\min_{x_i \in X} |\mathbf{w}^T \mathbf{x}_i + b| = 1. \quad (8)$$

The solid line $d(\mathbf{x}, \mathbf{w}, b) = -2x + 5$ in Fig. 7 fulfills (8) because its minimal absolute value for the given six training patterns belonging to two classes is 1. It achieves this value for two patterns, chosen as support vectors, namely for $x_3 = 2$, and $x_4 = 3$. For all other patterns, $|d| > 1$.

Note an interesting detail regarding the notion of a canonical hyperplane that is easily checked. There are many different hyperplanes (planes and straight lines for 2-D and 1-D problems in Figs. 6 and 7 respectively) that have the same separation boundary (solid line and a dot in Figs. 6 (right) and

[7](#) respectively). At the same time there are far fewer hyperplanes that can be defined as canonical ones fulfilling [\(8\)](#). In Fig. [7](#), i.e., for a 1-dimensional input vector x , the canonical hyperplane is unique. This is not the case for training patterns of higher dimension. Depending upon the configuration of class” elements, various canonical hyperplanes are possible.

Therefore, there is a need to define an *optimal* canonical hyperplane (OCSH) as a canonical hyperplane having a *maximal margin*. This search for a separating, maximal margin, canonical hyperplane is the ultimate learning goal in statistical learning theory underlying SV machines. Carefully note the adjectives used in the previous sentence. The hyperplane obtained from a limited training data must have a *maximal margin* because it will *probably* better classify new data. It must be in *canonical* form because this will ease the quest for significant patterns, here called support vectors. The canonical form of the hyperplane will also simplify the calculations. Finally, the resulting hyperplane must ultimately *separate* training patterns.

We avoid the derivation of an expression for the calculation of a distance (margin M) between the closest members from two classes for its simplicity. The curious reader can derive the expression for M as given below, or it can look in [\[15\]](#) or other books. The margin M can be derived by both the geometric and algebraic argument and is given as

$$M = \frac{2}{\|\mathbf{w}\|}. \quad (9)$$

This important result will have a great consequence for the constructive (i.e., learning) algorithm in a design of a maximal margin classifier. It will lead to solving a quadratic programming (QP) problem which will be shown shortly. Hence, the “good old” gradient learning in NNs will be replaced by solution of the QP problem here. This is the next important difference between the NNs and SVMs and follows from the implementation of SRM in designing SVMs, instead of a minimization of the sum of error squares, which is a standard cost function for NNs.

Equation [\(9\)](#) is a very interesting result showing that minimization of a norm of a hyperplane normal weight vector $\|\mathbf{w}\| = \sqrt{(\mathbf{w}^T \mathbf{w})} = \sqrt{w_1^2 + w_2^2 + \dots + w_n^2}$ leads to a maximization of a margin M . Because a minimization of \sqrt{f} is equivalent to the minimization of f , the minimization of a norm $\|\mathbf{w}\|$ equals a minimization of $\mathbf{w}^T \mathbf{w} = \sum_{i=1}^n w_i^2 = w_1^2 + w_2^2 + \dots + w_n^2$, and this leads to a maximization of a margin M . Hence, the learning problem is

$$\text{minimize } \frac{1}{2} \mathbf{w}^T \mathbf{w}, \quad (10a)$$

subject to constraints introduced and given in [\(10b\)](#) below.

(A multiplication of $\mathbf{w}^T \mathbf{w}$ by 0.5 is for numerical convenience only, and it doesn’t change the solution). Note that in the case of linearly separable classes empirical error equals zero ($R_{\text{emp}} = 0$ in [\(2a\)](#)) and minimization of $\mathbf{w}^T \mathbf{w}$ corresponds to a minimization of a confidence term Ω . The OCSH, i.e.,

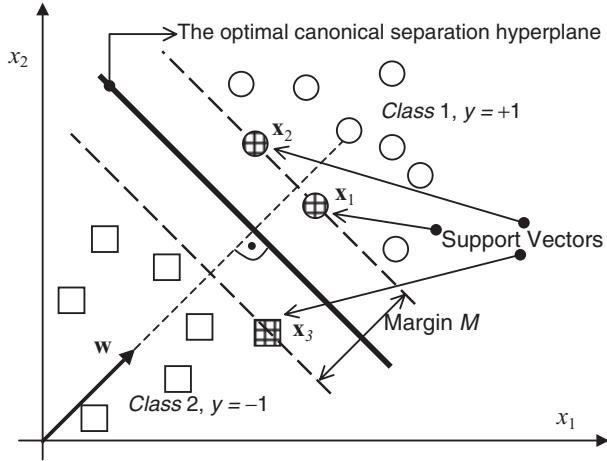


Fig. 8. The optimal canonical separating hyperplane with the largest margin intersects halfway between the two classes. The points closest to it (satisfying $y_j |\mathbf{w}^T \mathbf{x}_j + b| = 1, j = 1, N_{SV}$) are *support vectors* and the OCSH satisfies $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 i = 1, l$ (where l denotes the number of training data and N_{SV} stands for the number of SV). Three support vectors (\mathbf{x}_1 and \mathbf{x}_2 from class 1, and \mathbf{x}_3 from class 2) are the textured training data

a separating hyperplane with the largest margin defined by $M = 2/\|\mathbf{w}\|$, specifies *support vectors*, i.e., training data points closest to it, which satisfy $y_j [\mathbf{w}^T \mathbf{x}_j + b] \equiv 1, j = 1, N_{SV}$. For all the other (non-SVs data points) the OCSH satisfies inequalities $y_i [\mathbf{w}^T \mathbf{x}_i + b] > 1$. In other words, for all the data, OCSH should satisfy the following constraints

$$y_i [\mathbf{w}^T \mathbf{x}_i + b] \geq 1, \quad i = 1, l \quad (10b)$$

where l denotes a number of training data points, and N_{SV} stands for a number of SVs. The last equation can be easily checked visually in Figs. 6 and 7 for 2-dimensional and 1-dimensional input vectors \mathbf{x} respectively. Thus, in order to find the OCSH having a maximal margin, a learning machine should minimize $\|\mathbf{w}\|^2$ subject to the inequality constraints (10b). This is a *classic quadratic optimization problem with inequality constraints*. Such an optimization problem is solved by the *saddle point* of the Lagrange functional (Lagrangian)²

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^l \alpha_i \{y_i [\mathbf{w}^T \mathbf{x}_i + b] - 1\}, \quad (11)$$

²In forming the Lagrangian, for constraints of the form $f_i > 0$, the inequality constraints equations are multiplied by *nonnegative* Lagrange multipliers (i.e., $\alpha_i \geq 0$) and *subtracted* from the objective function.

where the α_i are Lagrange multipliers. The search for an optimal *saddle point* $(\mathbf{w}_o, b_o, \boldsymbol{\alpha}_o)$ is necessary because Lagrangian L must be *minimized* with respect to \mathbf{w} and b , and has to be *maximized* with respect to nonnegative α_i (i.e., $\alpha_i \geq 0$ should be found). This problem can be solved either in a *primal space* (which is the space of parameters \mathbf{w} and b) or in a *dual space* (which is the space of Lagrange multipliers α_i). The second approach gives insightful results and we will consider the solution in a dual space below. In order to do that, we use Karush-Kuhn-Tucker (KKT) conditions for the optimum of a constrained function. In our case, both the objective function (11) and constraints (10b) are *convex* and KKT conditions are *necessary* and *sufficient* conditions for a maximum of (11). These conditions are:

at the saddle point $(\mathbf{w}_o, b_o, \boldsymbol{\alpha}_o)$, derivatives of Lagrangian L with respect to primal variables should vanish which leads to,

$$\frac{\partial L}{\partial \mathbf{w}_o} = 0, \text{ i.e., } \mathbf{w}_o = \sum_{i=1}^l \alpha_i y_i \mathbf{x}_i \quad (12)$$

$$\frac{\partial L}{\partial b_o} = 0, \text{ i.e., } \sum_{i=1}^l \alpha_i y_i = 0 \quad (13)$$

and the KKT complementarity conditions below (stating that at the solution point the products between dual variables and constraints equals zero) must also be satisfied,

$$\alpha_i \{y_i [\mathbf{w}^T \mathbf{x}_i + b] - 1\} = 0, \quad i = 1, l. \quad (14)$$

Substituting (12) and (13) into a *primal variables Lagrangian* $L(\mathbf{w}, b, \boldsymbol{\alpha})$ (11), we change to the *dual variables Lagrangian* $L_d(\boldsymbol{\alpha})$

$$L_d(\boldsymbol{\alpha}) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j. \quad (15)$$

In order to find the optimal hyperplane, a dual Lagrangian $L_d(\boldsymbol{\alpha})$ has to be *maximized* with respect to nonnegative α_i (i.e., α_i must be in the nonnegative quadrant) and with respect to the equality constraint as follows

$$\alpha_i \geq 0, \quad i = 1, l \quad (16a)$$

$$\sum_{i=1}^l \alpha_i y_i = 0 \quad (16b)$$

Note that the dual Lagrangian $L_d(\boldsymbol{\alpha})$ is expressed in terms of training data and depends *only* on the *scalar products* of input patterns $(\mathbf{x}_i^T \mathbf{x}_j)$. The dependency of $L_d(\boldsymbol{\alpha})$ on a scalar product of inputs will be very handy later when analyzing nonlinear decision boundaries and for general nonlinear regression. Note also that the number of unknown variables equals the number of training data l .

After learning, the number of free parameters is equal to the number of SVs but it does not depend on the dimensionality of input space. Such a *standard quadratic optimization problem* can be expressed in a *matrix notation* and formulated as follows:

Maximize

$$L_d(\alpha) = -0.5\alpha^T \mathbf{H}\alpha + \mathbf{f}^T \alpha , \quad (17a)$$

subject to

$$\mathbf{y}^T \alpha = 0 , \quad (17b)$$

$$\alpha_i \geq 0, \quad i = 1, l \quad (17c)$$

where $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_l]^T$, \mathbf{H} denotes the Hessian matrix ($H_{ij} = y_i y_j (\mathbf{x}_i \mathbf{x}_j) = y_i y_j \mathbf{x}_i^T \mathbf{x}_j$) of this problem, and \mathbf{f} is an $(l, 1)$ unit vector $\mathbf{f} = \mathbf{1} = [1 \ 1 \ \dots \ 1]^T$. (Note that maximization of (17a) equals a minimization of $L_d(\alpha) = 0.5\alpha^T \mathbf{H}\alpha - \mathbf{f}^T \alpha$, subject to the same constraints). Solutions α_{0i} of the dual optimization problem above determine the parameters \mathbf{w}_o and b_o of the optimal hyperplane according to (12) and (14) as follows

$$\mathbf{w}_o = \sum_{i=1}^l \alpha_{oi} y_i \mathbf{x}_i , \quad (18a)$$

$$\begin{aligned} b_o &= \frac{1}{N_{SV}} \left(\sum_{s=1}^{N_{SV}} \left(\frac{1}{y_s} - \mathbf{x}_s^T \mathbf{w}_o \right) \right) \\ &= \frac{1}{N_{SV}} \left(\sum_{s=1}^{N_{SV}} (y_s - \mathbf{x}_s^T \mathbf{w}_o) \right), \quad s = 1, N_{SV} . \end{aligned} \quad (18b)$$

In deriving (18b) we used the fact that y can be either $+1$ or -1 , and $1/y = y \cdot N_{SV}$ denotes the number of support vectors. There are two important observations about the calculation of \mathbf{w}_o . First, an optimal weight vector \mathbf{w}_o , is obtained in (18a) as a linear combination of the training data points and second, \mathbf{w}_o (same as the bias term b_o) is calculated by using only the selected data points called *support vectors* (SVs). The fact that the summations in (18a) goes over all training data patterns (i.e., from 1 to l) is irrelevant because the Lagrange multipliers for all non-support vectors equal zero ($\alpha_{0i} = 0$, $i = N_{SV} + 1, l$). Finally, having calculated \mathbf{w}_o and b_o we obtain a decision hyperplane $d(\mathbf{x})$ and an indicator function $i_F = o = \text{sign}(d(\mathbf{x}))$ as given below

$$d(\mathbf{x}) = \sum_{i=1}^l w_{oi} x_i + b_o = \sum_{i=1}^l y_i \alpha_i \mathbf{x}_i^T \mathbf{x} + b_o , \quad i_F = o = \text{sign}(d(\mathbf{x})) . \quad (19)$$

Training data patterns having non-zero Lagrange multipliers are called *support vectors*. For linearly separable training data, all support vectors lie on the margin and they are generally just a small portion of all training data (typically, $N_{SV} \ll l$). Figures 6, 7 and 8 show the geometry of standard results for non-overlapping classes.

Before presenting applications of OCSH for both overlapping classes and classes having nonlinear decision boundaries, we will comment only on whether and how SV based linear classifiers actually implement the SRM principle. The more detailed presentation of this important property can be found in [15, 23]. First, it can be shown that an increase in margin reduces the number of points that can be shattered i.e., the increase in margin reduces the VC dimension, and this leads to the decrease of the SVM capacity. In short, by minimizing $\|\mathbf{w}\|$ (i.e., maximizing the margin) the SV machine training actually minimizes the VC dimension and consequently a generalization error (expected risk) at the same time. This is achieved by imposing a structure on the set of canonical hyperplanes and then, during the training, by choosing the one with a minimal VC dimension. A structure on the set of canonical hyperplanes is introduced by considering various hyperplanes having different $\|\mathbf{w}\|$. In other words, we analyze sets S_A such that $\|\mathbf{w}\| \leq A$. Then, if $A_1 \leq A_2 \leq A_3 \leq \dots \leq A_n$, we introduce a nested set $S_{A1} \subset S_{A2} \subset S_{A3} \subset \dots \subset S_{An}$. Thus, if we impose the constraint $\|\mathbf{w}\| \leq A$, then the canonical hyperplane cannot be closer than $1/A$ to any of the training points \mathbf{x}_i . Vapnik in [30] states that the VC dimension h of a set of canonical hyperplanes in \mathbb{R}^n such that $\|\mathbf{w}\| \leq A$ is

$$H \leq \min[R^2 A^2, n] + 1, \quad (20)$$

where all the training data points (vectors) are enclosed by a sphere of the smallest radius R . Therefore, a small $\|\mathbf{w}\|$ results in a small h , and minimization of $\|\mathbf{w}\|$ is an implementation of the SRM principle. In other words, a minimization of the canonical hyperplane weight norm $\|\mathbf{w}\|$ minimizes the VC dimension according to (20). See also Fig. 4 that shows how the estimation error, meaning the expected risk (because the empirical risk, due to the linear separability, equals zero) decreases with a decrease of a VC dimension. Finally, there is an interesting, simple and powerful result [31] connecting the generalization ability of learning machines and the number of support vectors. Once the support vectors have been found, we can calculate the bound on the expected probability of committing an error on a test example as follows

$$E_l[P(\text{error})] \leq \frac{E[\text{number of support vectors}]}{l}, \quad (21)$$

where E_l denotes expectation over all training data sets of size l . Note how easy it is to estimate this bound that is independent of the dimensionality of the input space. Therefore, an SV machine having a small number of support vectors will have good generalization ability even in a very high-dimensional space.

Example below shows the SVM's learning of the weights for a simple separable data problem in both the primal and the dual domain. The small number and low dimensionality of data pairs is used in order to show the optimization steps analytically and graphically. The same reasoning will be in the case of high dimensional and large training data sets but for them, one has to rely on computers and the insight in solution steps is necessarily lost.

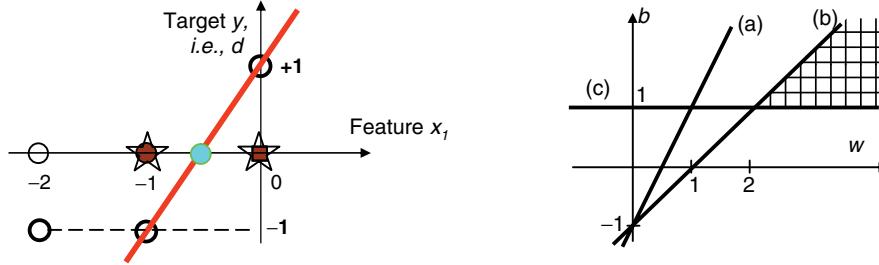


Fig. 9. *Left:* Solving SVM classifier for 3 data shown. SVs are star data. *Right:* Solution space $w - b$

Example: Consider a design of SVM classifier for 3 data shown in Fig. 9.

First we solve the problem in the *primal domain*: From the constraints (10b) it follows

$$\begin{aligned} 2w - 1 &\geq b, & (a) \\ w - 1 &\geq b, & (b) \\ b &\geq 1. & (c) \end{aligned}$$

The three straight lines corresponding to the equalities above are shown in Fig. 9 right. The textured area is a feasible domain for the weight w and bias b . Note that the area is not defined by the inequality (a), thus pointing to the fact that the point -1 is not a support vector. Points -1 and 0 define the textured area and they will be the supporting data for our decision function. The task is to minimize (10a), and this will be achieved by taking the value $w = 2$. Then, from (b), it follows that $b = 1$. Note that (a) must not be used for the calculation of the bias term b .

Because both the cost function (10a) and the constraints (10b) are convex, the primal and the dual solution must produce same w and b . Dual solution follows from maximizing (15) subject to (16a) as follows

$$L_d = \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2} [\alpha_1 \ \alpha_2 \ \alpha_3] \begin{bmatrix} 4 & 2 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix},$$

s.t. $-\alpha_1 - \alpha_2 + \alpha_3 = 0,$
 $\alpha_1 \geq 0, \ \alpha_2 \geq 0, \ \alpha_3 \geq 0,$

The dual Lagrangian is obtained in terms of α_1 and α_2 after expressing α_3 from the equality constraint and it is given as $L_d = 2\alpha_1 + 2\alpha_2 - 0.5(4\alpha_1^2 + 4\alpha_1\alpha_2 + \alpha_2^2)$. L_d will have maximum for $\alpha_1 = 0$, and it follows that we have to find the maximum of $L_d = 2\alpha_2 - 0.5\alpha_2^2$ which will be at $\alpha_2 = 2$. Note that the Hessian matrix is extremely bad conditioned and if the QP problem is to be solved by computer **H** should be regularized first. From the equality constraint it follows that $\alpha_3 = 2$ too. Now, we can calculate the weight vector w and the bias b from (18a) and (18b) as follows,

$$w = \sum_{i=1}^3 \alpha_i y_i \mathbf{x}_i = 0(-1)(-2) + 2(-1)(-1) + 2(1)0 = 2$$

The bias can be calculated by using SVs only, meaning from either point -1 or point 0 . Both result in same value as shown below

$$b = -1 - 2(-1) = 1, \text{ or } b = 1 - 2(0) = 1.$$

2.2 Linear Soft Margin Classifier for Overlapping Classes

The learning procedure presented above is valid for linearly separable data, meaning for training data sets without overlapping. Such problems are rare in practice. At the same time, there are many instances when linear separating hyperplanes can be good solutions even when data are overlapped (e.g., normally distributed classes having the same covariance matrices have a linear separation boundary). However, quadratic programming solutions as given above cannot be used in the case of overlapping because the constraints $y_i[\mathbf{w}^T \mathbf{x}_i + b] \geq 1, i = 1, l$ given by (10b) cannot be satisfied. In the case of an overlapping (see Fig. 10), the overlapped data points cannot be correctly classified and for any misclassified training data point \mathbf{x}_i , the corresponding α_i will tend to infinity. This particular data point (by increasing the corresponding α_i value) attempts to exert a stronger influence on the decision boundary

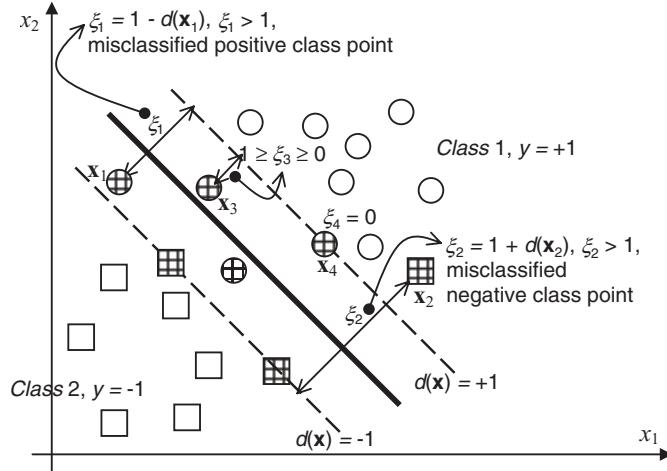


Fig. 10. The soft decision boundary for a dichotomization problem with data overlapping. Separation line (*solid*), margins (*dashed*) and support vectors (textured training data points)). 4 SVs in positive class (*circles*) and 3 SVs in negative class (*squares*). 2 misclassifications for positive class and 1 misclassification for negative class

in order to be classified correctly. When the α_i value reaches the maximal bound, it can no longer increase its effect, and the corresponding point will stay misclassified. In such a situation, the algorithm introduced above chooses (almost) all training data points as support vectors. To find a classifier with a maximal margin, the algorithm presented in the Sect. 2.1 above, must be changed allowing some data to be unclassified. Better to say, we must leave some data on the “wrong” side of a decision boundary. In practice, we allow a *soft* margin and all data inside this margin (whether on the correct side of the separating line or on the wrong one) are neglected. The width of a soft margin can be controlled by a corresponding penalty parameter C (introduced below) that determines the trade-off between the training error and VC dimension of the model.

The question now is how to measure the degree of misclassification and how to incorporate such a measure into the hard margin learning algorithm given by equations (10). The simplest method would be to form the following learning problem

$$\text{minimize } \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \text{ (number of misclassified data)} , \quad (22)$$

where C is a penalty parameter, trading off the margin size (defined by $\|\mathbf{w}\|$, i.e., by $\mathbf{w}^T \mathbf{w}$) for the number of misclassified data points. Large C leads to small number of misclassifications, bigger $\mathbf{w}^T \mathbf{w}$ and consequently to the smaller margin and vice versa. Obviously taking $C = \infty$ requires that the number of misclassified data is zero and, in the case of an overlapping this is not possible. Hence, the problem may be feasible only for some value $C < \infty$. However, the serious problem with (22) is that the error’s counting can’t be accommodated within the handy (meaning reliable, well understood and well developed) quadratic programming approach. Also, the counting only can’t distinguish between huge (or disastrous) errors and close misses! The possible solution is to measure the distances ξ_i of the points crossing the margin from the corresponding margin and trade their sum for the margin size as given below

$$\text{minimize } \frac{1}{2} \mathbf{w}^T \mathbf{w} + C(\text{sum of distances of the wrong side points}) , \quad (23)$$

In fact this is exactly how the problem of the data overlapping was solved in [5, 6] – by generalizing the optimal “hard” margin algorithm. They introduced the nonnegative *slack variables* ξ_i ($i = 1, l$) in the statement of the optimization problem for the overlapped data points. Now, instead of fulfilling (10a) and (10b), the separating hyperplane must satisfy

$$\text{minimize } \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^l \xi_i , \quad (24a)$$

subject to

$$y_i[\mathbf{w}^T \mathbf{x}_i + b] \geq 1 - \xi_i, \quad i = 1, l, \quad \xi_i \geq 0, \quad (24b)$$

i.e., subject to

$$\mathbf{w}^T \mathbf{x}_i + b \geq +1 - \xi_i, \quad \text{for } y_i = +1, \xi_i \geq 0, \quad (24c)$$

$$\mathbf{w}^T \mathbf{x}_i + b \leq -1 + \xi_i, \quad \text{for } y_i = -1, \xi_i \geq 0. \quad (24d)$$

Hence, for such a *generalized* optimal separating hyperplane, the functional to be minimized comprises an extra term accounting the cost of overlapping errors. In fact the cost function (24a) can be even more general as given below

$$\text{minimize} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^l \xi_i^k, \quad (24e)$$

subject to same constraints. This is a convex programming problem that is usually solved only for $k = 1$ or $k = 2$, and such soft margin SVMs are dubbed L1 and L2SVMs respectively. By choosing exponent $k = 1$, neither slack variables ξ_i nor their Lagrange multipliers β_i appear in a dual Lagrangian L_d . Same as for a linearly separable problem presented previously, for L1 SVMs ($k = 1$) here, the solution to a quadratic programming problem (24), is given by the saddle point of the primal Lagrangian $L_p(\mathbf{w}, b, \xi, \alpha, \beta)$ shown below

$$\begin{aligned} L_p(\mathbf{w}, b, \xi, \alpha, \beta) &= \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \left(\sum_{i=1}^l \xi_i \right) - \sum_{i=1}^l \alpha_i \{ y_i [\mathbf{w}^T \mathbf{x}_i + b] - 1 + \xi_i \} \\ &\quad - \sum_{i=1}^l \beta_i \xi_i, \quad \text{for L1 SVM} \end{aligned} \quad (25)$$

where α_i and β_i are the Lagrange multipliers. Again, we should find an *optimal* saddle point $(\mathbf{w}_o, b_o, \xi_o, \alpha_o, \beta_o)$ because the Lagrangian L_p has to be *minimized* with respect to \mathbf{w} , b and ξ , and *maximized* with respect to non-negative α_i and β_i . As before, this problem can be solved in either a *primal space* or *dual space* (which is the space of Lagrange multipliers α_i and β_i). Again, we consider a solution in a dual space as given below by using

– standard conditions for an optimum of a constrained function

$$\frac{\partial L}{\partial \mathbf{w}_o} = 0, \quad \text{i.e.,} \quad \mathbf{w}_o = \sum_{i=1}^l \alpha_i y_i \mathbf{x}_i, \quad (26)$$

$$\frac{\partial L}{\partial b_o} = 0, \quad \text{i.e.,} \quad \sum_{i=1}^l \alpha_i y_i = 0, \quad (27)$$

$$\frac{\partial L}{\partial \xi_{io}} = 0, \quad \text{i.e.,} \quad \alpha_i + \beta_i = C, \quad (28)$$

and the KKT complementarity conditions below,

$$\alpha_i \{y_i[\mathbf{w}^T \mathbf{x}_i + b] - 1 + \xi_i\} = 0, \quad i = 1, l, \quad (29a)$$

$$\beta_i \xi_i = (C - \alpha_i) \xi_i = 0, \quad i = 1, l. \quad (29b)$$

At the optimal solution, due to the KKT conditions (29), the last two terms in the primal Lagrangian L_p given by (25) vanish and the *dual variables Lagrangian* $L_d(\boldsymbol{\alpha})$, for L1 SVM, is not a function of β_i . In fact, it is same as the hard margin classifier's L_d given before and repeated here for the soft margin one,

$$L_d(\boldsymbol{\alpha}) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j. \quad (30)$$

In order to find the optimal hyperplane, a dual Lagrangian $L_d(\boldsymbol{\alpha})$ has to be *maximized* with respect to nonnegative and (unlike before) smaller than or equal to C, α_i . In other words with

$$C \geq \alpha_i \geq 0, \quad i = 1, l, \quad (31a)$$

and under the constraint (27), i.e., under

$$\sum_{i=1}^l \alpha_i y_i = 0. \quad (31b)$$

Thus, the final quadratic optimization problem is practically same as for the separable case the only difference being in the modified bounds of the Lagrange multipliers α_i . The penalty parameter C , which is now the upper bound on α_i , is determined by the user. The selection of a “good” or “proper” C is always done experimentally by using some cross-validation technique. Note that in the previous linearly separable case, without data overlapping, this upper bound $C = \infty$. We can also readily change to the matrix notation of the problem above as in equations (17). Most important of all is that the learning problem is expressed only in terms of unknown Lagrange multipliers α_i , and known inputs and outputs. Furthermore, optimization does not solely depend upon inputs \mathbf{x}_i which can be of a very high (inclusive of an infinite) dimension, but it depends upon a scalar product of input vectors \mathbf{x}_i . It is this property we will use in the next section where we design SV machines that can create nonlinear separation boundaries. Finally, expressions for both a *decision function* $d(\mathbf{x})$ and an *indicator function* $i_F = \text{sign}(d(\mathbf{x}))$ for a soft margin classifier are same as for linearly separable classes and are also given by (19).

From (29) follows that there are only three possible solutions for α_i (see Fig. 10)

1. $\alpha_i = 0, \xi_i = 0 \rightarrow$ data point \mathbf{x}_i is correctly classified,

2. $C > \alpha_i > 0, \rightarrow$ then, the two complementarity conditions must result in $y_i[\mathbf{w}^T \mathbf{x}_i + b] - 1 + \xi_i = 0$, and $\xi_i = 0$. Thus, $y_i[\mathbf{w}^T \mathbf{x}_i + b] = 1$ and \mathbf{x}_i is a support vector. The support vectors with $C \geq \alpha_i \geq 0$ are called *unbounded* or *free support vectors*. They lie on the two margins,
3. $\alpha_i = C, \rightarrow$ then, $y_i[\mathbf{w}^T \mathbf{x}_i + b] - 1 + \xi_i = 0$, and $\xi_i \geq 0$, and \mathbf{x}_i is a support vector. The support vectors with $\alpha_i = C$ are called *bounded support vectors*. They lie on the “wrong” side of the margin. For $1 > \xi_i \geq 0$, \mathbf{x}_i is still correctly classified, and if $\xi_i \geq 1$, \mathbf{x}_i is misclassified.

For L2 SVM the second term in the cost function (24e) is quadratic, i.e., $C \sum_{i=1}^l \xi_i^2$, and this leads to changes in a dual optimization problem which is now,

$$L_d(\boldsymbol{\alpha}) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l y_i y_j \alpha_i \alpha_j \left(\mathbf{x}_i^T \mathbf{x}_j + \frac{\delta_{ij}}{C} \right), \quad (32)$$

subject to

$$\alpha_i \geq 0, \quad i = 1, l, \quad (33a)$$

$$\sum_{i=1}^l \alpha_i y_i = 0. \quad (33b)$$

where, $\delta_{ij} = 1$ for $i = j$, and it is zero otherwise. Note the change in Hessian matrix elements given by second terms in (32), as well as that there is no upper bound on α_i . The detailed analysis and comparisons of the *L1* and *L2* SVMs is presented in [1]. Derivation of (32) and (33) is given in the Appendix. We use the most popular *L1* SVMs here, because they usually produce more sparse solutions, i.e., they create a decision function by using less SVs than the L2 SVMs.

2.3 The Nonlinear Classifier

The linear classifiers presented in two previous sections are very limited. Mostly, classes are not only overlapped but the genuine separation functions are nonlinear hypersurfaces. A nice and strong characteristic of the approach presented above is that it can be easily (and in a relatively straightforward manner) extended to create nonlinear decision boundaries. The motivation for such an extension is that an SV machine that can create a nonlinear decision hypersurface will be able to classify nonlinearly separable data. This will be achieved by considering a linear classifier in the so-called *feature space* that will be introduced shortly. A very simple example of a need for designing nonlinear models is given in Fig. 11 where the true separation boundary is quadratic. It is obvious that no errorless linear separating hyperplane can be found now. The best linear separation function shown as a dashed straight line would make six misclassifications (textured data points; 4 in the negative class and 2 in the positive one). Yet, if we use the nonlinear separation

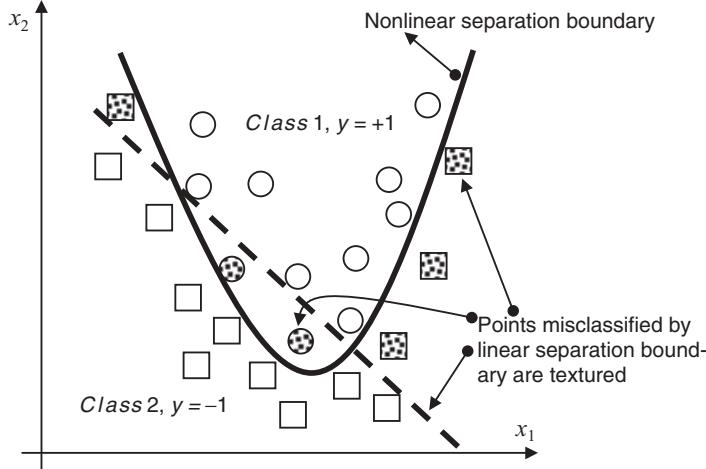


Fig. 11. A nonlinear SVM without data overlapping. A true separation is a quadratic curve. The nonlinear separation line (*solid*), the linear one (*dashed*) and data points misclassified by the linear separation line (the textured training data points) are shown. There are 4 misclassified negative data and 2 misclassified positive ones. SVs are not shown

boundary we are able to separate two classes without any error. Generally, for n -dimensional input patterns, instead of a nonlinear curve, an SV machine will create a nonlinear separating hypersurface.

The basic idea in designing nonlinear SV machines is to map input vectors $\mathbf{x} \in \Re^n$ into vectors $\Phi(\mathbf{x})$ of a higher dimensional *feature space* F (where Φ represents mapping: $\Re^n \rightarrow \Re^f$), and to solve a linear classification problem in this feature space

$$\mathbf{x} \in \Re^n \rightarrow \Phi(\mathbf{x}) = [\phi_1(\mathbf{x})\phi_2(\mathbf{x}), \dots, \phi_n(\mathbf{x})]^T \in \Re^f, \quad (34)$$

A mapping $\Phi(\mathbf{x})$ is chosen in advance. i.e., it is a fixed function. Note that an input space (\mathbf{x} -space) is spanned by components x_i of an input vector \mathbf{x} and a feature space F (Φ -space) is spanned by components $\phi_i(\mathbf{x})$ of a vector $\Phi(\mathbf{x})$. By performing such a mapping, we hope that in a Φ -space, our learning algorithm will be able to linearly separate images of \mathbf{x} by applying the linear SVM formulation presented above. (In fact, it can be shown that for a whole class of mappings the linear separation in a feature space is always possible. Such mappings will correspond to the positive definite kernels that will be shown shortly). We also expect this approach to again lead to solving a quadratic optimization problem with similar constraints in a Φ -space. The solution for an indicator function $i_F(\mathbf{x}) = \text{sign}(\mathbf{w}^T \Phi(\mathbf{x}) + b) = \text{sign}(\sum_{i=1}^l y_i \alpha_i \Phi^T(\mathbf{x}_i) \Phi(\mathbf{x}) + b)$, which is a linear classifier in a feature space, will create a nonlinear separating hypersurface in the original input space given by (35) below. (Compare this solution with (19) and note the

appearances of scalar products in both the original X -space and in the feature space F).

The equation for an $i_F(\mathbf{x})$ just given above can be rewritten in a “neural networks” form as follows

$$\begin{aligned} i_F(\mathbf{x}) &= \text{sign} \left(\sum_{i=1}^l y_i \alpha_i \Phi^T(\mathbf{x}_i) \Phi(\mathbf{x}) + b \right) \\ &= \text{sign} \left(\sum_{i=1}^l y_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b \right) = \text{sign} \left(\sum_{i=1}^l v_i k(\mathbf{x}_i, \mathbf{x}) + b \right) \quad (35) \end{aligned}$$

where v_i corresponds to the output layer weights of the “SVM’s network” and $k(\mathbf{x}_i, \mathbf{x})$ denotes the value of the kernel function that will be introduced shortly. (v_i equals $y_i \alpha_i$ in the classification case presented above and it is equal to $(\alpha_i - \alpha_i^*)$ in the regression problems). Note the difference between the weight vector \mathbf{w} which norm should be minimized and which is the vector of the same dimension as the feature space vector $\Phi(\mathbf{x})$ and the weightings $v_i = \alpha_i y_i$ that are scalar values composing the weight vector \mathbf{v} which dimension equals the number of training data points l . The $(l - N_{SV_s})$ of v_i components are equal to zero, and only N_{SV_s} entries of \mathbf{v} are nonzero elements.

A simple example below (Fig. 12) should exemplify the idea of a nonlinear mapping to (usually) higher dimensional space and how it happens that the data become linearly separable in the F -space.

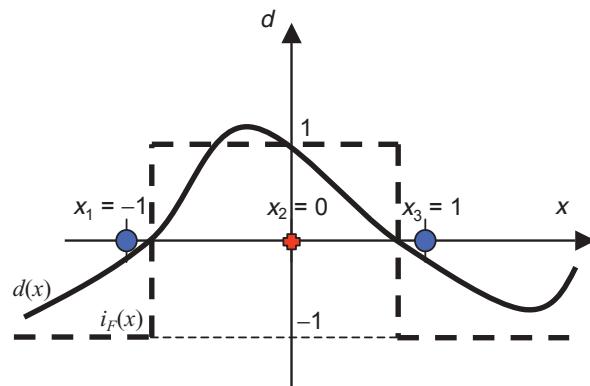


Fig. 12. A nonlinear 1-dimensional classification problem. One possible solution is given by the decision function $d(x)$ (solid curve) i.e., by the corresponding indicator function defined as $i_F = \text{sign}(d(x))$ (dashed stepwise function)

Consider solving the simplest 1-D classification problem given the input and the output (desired) values as follows: $\mathbf{x} = [-1 \ 0 \ 1]^T$ and $\mathbf{d} = \mathbf{y} = [-1 \ 1 \ -1]^T$. Here we choose the following mapping to the feature space:

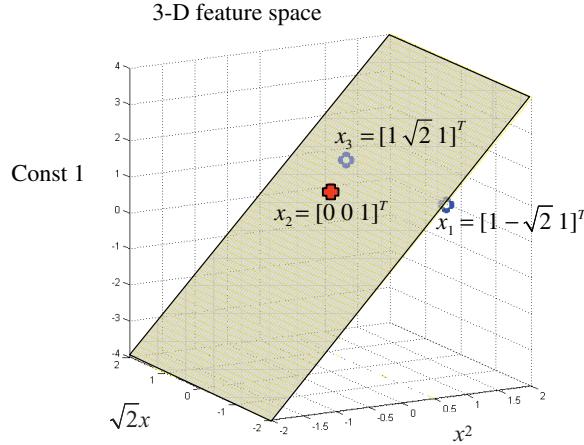


Fig. 13. The three data points of a problem in Fig. 12 are linearly separable in the feature space (obtained by the mapping $\Phi(\mathbf{x}) = [\varphi_1(\mathbf{x}) \varphi_2(\mathbf{x}) \varphi_3(\mathbf{x})]^T = [x^2 \sqrt{2}x 1]^T$). The separation boundary is given as the plane $\varphi_3(\mathbf{x}) = 2\varphi_1(\mathbf{x})$ shown in the figure

$\Phi(\mathbf{x}) = [\varphi_1(\mathbf{x}) \varphi_2(\mathbf{x}) \varphi_3(\mathbf{x})]^T = [x^2 \sqrt{2}x 1]^T$. The mapping produces the following three points in the feature space (shown as the rows of the matrix \mathbf{F} (F standing for features))

$$\mathbf{F} = \begin{bmatrix} 1 & -\sqrt{2} & 1 \\ 0 & 0 & 1 \\ 1 & \sqrt{2} & 1 \end{bmatrix} .$$

These three points are linearly separable by the plane $\varphi_3(\mathbf{x}) = 2\varphi_1(\mathbf{x})$ in a feature space as shown in Fig. 13. It is easy to show that the mapping obtained by $\Phi(\mathbf{x}) = [x^2 \sqrt{2}x 1]^T$ is a scalar product implementation of a quadratic kernel function $(\mathbf{x}_i^T \mathbf{x}_j + 1)^2 = k(\mathbf{x}_i, \mathbf{x}_j)$. In other words, $\Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j)$. This equality will be introduced shortly.

There are two basic problems when mapping an input \mathbf{x} -space into higher order F -space:

- (i) the choice of mapping $\Phi(\mathbf{x})$ that should result in a “rich” class of decision hypersurfaces,
- (ii) the calculation of the scalar product $\Phi^T(\mathbf{x})\Phi(\mathbf{x})$ that can be computationally very discouraging if the number of features f (i.e., dimensionality f of a feature space) is very large.

The second problem is connected with a phenomenon called the “*curse of dimensionality*”. For example, to construct a decision surface corresponding to a polynomial of degree *two* in an n -D input space, a dimensionality of a

feature space $f = n(n + 3)/2$. In other words, a feature space is spanned by f coordinates of the form

$$z_1 = x_1, \dots, z_n = x_n \text{ (}n\text{ coordinates)}, z_{n+1} = (x_1)^2, \dots, z_{2n} = (x_n)^2 \text{ (next }n\text{ coordinates)}, z_{2n+1} = x_1 x_2, \dots, z_f = x_n x_{n-1} \text{ (}n(n - 1)/2\text{ coordinates)},$$

and the separating hyperplane created in this space, is a second-degree polynomial in the input space [33]. Thus, constructing a polynomial of degree two only, in a 256-dimensional input space, leads to a dimensionality of a feature space $f = 33,152$. Performing a scalar product operation with vectors of such, or higher, dimensions, is not a cheap computational task. The problems become serious (and fortunately only seemingly unsolvable) if we want to construct a polynomial of degree 4 or 5 in the same 256-dimensional space leading to the construction of a decision hyperplane in a billion-dimensional feature space.

This explosion in dimensionality can be avoided by noticing that in the quadratic optimization problem given by (15) and (30), as well as in the final expression for a classifier, *training data only appear in the form of scalar products $\mathbf{x}_i^T \mathbf{x}_j$* . These products will be replaced by scalar products $\Phi^T(\mathbf{x})\Phi(\mathbf{x})_i = [\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \dots, \phi_n(\mathbf{x})]^T [\phi_1(\mathbf{x}_i), \phi_2(\mathbf{x}_i), \dots, \phi_n(\mathbf{x}_i)]$ in a feature space F , and the latter can be and will be expressed by using the *kernel function* $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j)$.

Note that a *kernel function* $K(\mathbf{x}_i, \mathbf{x}_j)$ is a function in input space. Thus, the basic advantage in using kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ is in avoiding performing a mapping $\Phi(\mathbf{x})$ et all. Instead, the required scalar products in a feature space $\Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j)$, are calculated directly by computing kernels $K(\mathbf{x}_i, \mathbf{x}_j)$ for given training data vectors in an input space. In this way, we bypass a possibly extremely high dimensionality of a feature space F . Thus, by using the chosen kernel $K(\mathbf{x}_i, \mathbf{x}_j)$, we can construct an SVM that operates in an infinite dimensional space (such an kernel function is a Gaussian kernel function given in Table 2). In addition, as will be shown below, by applying kernels we do not even have to know what the actual mapping $\Phi(\mathbf{x})$ is. A kernel is a function K such that

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j). \quad (36)$$

There are many possible kernels, and the most popular ones are given in Table 2. All of them should fulfill the so-called Mercer's conditions. The Mercer's kernels belong to a set of *reproducing kernels*. For further details see [2, 15, 19, 24, 33].

The simplest is a linear kernel defined as $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$. Below we show a few more kernels:

Polynomial Kernels

Let $\mathbf{x} \in \Re^2$ i.e., $\mathbf{x} = [x_1 \ x_2]^T$, and if we choose $\Phi(\mathbf{x}) = [x_1^2 \ \sqrt{2}x_1x_2 \ x_2^2]^T$ (i.e., there is an $\Re^2 \rightarrow \Re^3$ mapping), then the dot product

Table 2. Popular Admissible Kernels

| Kernel Functions | Type of Classifier |
|---|--|
| $K(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x}^T \mathbf{x}_i)$ | Linear, dot product, kernel, CPD |
| $K(\mathbf{x}, \mathbf{x}_i) = [(\mathbf{x}^T \mathbf{x}_i) + 1]^d$ | Complete polynomial of degree d , PD |
| $K(\mathbf{x}, \mathbf{x}_i) = e^{\frac{1}{2}[(\mathbf{x} - \mathbf{x}_i)^T \Sigma^{-1}(\mathbf{x} - \mathbf{x}_i)]}$ | Gaussian RBF, PD |
| $K(\mathbf{x}, \mathbf{x}_i) = \tanh [(\mathbf{x}^T \mathbf{x}_i) + b]^*$ | Multilayer perceptron, CPD |
| $K(\mathbf{x}, \mathbf{x}_i) = \frac{1}{\sqrt{\ \mathbf{x} - \mathbf{x}_i\ ^2 + \beta}}$ | Inverse multiquadric function, PD |

* only for certain values of b , (C)PD = (conditionally) positive definite

$$\begin{aligned}\Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j) &= \begin{bmatrix} x_{i1}^2 & \sqrt{2}x_{i1}x_{i2} & x_{i1}^2 \end{bmatrix} \begin{bmatrix} x_{j1}^2 & \sqrt{2}x_{j1}x_{j2} & x_{j1}^2 \end{bmatrix}^T \\ &= [x_{i1}^2 x_{j1}^2 + 2x_{i1}x_{i2}x_{j1}x_{j2} + x_{i2}^2 x_{j2}^2] \\ &= (\mathbf{x}_i^T \mathbf{x}_j)^2 = K(\mathbf{x}_i, \mathbf{x}_j), \text{ or} \\ &K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^2 = \Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j)\end{aligned}$$

Note that in order to calculate the scalar product in a feature space $\Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j)$, we do not need to perform the mapping $\Phi(\mathbf{x}) = [x_1^2 \sqrt{2}x_1x_2 x_1^2]^T$ et all. Instead, we calculate this product directly in the input space by computing $(\mathbf{x}_i^T \mathbf{x}_j)^2$. This is very well known under the popular name of *the kernel trick*. Interestingly, note also that other mappings such as an

$$\begin{aligned}\Re^2 \rightarrow \Re^3 \text{ mapping given by } \Phi(\mathbf{x}) &= [x_1^2 - x_2^2 \quad 2x_1x_2 \quad x_1^2 + x_2^2], \text{ or an} \\ \Re^2 \rightarrow \Re^4 \text{ mapping given by } \Phi(\mathbf{x}) &= [x_1^2 \quad x_1x_2 \quad x_1x_2 \quad x_2^2]\end{aligned}$$

also accomplish the same task as $(\mathbf{x}_i^T \mathbf{x}_j)^2$.

Now, assume the following mapping

$$\Phi(\mathbf{x}) = [1 \quad \sqrt{2}x_1 \quad \sqrt{2}x_2 \quad \sqrt{2}x_1x_2 \quad x_1^2 \quad x_2^2],$$

i.e., there is an $\Re^2 \rightarrow \Re^5$ mapping plus bias term as the constant 6th dimension's value. Then the dot product in a feature space F is given as

$$\begin{aligned}\Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j) &= 1 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} + 2x_{i1}x_{i2}x_{j1}x_{i2} + x_{i1}^2x_{j1}^2 + x_{i2}^2x_{j2}^2 \\ &= 1 + 2(\mathbf{x}_i^T \mathbf{x}_j) + (\mathbf{x}_i^T \mathbf{x}_j)^2 = (\mathbf{x}_i^T \mathbf{x}_j + 1)^2 = K(\mathbf{x}_i, \mathbf{x}_j), \text{ or} \\ &K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + 1)^2 = \Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j).\end{aligned}$$

Thus, the last mapping leads to the second order *complete* polynomial.

Many candidate functions can be applied to a convolution of an inner product (i.e., for kernel functions) $K(\mathbf{x}, \mathbf{x}_i)$ in an SV machine. Each of these functions constructs a different nonlinear decision hypersurface in an input space. In the first three rows, the Table 2 shows the three most popular kernels

in SVMs' in use today, and the inverse multiquadric one as an interesting and powerful kernel to be proven yet.

The positive definite (PD) kernels are the kernels which Gramm matrix \mathbf{G} (a.k.a. Grammian) calculated by using all the l training data points is positive definite (meaning all its eigenvalues are strictly positive, i.e., $\lambda_i > 0$, $i = 1, l$)

$$\mathbf{G} = \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \cdots & k(\mathbf{x}_1, \mathbf{x}_l) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \vdots & k(\mathbf{x}_2, \mathbf{x}_l) \\ \vdots & \vdots & \vdots & \vdots \\ k(\mathbf{x}_l, \mathbf{x}_1) & k(\mathbf{x}_l, \mathbf{x}_2) & \cdots & k(\mathbf{x}_l, \mathbf{x}_l) \end{bmatrix} \quad (37)$$

The kernel matrix \mathbf{G} is a symmetric one. Even more, any symmetric positive definite matrix can be regarded as a kernel matrix, that is - as an inner product matrix in some space.

Finally, we arrive at the point of presenting the learning in nonlinear classifiers (in which we are ultimately interested here). The learning algorithm for a nonlinear SV machine (classifier) follows from the design of an *optimal separating hyperplane* in a *feature space*. This is the same procedure as the construction of a "hard" (15) and "soft" (30) margin classifiers in an \mathbf{x} -space previously. In a $\Phi(\mathbf{x})$ -space, the dual Lagrangian, given previously by (15) and (30), is now

$$L_d(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l \alpha_i \alpha_j y_i y_j \Phi_i^T \Phi_j, \quad (38)$$

and, according to (36), by using chosen kernels, we should maximize the following dual Lagrangian

$$L_d(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \quad (39)$$

subject to

$$\alpha_i \geq 0, \quad i = 1, l \quad \text{and} \quad \sum_{i=1}^l \alpha_i y_i = 0. \quad (39a)$$

In a more general case, because of a noise or due to generic class" features, there will be an overlapping of training data points. Nothing but constraints for α_i change. Thus, the nonlinear "soft" margin classifier will be the solution of the quadratic optimization problem given by (39) subject to constraints

$$C \geq \alpha_i \geq 0, \quad i = 1, l \quad \text{and} \quad \sum_{i=1}^l \alpha_i y_i = 0. \quad (39b)$$

Again, the only difference to the separable nonlinear classifier is the upper bound C on the Lagrange multipliers α_i . In this way, we limit the influence

of training data points that will remain on the “wrong” side of a separating nonlinear hypersurface. After the dual variables are calculated, the decision hypersurface $d(\mathbf{x})$ is determined by

$$d(\mathbf{x}) = \sum_{i=1}^l y_i \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b = \sum_{i=1}^l v_i K(\mathbf{x}, \mathbf{x}_i) + b, \quad (40)$$

and the indicator function is $i_F(\mathbf{x}) = \text{sign}[d(\mathbf{x})] = \text{sign}[\sum_{i=1}^l v_i K(\mathbf{x}, \mathbf{x}_i) + b]$.

Note that the summation is not actually performed over all training data but rather over the support vectors, because only for them do the Lagrange multipliers differ from zero. The existence and calculation of a bias b is now not a direct procedure as it is for a linear hyperplane. Depending upon the applied kernel, the bias b can be implicitly part of the kernel function. If, for example, Gaussian RBF is chosen as a kernel, it can use a bias term as the $f+1^{\text{st}}$ feature in F -space with a constant output = +1, but not necessarily. In short, all PD kernels do not necessarily need an explicit bias term b , but b can be used. (More on this can be found in [17] as well as in the [34]. Same as for the linear SVM, (39) can be written in a matrix notation as

maximize

$$L_d(\boldsymbol{\alpha}) = -0.5 \boldsymbol{\alpha}^T \mathbf{H} \boldsymbol{\alpha} + \mathbf{f}^T \boldsymbol{\alpha}, \quad (41a)$$

subject to

$$\mathbf{y}^T \boldsymbol{\alpha} = 0, \quad (41b)$$

$$C \geq \alpha_i \geq 0, \quad i = 1, l, \quad (41c)$$

where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \dots, \alpha_l]^T$, \mathbf{H} denotes the Hessian matrix ($H_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$) of this problem and \mathbf{f} is an $(l, 1)$ unit vector $\mathbf{f} = \mathbf{1} = [1 1 \dots 1]^T$. Note that if $K(\mathbf{x}_i, \mathbf{x}_j)$ is the positive definite matrix, so is the matrix $y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$. too.

The following 1-D example (just for the sake of graphical presentation) will show the creation of a linear decision function in a feature space and a corresponding nonlinear (quadratic) decision function in an input space.

Suppose we have 4 1-D data points given as $x_1 = 1$, $x_2 = 2$, $x_3 = 5$, $x_4 = 6$, with data at 1, 2, and 6 as class 1 and the data point at 5 as class 2, i.e., $y_1 = -1$, $y_2 = -1$, $y_3 = 1$, $y_4 = -1$. We use the polynomial kernel of degree 2, $K(x, y) = (xy + 1)^2$. C is set to 50, which is of lesser importance because the constraints will be not imposed in this example for maximal value for the dual variables alpha will be smaller than $C = 50$.

Case 1: Working with a bias term b as given in (40).

We first find $\alpha_i (i = 1, \dots, 4)$ by solving dual problem (41) having a Hessian matrix

$$\mathbf{H} = \begin{bmatrix} 4 & 9 & -36 & 49 \\ 9 & 25 & -121 & 169 \\ -36 & -121 & 676 & -961 \\ 49 & 169 & -961 & 1369 \end{bmatrix}.$$

Alphas are $\alpha_1 = 0$, $\alpha_2 = 2.499999$, $\alpha_3 = 7.333333$, $\alpha_4 = 4.833333$ and the bias b will be found by using (18b), or by fulfilling the requirements that the values of a decision function at the support vectors should be the given y_i . The model (decision function) is given by

$$\begin{aligned} d(x) &= \sum_{i=1}^4 y_i \alpha_i K(x, x_i) + b = \sum_{i=1}^4 v_i (x x_i + 1)^2 + b, \text{ or by} \\ d(x) &= 2.499999(-1)(2x+1)^2 + 7.333333(1)(5x+1)^2 \\ &\quad + 4.833333(-1)(6x+1)^2 + b \\ d(x) &= -0.666667x^2 + 5.333333x + b. \end{aligned}$$

Bias b is determined from the requirement that at the SV points 2, 5 and 6, the outputs must be -1 , 1 and -1 respectively. Hence, $b = -9$, resulting in the decision function

$$d(x) = -0.666667x^2 + 5.333333x - 9.$$

The nonlinear (quadratic) decision function and the indicator one are shown in Fig. 14. Note that in calculations above 6 decimal places have been used for alpha values. The calculation is numerically very sensitive, and working with fewer decimals can give very approximate or wrong results.

The complete polynomial kernel as used in the case 1, is *positive definite* and there is no need to use an explicit bias term b as presented above. Thus, one can use the same second order polynomial model without the bias term b . Note that in this particular case there is no equality constraint equation that originates from an equalization of the primal Lagrangian derivative in respect to the bias term b to zero. Hence, we do not use (41b) while using a positive definite kernel without bias as it will be shown below in the case 2.

Case 2: Working without a bias term b

Because we use the same second order polynomial kernel, the Hessian matrix \mathbf{H} is same as in the case 1. The solution without the equality constraint for alphas is: $\alpha_1 = 0$, $\alpha_2 = 24.999999$, $\alpha_3 = 43.333333$, $\alpha_4 = 27.333333$. The model (decision function) is given by

$$\begin{aligned} d(x) &= \sum_{i=1}^4 y_i \alpha_i K(x, x_i) = \sum_{i=1}^4 v_i (x x_i + 1)^2, \text{ or by} \\ d(x) &= 2.499999(-1)(2x+1)^2 + 43.333333(1)(5x+1)^2 \\ &\quad + 27.333333(-1)(6x+1)^2 \\ d(x) &= -0.666667x^2 + 5.333333x - 9. \end{aligned}$$

Thus the nonlinear (quadratic) decision function and consequently the indicator function in the two particular cases are equal.

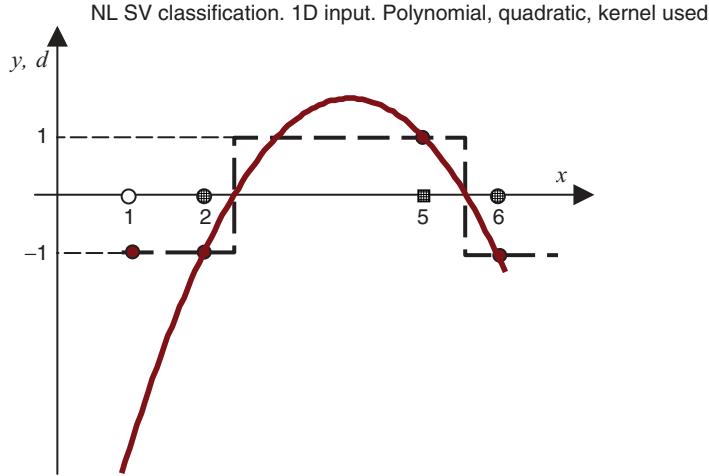


Fig. 14. The nonlinear decision function (*solid*) and the indicator function (*dashed*) for 1-D overlapping data. By using a complete second order polynomial the model with and without a bias term b are same

XOR Example

In the *next example* shown by Figs. 14 and 15 we present all the important mathematical objects of a nonlinear SV classifier by using a classic XOR (*exclusive-or*) problem. The graphs show all the mathematical functions (objects) involved in a nonlinear classification. Namely, the nonlinear decision function $d(\mathbf{x})$, the NL indicator function $i_F(\mathbf{x})$, training data (\mathbf{x}_i) , support vectors $(\mathbf{x}_{SV})_i$ and separation boundaries.

The same objects will be created in the cases when the input vector \mathbf{x} is of a dimensionality $n > 2$, but the visualization in these cases is not possible. In such cases one talks about the decision hyperfunction (hypersurface) $d(\mathbf{x})$, indicator hyperfunction (hypersurface) $i_F(\mathbf{x})$, training data (\mathbf{x}_i) , support vectors $(\mathbf{x}_{SV})_i$ and separation hyperboundaries (hypersurfaces).

Note the different character of a $d(\mathbf{x})$, $i_F(\mathbf{x})$ and separation boundaries in the two graphs given below. However, in both graphs all the data are correctly classified.

The analytic solution to the Fig. 16 for the *second order polynomial kernel* (i.e., for $(\mathbf{x}_i^T \mathbf{x}_j + 1)^2 = \Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j)$, where $\Phi(\mathbf{x}) = [1 \sqrt{2}x_1 \sqrt{2}x_2 \sqrt{2}x_1x_2 x_1^2 x_2^2]$, no explicit bias and $C = \infty$) goes as follows. Inputs and desired outputs are, $\mathbf{x} = [0 \ 1 \ 1 \ 0]^T$, $\mathbf{y} = \mathbf{d} = [1 \ 1 \ -1 \ -1]^T$. The dual Lagrangian (39) has the Hessian matrix

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & 9 & -4 & -4 \\ -1 & -4 & 4 & 1 \\ -1 & -4 & 1 & 4 \end{bmatrix}.$$

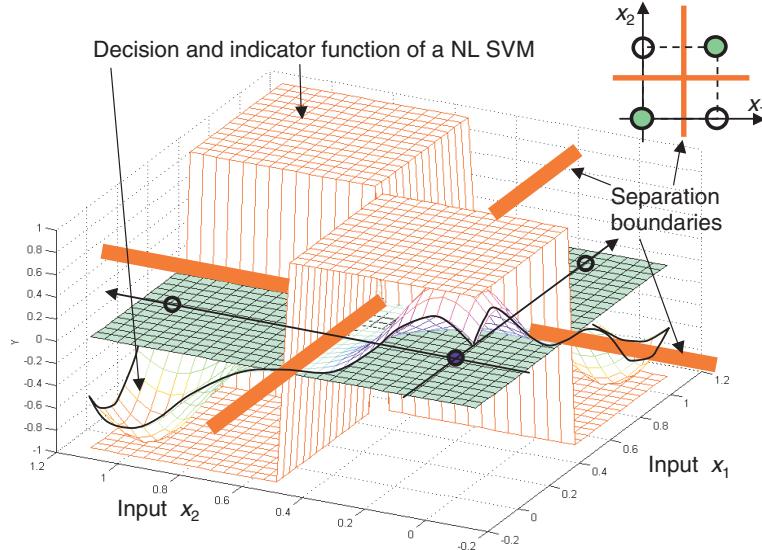


Fig. 15. XOR problem. Kernel functions (2-D Gaussians) are not shown. The non-linear decision function, the nonlinear indicator function and the separation boundaries are shown. All four data are chosen as support vectors

The optimal solution can be obtained by taking the derivative of L_d with respect to dual variables α_i ($i = 1, 4$) and by solving the resulting linear system of equations taking into account the constraints, see [17]. The solution to

$$\begin{aligned} \alpha_1 + \alpha_2 - \alpha_3 - \alpha_4 &= 1, \\ \alpha_1 + 9\alpha_2 - 4\alpha_3 - 4\alpha_4 &= 1, \\ -\alpha_1 - 4\alpha_2 + 4\alpha_3 + \alpha_4 &= 1, \\ -\alpha_1 - 4\alpha_2 + \alpha_3 + 4\alpha_4 &= 1, \end{aligned}$$

subject to $\alpha_i > 0$, ($i = 1, 4$), is $\alpha_1 = 4.3333$, $\alpha_2 = 2.0000$, $\alpha_3 = 2.6667$ and $\alpha_4 = 2.6667$. The decision function in a 3-D space is

$$\begin{aligned} d(\mathbf{x}) &= \sum_{i=1}^4 y_i \alpha_i \Phi^T(\mathbf{x}_i) \Phi(\mathbf{x}) \\ &= (4.3333 [1 \ 0 \ 0 \ 0 \ 0 \ 0] + 2 [1 \ \sqrt{2} \ \sqrt{2} \ \sqrt{2} \ 1 \ 1] \\ &\quad - 2.6667 [1 \ \sqrt{2} \ 0 \ 0 \ 1 \ 0] \\ &\quad - 2.6667 [1 \ 0 \ \sqrt{2} \ 0 \ 0 \ 1]) \Phi(\mathbf{x}) \\ &= [1 \ -0.9429 \ -0.9429 \ 2.8284 \ -0.6667 \ -0.6667] \\ &\quad \times [1 \ \sqrt{2}x_1 \ \sqrt{2}x_2 \ \sqrt{2}x_1x_2 \ x_1^2 \ x_2^2]^T, \end{aligned}$$

and finally

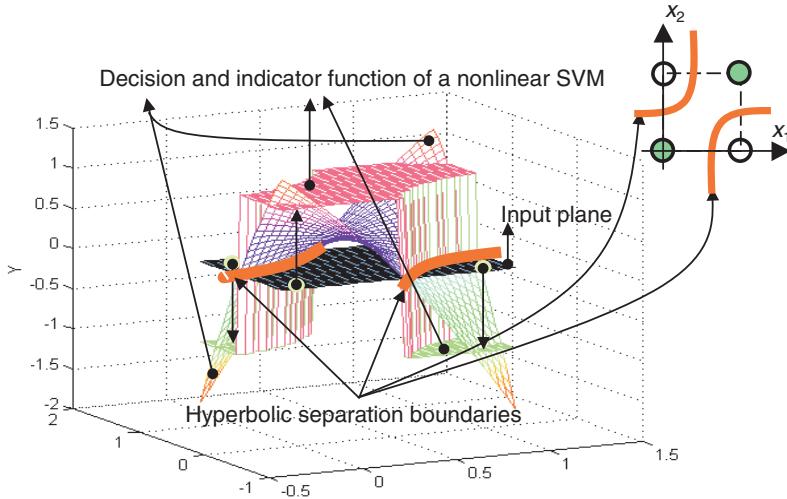


Fig. 16. XOR problem. Kernel functions (2-D polynomial.) The nonlinear decision function, the nonlinear indicator function and the separation boundaries are shown. All four data are support vectors

$$d(\mathbf{x}) = 1 - 1.3335x_1 - 1.3335x_2 + 4x_1x_2 - 0.6667x_1^2 - 0.6667x_2^2 .$$

It is easy to check that the values of $d(\mathbf{x})$ for all the training inputs in \mathbf{x} equal the desired values in \mathbf{d} . The $d(\mathbf{x})$ is the saddle-like function shown in Fig. 16.

Here we have shown the derivation of an expression for $d(\mathbf{x})$ by using explicitly a mapping Φ . Again, we do not have to know what mapping Φ is at all. By using *kernels* in *input space*, we calculate a *scalar product* required in a (*possibly high dimensional*) *feature space* and we avoid mapping $\Phi(\mathbf{x})$. This is known as kernel “trick”. It can also be useful to remember that the way in which the kernel “trick” was applied in designing an SVM can be utilized in all other algorithms that depend on the scalar product (e.g., in principal component analysis or in the nearest neighbor procedure).

2.4 Regression by Support Vector Machines

In the regression, we estimate the functional dependence of the dependent (output) variable $y \in \Re$ on an n -dimensional input variable \mathbf{x} . Thus, unlike in pattern recognition problems (where the desired outputs y_i are discrete values e.g., Boolean) we deal with *real valued* functions and we model an \Re^n to \Re^1 mapping here. Same as in the case of classification, this will be achieved by training the SVM model on a training data set first. Interestingly and importantly, a learning stage will end in the same shape of a dual Lagrangian as in classification, only difference being in a dimensionalities of the Hessian matrix and corresponding vectors which are of a double size now e.g., \mathbf{H} is a $(2l, 2l)$ matrix.

Initially developed for solving classification problems, SV techniques can be successfully applied in regression, i.e., for a functional approximation problems [8, 32]. The general regression learning problem is set as follows – the learning machine is given l training data from which it attempts to learn the input-output relationship (dependency, mapping or function) $f(\mathbf{x})$. A training data set $D = \{[\mathbf{x}(i), y(i)] \in \Re^n \times \Re, i = 1, \dots, l\}$ consists of l pairs (\mathbf{x}_1, y_1) , (\mathbf{x}_2, y_2) , \dots , (\mathbf{x}_l, y_l) , where the inputs \mathbf{x} are n -dimensional vectors $\mathbf{x} \in \Re^n$ and system responses $y \in \Re$, are continuous values.

We introduce all the relevant and necessary concepts of SVM's regression in a gentle way starting again with a *linear regression hyperplane* $f(\mathbf{x}, \mathbf{w})$ given as

We introduce all the relevant and necessary concepts of SVM's regression in a gentle way starting again with a *linear regression hyperplane* $f(\mathbf{x}, \mathbf{w})$ given as

$$f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \mathbf{x} + b. \quad (42)$$

In the case of SVM's regression, we measure the *error of approximation* instead of the margin used in classification. The most important difference in respect to classic regression is that we use a novel loss (error) functions here. This is the Vapnik's *linear loss function* with ε -insensitivity zone defined as

$$E(\mathbf{x}, y, f) = |y - f(\mathbf{x}, \mathbf{w})|_\varepsilon = \begin{cases} 0 & \text{if } |y - f(\mathbf{x}, \mathbf{w})| \leq \varepsilon \\ |y - f(\mathbf{x}, \mathbf{w})| - \varepsilon, & \text{otherwise.} \end{cases}, \quad (43a)$$

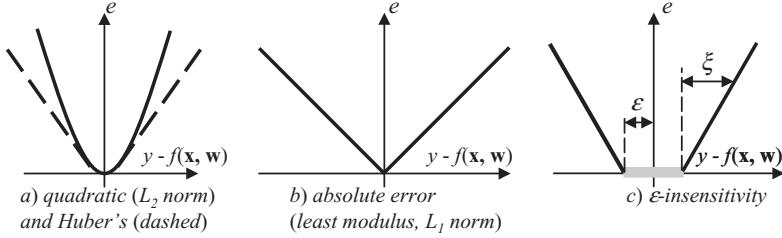
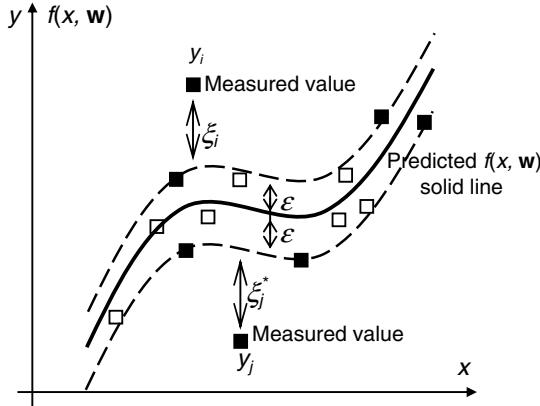
or as,

$$e(\mathbf{x}, y, f) = \max(0, |y - f(\mathbf{x}, \mathbf{w})| - \varepsilon). \quad (43b)$$

Thus, the loss is equal to 0 if the difference between the predicted $f(\mathbf{x}_i, \mathbf{w})$ and the measured value y_i is less than ε . Vapnik's ε -insensitivity loss function (43) defines an ε tube (Fig. 18). If the predicted value is within the tube the loss (error or cost) is zero. For all other predicted points outside the tube, the loss equals the magnitude of the difference between the predicted value and the radius ε of the tube.

The two classic error functions are: a square error, i.e., L_2 norm $(y - f)^2$, as well as an absolute error, i.e., L_1 norm, least modulus $|y - f|$ introduced by Yugoslav scientist Rudjer Boskovic in 18th century [9]. The latter error function is related to Huber's error function. An application of Huber's error function results in a *robust regression*. It is the most reliable technique if nothing specific is known about the model of a noise. We do no present Huber's loss function here in analytic form. Instead, we show it by a dashed curve in Fig. 16a. In addition, Fig. 16 shows typical shapes of all mentioned error (loss) functions above.

Note that for $\varepsilon = 0$, Vapnik's loss function equals a least modulus function. Typical graph of a (nonlinear) regression problem as well as all relevant mathematical objects required in learning unknown coefficients w_i are shown in Fig. 17.

**Fig. 17.** Loss (error) functions**Fig. 18.** The parameters used in (1-D) support vector regression. Filled square data ■ are support vectors, and the empty □ ones are not. Hence, SVs can appear only on the tube boundary or outside the tube

We will formulate an SVM regression's algorithm for the linear case first and then, for the sake of an NL model design, we will apply mapping to a feature space, utilize the kernel "trick" and construct a nonlinear regression hypersurface. This is actually the same order of presentation as in classification tasks. Here, for the regression, we "measure" the empirical error term R_{emp} by Vapnik's ϵ -insensitivity loss function given by (43) and shown in Fig. 16c (while the minimization of the confidence term Ω will be realized through a minimization of $\mathbf{w}^T \mathbf{w}$ again). The empirical risk is given as

$$R_{\text{emp}}^\epsilon(\mathbf{w}, b) = \frac{1}{l} \sum_{i=1}^l |y_i - \mathbf{w}^T \mathbf{x}_i - b|_\epsilon, \quad (44)$$

Figure 18 shows two linear approximating functions as dashed lines inside an ϵ -tube having the same empirical risk R_{emp}^ϵ as the regression function $f(\mathbf{x}, \mathbf{w})$ on the training data.

As in classification, we try to minimize both the empirical risk R_{emp}^ϵ and $\|\mathbf{w}\|^2$ simultaneously. Thus, we construct a linear regression hyperplane

$f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \mathbf{x} + b$ by minimizing

$$R = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^l |y_i - f(\mathbf{x}_i, \mathbf{w})|_\varepsilon . \quad (45)$$

Note that the last expression resembles the ridge regression scheme. However, we use Vapnik's ε -insensitivity loss function instead of a squared error now. From (43a) and Fig. 17 it follows that for all training data outside an ε -tube,

$$\begin{aligned} |y - f(\mathbf{x}, \mathbf{w})| - \varepsilon &= \xi && \text{for data "above" an } \varepsilon\text{-tube, or} \\ |y - f(\mathbf{x}, \mathbf{w})| - \varepsilon &= \xi^* && \text{for data "below" an } \varepsilon\text{-tube .} \end{aligned}$$

Thus, minimizing the risk R above equals the minimization of the following risk

$$R_{\mathbf{w}, \xi, \xi^*} = \left[\frac{1}{2} \|\mathbf{w}\|^2 + C \left(\sum_{i=1}^l \xi_i + \sum_{i=1}^l \xi_i^* \right) \right] , \quad (46)$$

under constraints

$$y_i - \mathbf{w}^T \mathbf{x}_i - b \leq \varepsilon + \xi_i, \quad i = 1, l , \quad (47a)$$

$$\mathbf{w}^T \mathbf{x}_i + b - y_i \leq \varepsilon + \xi_i^*, \quad i = 1, l , \quad (47b)$$

$$\xi_i \geq 0, \xi_i^* \geq 0, \quad i = 1, l . \quad (47c)$$

where ξ_i and ξ_i^* are slack variables shown in Fig. 17 for measurements "above" and "below" an ε -tube respectively. Both slack variables are positive values. Lagrange multipliers α_i and α_i^* (that will be introduced during the minimization below) related to the first two sets of inequalities above, will be nonzero values for training points "above" and "below" an ε -tube respectively. Because no training data can be on both sides of the tube, either α_i or α_i^* will be nonzero. For data points inside the tube, both multipliers will be equal to zero. Thus $\alpha_i \alpha_i^* = 0$.

Note also that the constant C that influences a trade-off between an approximation error and the weight vector norm $\|\mathbf{w}\|$ is a design parameter that is chosen by the user. An increase in C penalizes larger errors i.e., it forces ξ_i and ξ_i^* to be small. This leads to an approximation error decrease which is achieved only by increasing the weight vector norm $\|\mathbf{w}\|$. However, an increase in $\|\mathbf{w}\|$ increases the confidence term Ω and does not guarantee a small generalization performance of a model. Another design parameter which is chosen by the user is the required precision embodied in an ε value that defines the size of an ε -tube. The choice of ε value is easier than the choice of C and it is given as either maximally allowed or some given or desired percentage of the output values y_i (say, $\varepsilon = 0.1$ of the mean value of \mathbf{y}).

Similar to procedures applied in the SV classifiers' design, we solve the constrained optimization problem above by forming a *primal variables Lagrangian* as follows,

$$\begin{aligned}
L_p(\mathbf{w}, b, \xi_i, \xi_i^*, \alpha_i, \alpha_i^*, \beta_i, \beta_i^*) &= \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^l (\xi_i + \xi_i^*) - \sum_{i=1}^l (\beta_i^* \xi_i^* + \beta_i \xi_i) \\
&\quad - \sum_{i=1}^l \alpha_i [\mathbf{w}^T \mathbf{x}_i + b - y_i + \varepsilon + \xi_i] \\
&\quad - \sum_{i=1}^l \alpha_i^* [y_i - \mathbf{w}^T \mathbf{x}_i, -b + \varepsilon + \xi_i^*]. \tag{48}
\end{aligned}$$

A primal variables Lagrangian $L_p(\mathbf{w}, b, \xi_i, \xi_i^*, \alpha_i, \alpha_i^*, \beta_i, \beta_i^*)$ has to be *minimized* with respect to primal variables \mathbf{w}, b, ξ_i and ξ_i^* and *maximized* with respect to nonnegative Lagrange multipliers $\alpha, \alpha_i^*, \beta$ and β_i^* . Hence, the function has the saddle point at the optimal solution $(\mathbf{w}_o, b_o, \xi_{io}, \xi_{io}^*)$ to the original problem. At the optimal solution the partial derivatives of L_p in respect to primal variables vanishes. Namely,

$$\frac{\partial L_p(\mathbf{w}_o, b_o, \xi_{io}, \xi_{io}^*, \alpha_i, \alpha_i^*, \beta_i, \beta_i^*)}{\partial \mathbf{w}} = \mathbf{w}_o - \sum_{i=1}^l (\alpha_i - \alpha_i^*) \mathbf{x}_i = 0, \tag{49}$$

$$\frac{\partial L_p(\mathbf{w}_o, b_o, \xi_{io}, \xi_{io}^*, \alpha_i, \alpha_i^*, \beta_i, \beta_i^*)}{\partial b} = \sum_{i=1}^l (\alpha_i - \alpha_i^*) = 0, \tag{50}$$

$$\frac{\partial L_p(\mathbf{w}_o, b_o, \xi_{io}, \xi_{io}^*, \alpha_i, \alpha_i^*, \beta_i, \beta_i^*)}{\partial \xi_i} = C - \alpha_i - \beta_i = 0, \tag{51}$$

$$\frac{\partial L_p(\mathbf{w}_o, b_o, \xi_{io}, \xi_{io}^*, \alpha_i, \alpha_i^*, \beta_i, \beta_i^*)}{\partial \xi_i^*} = C - \alpha_i^* - \beta_i^* = 0. \tag{52}$$

Substituting the KKT above into the primal L_p given in (48), we arrive at the problem of the *maximization of a dual variables Lagrangian* $L_d(\alpha, \alpha^*)$ below,

$$\begin{aligned}
L_d(\alpha_i, \alpha_i^*) &= -\frac{1}{2} \sum_{i,j=1}^l (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) \mathbf{x}_i^T \mathbf{x}_j - \varepsilon \sum_{i=1}^l (\alpha_i + \alpha_i^*) + \sum_{i=1}^l (\alpha_i - \alpha_i^*) y_i \\
&= -\frac{1}{2} \sum_{i,j=1}^l (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) \mathbf{x}_i^T \mathbf{x}_j - \sum_{i=1}^l (\varepsilon - y_i) \alpha_i - \sum_{i=1}^l (\varepsilon + y_i) \alpha_i^*. \tag{53}
\end{aligned}$$

subject to constraints

$$\sum_{i=1}^l \alpha_i^* = \sum_{i=1}^l \alpha_i \quad \text{or} \quad \sum_{i=1}^l (\alpha_i - \alpha_i^*) = 0, \tag{54a}$$

$$0 \leq \alpha_i \leq C \quad i = 1, l, \tag{54b}$$

$$0 \leq \alpha_i^* \leq C \quad i = 1, l. \tag{54c}$$

Note that the dual variables Lagrangian $L_d(\alpha, \alpha^*)$ is expressed in terms of Lagrange multipliers α_i and α_i^* only. However, the size of the problem, with

respect to the size of an SV classifier design task, is doubled now. There are $2l$ unknown dual variables ($l\alpha_i - s$ and $l\alpha_i^* - s$) for a linear regression and the Hessian matrix \mathbf{H} of the quadratic optimization problem in the case of regression is a $(2l, 2l)$ matrix. The *standard quadratic optimization problem* above can be expressed in a *matrix notation* and formulated as follows:

$$\text{minimize } L_d(\boldsymbol{\alpha}) = -0.5\boldsymbol{\alpha}^T \mathbf{H}\boldsymbol{\alpha} + \mathbf{f}^T \boldsymbol{\alpha}, \quad (55)$$

subject to (54) where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \dots, \alpha_l, \alpha_1^*, \alpha_2^*, \dots, \alpha_l^*]^T$, $\mathbf{H} = [\mathbf{G} \ -\mathbf{G}; \ -\mathbf{G} \ \mathbf{G}]$, \mathbf{G} is an (l, l) matrix with entries $G_{ij} = [\mathbf{x}_i^T \mathbf{x}_j]$ for a linear regression, and $\mathbf{f} = [\varepsilon - y_1, \varepsilon - y_2, \dots, \varepsilon - y_l, \varepsilon + y_1, \varepsilon + y_2, \dots, \varepsilon + y_l]^T$. (Note that G_{ij} , as given above, is a badly conditioned matrix and we rather use $G_{ij} = [\mathbf{x}_i^T \mathbf{x}_j + 1]$ instead). Again, (55) is written in a form of some standard optimization routine that typically *minimizes* given objective function subject to same constraints (54).

The learning stage results in l Lagrange multiplier *pairs* (α_i, α_i^*) . After the learning, the number of nonzero parameters α_i or α_i^* is equal to the number of SVs. However, this number does not depend on the dimensionality of input space and this is particularly important when working in very high dimensional spaces. Because at least one element of each pair (α_i, α_i^*) , $i = 1, l$, is zero, the product of α_i and α_i^* is always zero, i.e., $\alpha_i \alpha_i^* = 0$.

At the optimal solution the following *KKT complementarity conditions* must be fulfilled

$$\alpha_i(\mathbf{w}^T \mathbf{x}_i + b - y_i + \varepsilon + \xi_i) = 0, \quad (56)$$

$$\alpha_i^*(-\mathbf{w}^T \mathbf{x}_i - b + y_i + \varepsilon + \xi_i^*) = 0, \quad (57)$$

$$\beta_i \xi_i = (C - \alpha_i) \xi_i = 0, \quad (58)$$

$$\beta_i^* \xi_i^* = (C - \alpha_i^*) \xi_i^* = 0, \quad (59)$$

(58) states that for $0 < \alpha_i < C$, $\xi_i = 0$ holds. Similarly, from (59) follows that for $0 < \alpha_i^* < C$, $\xi_i^* = 0$ and, for $0 < \alpha_i, \alpha_i^* < C$, from (56) and (57) follows,

$$\mathbf{w}^T \mathbf{x}_i + b - y_i + \varepsilon = 0, \quad (60)$$

$$-\mathbf{w}^T \mathbf{x}_i - b + y_i + \varepsilon = 0. \quad (61)$$

Thus, for all the data points fulfilling $y - f(\mathbf{x}) = +\varepsilon$, dual variables α_i must be between 0 and C , or $0 < \alpha_i < C$, and for the ones satisfying $y - f(\mathbf{x}) = -\varepsilon$, α_i^* take on values $0 < \alpha_i^* < C$. These data points are called the *free* (or *unbounded*) support vectors. They allow computing the value of the bias term b as given below

$$b = y_i - \mathbf{w}^T \mathbf{x}_i - \varepsilon, \quad \text{for } 0 < \alpha_i < C, \quad (62a)$$

$$b = y_i - \mathbf{w}^T \mathbf{x}_i + \varepsilon, \quad \text{for } 0 < \alpha_i^* < C. \quad (62b)$$

The calculation of a bias term b is numerically very sensitive, and it is better to compute the bias b by averaging over all the *free* support vector data points.

The final observation follows from (58) and (59) and it tells that for all the data points outside the ε -tube, i.e., when $\xi_i > 0$ and $\xi_i^* > 0$, both α_i and α_i^* equal C , i.e., $\alpha_i = C$ for the points above the tube and $\alpha_i^* = C$ for the points below it. These data are the so-called *bounded support vectors*. Also, for all the training data points within the tube, or when $|y - f(\mathbf{x})| < \varepsilon$, both α_i and α_i^* equal zero and they are neither the support vectors nor do they construct the decision function $f(\mathbf{x})$.

After calculation of Lagrange multipliers α_i and α_i^* , using (49) we can find an *optimal* (desired) weight vector of the *regression hyperplane* as

$$\mathbf{w}_o = \sum_{i=1}^l (\alpha_i - \alpha_i^*) \mathbf{x}_i . \quad (63)$$

The best regression hyperplane obtained is given by

$$f(\mathbf{x}, \mathbf{w}) = \mathbf{w}_o^T \mathbf{x} + b = \sum_{i=1}^l (\alpha_i - \alpha_i^*) \mathbf{x}_i^T \mathbf{x} + b . \quad (64)$$

More interesting, more common and the most challenging problem is to aim at solving the *nonlinear regression tasks*. A generalization to nonlinear regression is performed in the same way the nonlinear classifier is developed from the linear one, i.e., by carrying the mapping to the feature space, or by using kernel functions instead of performing the complete mapping which is usually of extremely high (possibly of an infinite) dimension. Thus, the nonlinear regression function in an input space will be devised by considering a linear regression hyperplane in the *feature space*.

We use the same basic idea in designing SV machines for creating a *nonlinear regression function*. First, a mapping of input vectors $\mathbf{x} \in \Re^n$ into vectors $\Phi(\mathbf{x})$ of a higher dimensional *feature space* F (where Φ represents mapping: $\Re^n \rightarrow \Re^f$) takes place and then, we solve a linear regression problem in this feature space. A mapping $\Phi(\mathbf{x})$ is again the chosen in advance, or fixed, function. Note that an input space (\mathbf{x} -space) is spanned by components x_i of an input vector \mathbf{x} and a feature space F (Φ -space) is spanned by components $\phi_i(\mathbf{x})$ of a vector $\Phi(\mathbf{x})$. By performing such a mapping, we hope that in a Φ -space, our learning algorithm will be able to perform a linear regression hyperplane by applying the linear regression SVM formulation presented above. We also expect this approach to again lead to solving a quadratic optimization problem with inequality constraints in the feature space. The (linear in a feature space F) solution for the regression hyperplane $f = \mathbf{w}^T \Phi(\mathbf{x}) + b$, will create a nonlinear regressing hypersurface in the original input space. The most popular kernel functions are *polynomials* and *RBF* with *Gaussian kernels*. Both kernels are given in Table 2.

In the case of the nonlinear regression, the learning problem is again formulated as the maximization of a dual Lagrangian (55) with the Hessian matrix \mathbf{H} structured in the same way as in a linear case, i.e. $\mathbf{H} = [\mathbf{G} \ -\mathbf{G}; \ -\mathbf{G} \ \mathbf{G}]$ but with the changed Grammian matrix \mathbf{G} that is now given as

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_{11} & \cdots & \mathbf{G}_{1l} \\ \vdots & \mathbf{G}_{ii} & \vdots \\ \mathbf{G}_{l1} & \cdots & \mathbf{G}_{ll} \end{bmatrix}, \quad (65)$$

where the entries $G_{ij} = \Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$, $i, j = 1, l$.

After calculating Lagrange multiplier vectors α and α^* , we can find an optimal weighting vector of the *kernels expansion* as

$$\mathbf{v}_0 = \alpha - \alpha^*. \quad (66)$$

Note however the difference in respect to the linear regression where the expansion of a decision function is expressed by using the optimal weight vector \mathbf{w}_o . Here, in an NL SVMs' regression, the optimal weight vector \mathbf{w}_o could often be of infinite dimension (which is the case if the Gaussian kernel is used). Consequently, we neither calculate \mathbf{w}_o nor we have to express it in a closed form. Instead, we create the best nonlinear regression function by using the weighting vector \mathbf{v}_0 and the kernel (Grammian) matrix \mathbf{G} as follows,

$$f(\mathbf{x}, \mathbf{w}) = \mathbf{G}\mathbf{v}_0 + b. \quad (67)$$

In fact, the last result follows from the very setting of the learning (optimizing) stage in a feature space where, in all the equations above from (47) to (64), we replace \mathbf{x}_i by the corresponding feature vector $\Phi(\mathbf{x}_i)$. This leads to the following changes:

- instead $G_{ij} = \mathbf{x}_i^T \mathbf{x}_j$ we get $G_{ij} = \Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j)$ and, by using the kernel function $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j)$, it follows that $G_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$.
- Similarly, (63) and (64) change as follows:

$$\mathbf{w}_o = \sum_{i=1}^l (\alpha_i - \alpha_i^*)\Phi(\mathbf{x}_i), \quad (68)$$

$$\begin{aligned} f(\mathbf{x}, \mathbf{w}) &= \mathbf{w}_o^T \Phi(\mathbf{x}) + b = \sum_{i=1}^l (\alpha_i - \alpha_i^*)\Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}) + b \\ &= \sum_{i=1}^l (\alpha_i - \alpha_i^*)K(\mathbf{x}_i, \mathbf{x}) + b. \end{aligned} \quad (69)$$

If the bias term b is explicitly used as in (67) then, for an NL SVMs' regression, it can be calculated from the upper SVs as,

$$\begin{aligned} b &= y_i - \sum_{j=1}^{N \text{ freeupper SVs}} (\alpha_j - \alpha_j^*)\Phi^T(\mathbf{x}_j)\Phi(\mathbf{x}_i) - \varepsilon \\ &= y_i - \sum_{j=1}^{N \text{ freeupper SVs}} (\alpha_j - \alpha_j^*)K(\mathbf{x}_i, \mathbf{x}_j) - \varepsilon \end{aligned}, \quad \text{for } 0 < \alpha_i < C, \quad (70a)$$

or from the lower ones as,

$$\begin{aligned}
 b &= y_i - \sum_{j=1}^{N \text{ free lower SVs}} (\alpha_j - \alpha_j^*) \Phi^T(\mathbf{x}_j) \Phi(\mathbf{x}_i) + \varepsilon \\
 &= y_i - \sum_{j=1}^{N \text{ free lower SVs}} (\alpha_j - \alpha_j^*) K(\mathbf{x}_i, \mathbf{x}_j) + \varepsilon
 \end{aligned} \quad , \quad \text{for } 0 < \alpha_i^* < C . \quad (70b)$$

Note that $\alpha_j^* = 0$ in (70a) and so is $\alpha_j = 0$ in (70b). Again, it is much better to calculate the bias term b by an averaging over all the free support vector data points.

There are a few learning parameters in constructing SV machines for regression. The three most relevant are the insensitivity zone ε , the penalty parameter C (that determines the trade-off between the training error and VC dimension of the model), and the shape parameters of the kernel function (variances of a Gaussian kernel, order of the polynomial, or the shape parameters of the inverse multiquadratics kernel function). All three parameters' sets should be selected by the user. To this end, the most popular method is a cross-validation. Unlike in a classification, for not too noisy data (primarily without huge outliers), the penalty parameter C could be set to infinity and the modeling can be controlled by changing the insensitivity zone ε and shape parameters only.

The example below shows how an increase in an insensitivity zone ε has smoothing effects on modeling highly noise polluted data. Increase in ε means a reduction in requirements on the accuracy of approximation. It decreases the number of SVs leading to higher data compression too. This can be readily followed in the lines and Fig. 19 below.

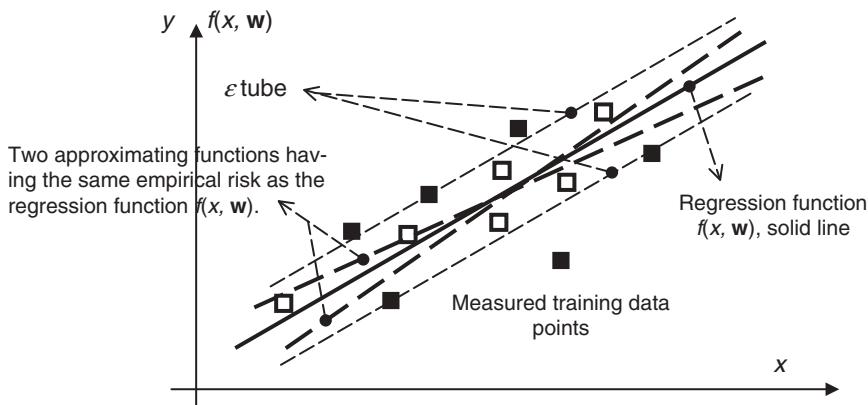


Fig. 19. Two linear approximations inside an ε tube (dashed lines) have the same empirical risk $R_{\text{emp}}^\varepsilon$ on the training data as the regression function (solid line)

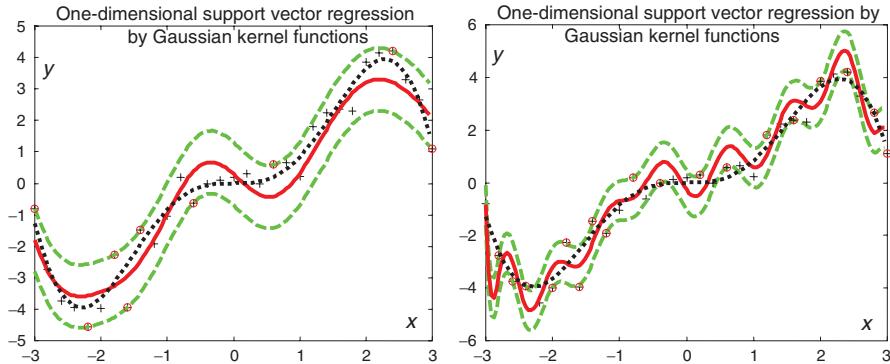


Fig. 20. The influence of an insensitivity zone ε on the model performance. A nonlinear SVM creates a regression function f with Gaussian kernels and models a highly polluted (25% noise) function $x^2 \sin(x)$ (dotted). 31 training data points (plus signs) are used. *Left:* $\varepsilon = 1$; 9 SVs are chosen (*encircled plus signs*). *Right:* $\varepsilon = 0.75$; the 18 chosen SVs produced a better approximation to noisy data and, consequently, there is the tendency of overfitting

Example: The task here is to construct an SV machine for modeling measured data pairs. The underlying function (known to us but, not to the SVM) is a sinus function multiplied by the square one (i.e., $f(x) = x^2 \sin(x)$) and it is corrupted by 25% of normally distributed noise with a zero mean. Analyze the influence of an insensitivity zone ε on modeling quality and on a compression of data, meaning on the number of SVs.

Figure (19) shows that for a very noisy data a decrease of an insensitivity zone ε (i.e., shrinking of the tube shown by dashed line) approximates the noisy data points more closely. The related more and more wiggly shape of the regression function can be achieved only by including more and more support vectors. However, being good on the noisy training data points easily leads to an overfitting. The cross-validation should help in finding correct ε value, resulting in a regression function that filters the noise out but not the true dependency and which, consequently, approximate the underlying function as close as possible.

The approximation function shown in Fig. 20 is created by 9 and 18 weighted Gaussian basis functions for $\varepsilon = 1$ and $\varepsilon = 0.75$ respectively. These supporting functions are not shown in the figure. However, the way how the learning algorithm selects SVs is an interesting property of support vector machines and in Fig. 21 we also present the supporting Gaussian functions.

Note that the selected Gaussians lie in the dynamic area of the function in Fig. 21. Here, these areas are close to both the left hand and the right hand boundary. In the middle, the original function is pretty flat and there is no need to cover this part by supporting Gaussians. The learning algorithm realizes this fact and simply, it does not select any training data point in this area as a support vector. Note also that the Gaussians are not weighted

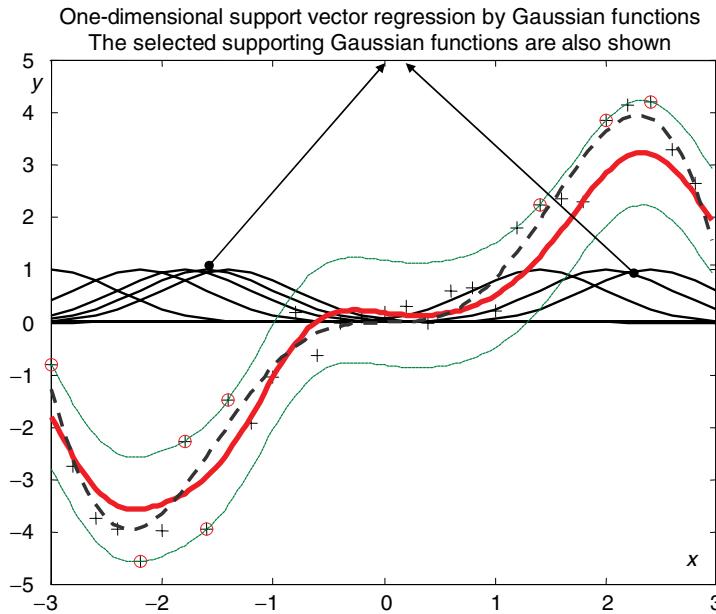


Fig. 21. Regression function f created as the sum of 8 weighted Gaussian kernels. A standard deviation of Gaussian bells $\sigma = 0.6$. Original function (dashed line) is $x^2 \sin(x)$ and it is corrupted by .25% noise. 31 training data points are shown as plus signs. Data points selected as the SVs are encircled. The 8 selected supporting Gaussian functions are centered at these data points

in Fig. 21, and they all have the peak value of 1. The standard deviation of Gaussians is chosen in order to see Gaussian supporting functions better. Here, in Fig. 21, $\sigma = 0.6$. Such a choice is due the fact that for the larger σ values the basis functions are rather flat and the supporting functions are covering the whole domain as the broad umbrellas. For very big variances one can't distinguish them visually. Hence, one can't see the true, bell shaped, basis functions for the large variances.

3 Implementation Issues

In both the classification and the regression the learning problem boils down to solving the QP problem subject to the so-called “box-constraints and to the equality constraint in the case that a model with a bias term b is used. The SV training works almost perfectly for not too large data basis. However, when the number of data points is large (say $l > 2,000$) the QP problem becomes extremely difficult to solve with standard QP solvers and methods. For example, a classification training set of 50,000 examples amounts to a Hessian matrix \mathbf{H} with 2.5×10^9 (2.5 billion) elements. Using an 8-byte floating-point

representation we need $20,000 \text{ Megabytes} = 20 \text{ Gigabytes}$ of memory [20]. This cannot be easily fit into memory of present standard computers, and this is the single basic disadvantage of the SVM method. There are three approaches that resolve the QP for large data sets. Vapnik in [31] proposed the *chunking method* that is the decomposition approach. Another *decomposition* approach is suggested in [20]. The sequential minimal optimization [21] algorithm is of different character and it seems to be an “error back propagation” for an SVM learning. A systematic exposition of these various techniques is not given here, as all three would require a lot of space. However, the interested reader can find a description and discussion about the algorithms mentioned above in [17, 34]. The Vogt and Kecman’s chapter discusses the application of an *active set* algorithm in solving small to medium sized QP problems. For such data sets and when the high precision is required the active set approach in solving QP problems seems to be superior to other approaches (notably the interior point methods and SMO algorithm). The Kecman, Huang, and Vogt’s chapter introduces the efficient *iterative single data algorithm (ISDA)* for solving huge data sets (say more than 100,000 or 500,000 or over 1 million training data pairs). It seems that ISDA is the fastest algorithm at the moment for such large data sets still ensuring the convergence to the global minimum (see the comparisons with SMO in [17]). This means that the ISDA provides the exact, and not the approximate, solution to original dual problem.

Let us conclude the presentation of SVMs part by summarizing the basic constructive steps that lead to the SV machine.

A training and design of a support vector machine is an *iterative* algorithm and it involves the following steps:

- (a) define your problem as the classification or as the regression one,
- (b) preprocess your input data: select the most relevant features, scale the data between $[-1, 1]$, or to the ones having zero mean and variances equal to one, check for possible outliers (strange data points),
- (c) select the kernel function that determines the hypothesis space of the decision and regression function in the classification and regression problems respectively,
- (d) select the “shape”, i.e., “smoothing” parameter of the kernel function (for example, polynomial degree for polynomials and variances of the Gaussian RBF kernels respectively),
- (e) choose the penalty factor C and, in the regression, select the desired accuracy by defining the insensitivity zone ε too,
- (f) solve the QP problem in l and $2l$ variables in the case of classification and regression problems respectively,
- (g) validate the model obtained on some previously, during the training, unseen test data, and if not pleased iterate between steps d (or, eventually c) and g.

The optimizing part f) is computationally extremely demanding. First, the Hessian matrix \mathbf{H} scales with the size of a data set – it is an (l, l) and an $(2l, 2l)$ matrix in classification and regression respectively. Second, unlike in classic original QP problems \mathbf{H} is very dense matrix and it is usually badly conditioned requiring a regularization before any numeric operation. Regularization means an addition of a small number to the diagonal elements of \mathbf{H} . Luckily, there are many reliable and fast QP solvers. A simple search on an internet will reveal many of them. Particularly, in addition to the classic ones such as MINOS or LOQO for example, there are many more free QP solvers designed specially for the SVMs. The most popular ones are – the LIBSVM, SVMlight, SVM Torch, mySVM and SVM Fu. All of them can be downloaded from their corresponding sites. Good educational software in matlab named LEARNSC, with a very good graphic presentations of all relevant objects in a SVM modeling, can be downloaded from the author's book site www.support-vector.ws too.

Finally we mention that there are many alternative formulations and approaches to the QP based SVMs described above. Notably, they are the linear programming SVMs [10, 13, 14, 15, 16, 18, 25], v -SVMs [23] and least squares support vector machines [27]. Their description is far beyond this chapter and the curious readers are referred to references given above.

Appendix

L2 Support Vector Machines Models Derivation

While introducing the soft SVMs by allowing some unavoidable errors and, at the same time, while trying to minimize the distances of the erroneous data points to the margin, or to the tube in the regression problems, we have augmented the cost $0.5\mathbf{w}^T\mathbf{w}$ by the term $\sum_{i=1}^l (\xi_i^k + \xi_i^{*k})$ as the measure of these distances. Obviously, by using $k = 2$ we are punishing more strongly the far away points, than by using $k = 1$. There is a natural question then – what choice might be better in application. The experimental results [1] as well as the theoretically oriented papers [3, 26] point to the two interesting characteristics of the L1 and L2 SVMs. At this point, it is hard to say about some particular advantages. By far, L1 is more popular and used model. It seems that this is a consequence of the fact that L1 SVM produces sparser models (less SVs for a given data). Sparseness is but one of the nice properties of kernel machines. The other nice property is a performance on a real data set and a capacity of SVMs to provide good estimation of either unknown decision functions or the regression ones. In classification, we talk about the possibility to estimate the conditional probability of the class label. For this task, it seems that the L2 SVMs may be better. A general facts are that the L1-SVMs can be expected to produce sparse solutions and that L2-SVMs will typically not produce sparse solutions, but may be better in estimating

conditional probabilities. Thus, it may be interesting to investigate the relationship between these two properties. Two nice theoretical papers discussing the issues of sparseness and its trade-off for a good prediction performance are mentioned above. We can't go into these subtleties here. Instead, we provide to the reader the derivation of the L2 SVMs model, and we hope the models presented here may help the reader in his/hers own search for better SVMs model.

Below we present the derivation of the L2 soft NL classifier given by (32) and (33a) following by the derivation of the L2 soft NL regressor. Both derivations are performed in the feature space F . Thus the input vector to the SVM is the $\Phi(\mathbf{x})$ vector. All the results are valid for a linear model too (where we work in the original input space) by replacing $\Phi(\mathbf{x})$ by \mathbf{x} .

L2 Soft Margin Classifier

Now, we start from the (24) but instead of a linear distance ξ_i we work with a quadratic one ξ_i^2 . Thus the task is to

$$\text{minimize} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + \frac{C}{2} \sum_{i=1}^l \xi_i^2, \quad (\text{A24a})$$

subject to

$$y_i [\mathbf{w}^T \Phi(\mathbf{x}_i) + b] \geq 1 - \xi_i, \quad i = 1, l, \xi_i \geq 0, \quad (\text{A24b})$$

i.e., subject to

$$\mathbf{w}^T \mathbf{x}_i + b \geq +1 - \xi_i, \quad \text{for } y_i = +1, \xi_i \geq 0, \quad (\text{A24c})$$

$$\mathbf{w}^T \mathbf{x}_i + b \leq -1 + \xi_i, \quad \text{for } y_i = -1, \xi_i \geq 0 \quad (\text{A24d})$$

Now, both the \mathbf{w} and the $\Phi(\mathbf{x})$ are the f -dimensional vectors. Note that the dimensionality f can also be infinite and this happens very often (e.g., when the Gaussian kernels are used). Again, the solution to the quadratic programming problem (A24) is given by the saddle point of the primal Lagrangian $L_p(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha})$, shown below

$$L_p(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + \frac{C}{2} \left(\sum_{i=1}^l \xi_i^2 \right) - \sum_{i=1}^l \alpha_i \{ y_i [\mathbf{w}^T \mathbf{x}_i + b] - 1 + \xi_i \}, \quad (\text{A25})$$

Note that the Lagrange multiplier β associated with $\boldsymbol{\xi}$ is missing here. It vanishes by combining (29b) and (28) which is now equal to $\alpha_i + \beta_i = C\xi_i$. Again, we should find an *optimal* saddle point $(\mathbf{w}_o, b_o, \boldsymbol{\xi}_o, \boldsymbol{\alpha}_o)$ because the Lagrangian L_p has to be *minimized* with respect to \mathbf{w} , b and $\boldsymbol{\xi}$, and *maximized* with respect to nonnegative α_i . And yet again, we consider a solution in a dual space as given below by using

- standard conditions for an optimum of a constrained function

$$\frac{\partial L}{\partial \mathbf{w}_o} = 0, \text{ i.e., } \quad \mathbf{w}_o = \sum_{i=1}^l \alpha_i y_i \Phi(\mathbf{x}_i), \quad (\text{A26})$$

$$\frac{\partial L}{\partial b_o} = 0, \text{ i.e., } \quad \sum_{i=1}^l \alpha_i y_i = 0, \quad (\text{A27})$$

$$\frac{\partial L}{\partial \xi_{io}} = 0, \text{ i.e., } \quad C\xi_i - \alpha_i = 0, \quad (\text{A28})$$

– and the KKT complementarity conditions below,

$$\begin{aligned} \alpha_{io}\{y_i[\mathbf{w}^T \Phi(\mathbf{x}_i) + b] - 1 + \xi_i\} &= 0, \text{ i.e.,} \\ \alpha_{io}\{y_i \left[\sum_{j=1}^l \alpha_j y_j k(\mathbf{x}_j, \mathbf{x}_i) + b_o \right] - 1 + \xi_i\} &= 0 \quad i = 1, l. \end{aligned} \quad (\text{A29})$$

A substitution of (A26) and (A28) into the L_p leads to the search for a maximum of a *dual Lagrangian*

$$L_d(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l y_i y_j \alpha_i \alpha_j \left(k(\mathbf{x}_i, \mathbf{x}_j) + \frac{\delta_{ij}}{C} \right), \quad (\text{A30})$$

subject to

$$\alpha_i \geq 0, i = 1, l, \quad (\text{A31a})$$

and under the equality constraints

$$\sum_{i=1}^l \alpha_i y_i = 0, \quad (\text{A31b})$$

where, $\delta_{ij} = 1$ for $i = j$, and it is zero otherwise. There are three tiny differences in respect to the most standard L1 SVM. First, in a Hessian matrix, a term $1/C$ is added to its diagonal elements which ensures positive definiteness of \mathbf{H} and stabilizes the solution. Second, there is no upper bound on α_i and the only requirement is α_i to be non-negative. Third, there are no longer complementarity constraints (29b), $(C - \alpha_i)\xi_i = 0$.

L2 Soft Regressor

An entirely similar procedure leads to the soft L2 SVM regressors. We start from the reformulated (46) as given below

$$R_{\mathbf{w}, \xi, \xi^*} = \left[\frac{1}{2} \|\mathbf{w}\|^2 + C \left(\sum_{i=1}^l \xi_i^2 + \sum_{i=1}^l \xi_i^{*2} \right) \right], \quad (\text{A46})$$

and after an introduction of the Lagrange multipliers α_i or α_i^* we change to the unconstrained primal Lagrangian L_p as given below,

$$\begin{aligned}
L_p(\mathbf{w}, b, \xi_i, \xi_i^*, \alpha_i, \alpha_i^*) &= \frac{1}{2} \mathbf{w}^T \mathbf{w} + \frac{C}{2} \sum_{i=1}^l (\xi_i^2 + \xi_i^{*2}) \\
&\quad - \sum_{i=1}^l \alpha_i [\mathbf{w}^T \mathbf{x}_i + b - y_i + \varepsilon + \xi_i] \\
&\quad - \sum_{i=1}^l \alpha_i^* [y_i - \mathbf{w}^T \mathbf{x}_i - b + \varepsilon + \xi_i^*] . \quad (\text{A47})
\end{aligned}$$

Again, the introduction of the dual variables β and β_i^* associated with ξ_i and ξ_i^* is not needed for the L2 SVM regression models. At the optimal solution the partial derivatives of L_p in respect to primal variables vanish. Namely,

$$\frac{\partial L_p(\mathbf{w}_o, b_o, \xi_{io}, \xi_{io}^*, \alpha_i, \alpha_i^*)}{\partial \mathbf{w}} = \mathbf{w}_o - \sum_{i=1}^l (\alpha_i - \alpha_i^*) \Phi(\mathbf{x}_i) = 0 , \quad (\text{A48})$$

$$\frac{\partial L_p(\mathbf{w}_o, b_o, \xi_{io}, \xi_{io}^*, \alpha_i, \alpha_i^*)}{\partial b} = \sum_{i=1}^l (\alpha_i - \alpha_i^*) = 0 , \quad (\text{A49})$$

$$\frac{\partial L_p(\mathbf{w}_o, b_o, \xi_{io}, \xi_{io}^*, \alpha_i, \alpha_i^*)}{\partial \xi_i} = C \xi_i - \alpha_i = 0 , \quad (\text{A50})$$

$$\frac{\partial L_p(\mathbf{w}_o, b_o, \xi_{io}, \xi_{io}^*, \alpha_i, \alpha_i^*)}{\partial \xi_i^*} = C \xi_i^* - \alpha_i^* = 0 . \quad (\text{A51})$$

Substituting the KKT above into the primal L_p given in (A47), we arrive at the problem of the *maximization of a dual variables Lagrangian* $L_d(\alpha, \alpha^*)$ below,

$$\begin{aligned}
L_d(\alpha_i, \alpha_i^*) &= -\frac{1}{2} \sum_{i,j=1}^l (\alpha_j - \alpha_j^*)(\alpha_j - \alpha_j^*) \left(\Phi^T(\mathbf{x}_i) \Phi(\mathbf{x}_j) + \frac{\delta_{ij}}{C} \right) \\
&\quad - \varepsilon \sum_{i=1}^l (\alpha_i + \alpha_i^*) \sum_{i=1}^l (\alpha_i - \alpha_i^*) y_i \\
&= -\frac{1}{2} \sum_{i,j=1}^l (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) \left(k(\mathbf{x}_i, \mathbf{x}_j) + \frac{\delta_{ij}}{C} \right) \\
&\quad - \sum_{i=1}^l (\varepsilon - y_i) \alpha_i - \sum_{i=1}^l (\varepsilon - y_i) \alpha_i^* \quad (\text{A52})
\end{aligned}$$

subject to constraints

$$\sum_{i=1}^l \alpha_i^* = \sum_{i=1}^l \alpha_i \quad \text{or} \quad \sum_{i=1}^l (\alpha_i - \alpha_i^*) = 0 \quad (\text{A53a})$$

$$0 \leq \alpha_i \quad i = 1, l , \quad (\text{A53b})$$

$$0 \leq \alpha_i^* \quad i = 1, l . \quad (\text{A53c})$$

At the optimal solution the following *KKT complementarity conditions* must be fulfilled

$$\alpha_i (\mathbf{w}^T \mathbf{x}_i + b - y_i + \varepsilon + \xi_i) = 0 , \quad (\text{A54})$$

$$\alpha_i^* (-\mathbf{w}^T \mathbf{x}_i - b + y_i + \varepsilon + \xi_i^*) = 0 , \quad (\text{A55})$$

$$\alpha_i \alpha_i^* = 0, \quad \xi_i \xi_i^* = 0 \quad i = 1, l . \quad (\text{A56})$$

Note that for the L2 SVM regression models the complementarity conditions (58) and (59) are eliminated here. After the calculation of Lagrange multipliers α_i and α_i^* , and by using (A48) we can find an *optimal* (desired) weight vector of the *L2 regression hyperplane* in a feature space as

$$\mathbf{w}_o = \sum_{i=1}^l (\alpha_i - \alpha_i^*) \Phi(\mathbf{x}_i) . \quad (\text{A57})$$

The best L2 regression hyperplane obtained is given by

$$F(\mathbf{x}, \mathbf{w}) = \mathbf{w}_o^T \Phi(\mathbf{x}) + b = \sum_{i=1}^l (\alpha_i - \alpha_i^*) k(\mathbf{x}_i, \mathbf{x}) + b . \quad (\text{A58})$$

Same as for the L1 SVM classifiers, there are three tiny differences in respect to the most standard L1 SVM regressors. First, in a Hessian matrix, a term $1/C$ is added to its diagonal elements which ensures positive definiteness of \mathbf{H} and stabilizes the solution. Second, there is no upper bound on α_i and the only requirement is α_i to be non-negative. Third, there are no longer complementarity constraints (58) and (59), namely the conditions $(C - \alpha_i)\xi_i = 0$ and $(C - \alpha_i^*)\xi_i^* = 0$ are missing in the L2 SVM regressors.

Finally, same as for the L1 SVMs, note that the NL decision functions here depend neither upon \mathbf{w} nor on the true mapping $\Phi(\mathbf{x})$. The last remark is same for all NL SVMs models shown here and it reminds that we *neither have to express, nor to know* the weight vector \mathbf{w} and the true mapping $\Phi(\mathbf{x})$ et al. The complete data modeling job will be done by finding the dual variables $\alpha_i^{(*)}$ and the kernel values $k(\mathbf{x}_i; \mathbf{x}_j)$ only.

With these remarks we left the SVMs models developed and presented in the chapter to both the mind and the hand of a curious reader. However, we are aware that the most promising situation would be if the kernel models reside in the heart of the reader. We wish and hope that this booklet paved, at least, a part of the way to this veiled place.

References

1. Abe, S., Support Vector Machines for Pattern Classification" (in print), Springer-Verlag, London, 2004 [73](#), [96](#)
2. Aizerman, M.A., E.M. Braverman, and L.I. Rozonoer, 1964. Theoretical foundations of the potential function method in pattern recognition learning, Automation and Remote Control 25, 821–837. [77](#)
3. Bartlett, P.L., A. Tewari, 2004, Sparseness vs estimating conditional probabilities: Some asymptotic results. (submitted for a publication and taken from the P.L. Bartlett's site) [96](#)
4. Cherkassky, V., F. Mulier, 1998. Learning From Data: Concepts, Theory and Methods, John Wiley & Sons, New York, NY [49](#), [57](#), [58](#)
5. Cortes, C., 1995. Prediction of Generalization Ability in Learning Machines. PhD Thesis, Department of Computer Science, University of Rochester, NY [70](#)
6. Cortes, C., Vapnik, V. 1995. Support Vector Networks. Machine Learning 20:273–297 [70](#)
7. Cristianini, N., Shawe-Taylor, J., 2000, An introduction to Support Vector Machines and other kernel-based learning methods, Cambridge University Press, Cambridge, UK
8. Drucker, H., C.J.C. Burges, L. Kaufman, A. Smola, V. Vapnik. 1997. Support vector regression machines, Advances in Neural Information Processing Systems 9, 155–161, MIT Press, Cambridge, MA [85](#)
9. Eisenhart, C., 1962. Roger Joseph Boscovich and the Combination of Observations, Actes International Symposium on R.J. Boskovic, pp. 19–25, Belgrade – Zagreb - Ljubljana, YU [85](#)
10. Frieß, T, R.F. Harrison, 1998, Linear programming support vectors machines for pattern classification and regression estimation and the set reduction algorithm, TR RR-706, University of Sheffield, Sheffield, UK [96](#)
11. Girosi, F., 1997. An Equivalence Between Sparse Approximation and Support Vector Machines, AI Memo 1606, MIT [52](#)
12. Graepel, T., R. Herbrich, B. Schölkopf, A. Smola, P. Bartlett, K.-R. Müller, K. Obermayer, R. Williamson, 1999, Classification on proximity data with LP-machines, Proc. of the 9th Intl. Conf. on Artificial NN, ICANN 99, Edinburgh, 7–10 Sept.
13. Hadžić, I., V. Kecman, 1999, Learning from Data by Linear Programming, NZ Postgraduate Conference Proceedings, Auckland, Dec. 15–16 [96](#)
14. Kecman, V., Arthanari T., Hadžić I, 2001, LP and QP Based Learning From Empirical Data, IEEE Proceedings of IJCNN 2001, Vol 4., pp. 2451–2455, Washington, DC [96](#)
15. Kecman, V., 2001, Learning and Soft Computing, Support Vector machines, Neural Networks and Fuzzy Logic Models, The MIT Press, Cambridge, MA, the book's web site is: <http://www.support-vector.ws> [49](#), [63](#), [67](#), [77](#), [96](#)
16. Kecman, V., Hadžić I., 2000, *Support Vectors Selection by Linear Programming*, Proceedings of the International Joint Conference on Neural Networks (IJCNN 2000), Vol. 5, pp. 193–198, Como, Italy [96](#)
17. Kecman, V., T.M. Huang, M. Vogt, 2005, Chapter “Iterative Single Data Algorithm for Training Kernel Machines from Huge Data Sets: Theory and Performance”, in a Springer-Verlag book, “Support Vector Machines: Theory and Applications”, Ed. Lipo Wang [80](#), [83](#), [95](#)

18. Mangasarian, O.L., 1965, Linear and Nonlinear Separation of Patterns by Linear Programming, *Operations Research* 13, pp. 444–452 [96](#)
19. Mercer, J., 1909. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London, A* 209:415{446} [77](#)
20. Osuna, E., R. Freund, F. Girosi. 1997. Support vector machines: Training and applications. AI Memo 1602, Massachusetts Institute of Technology, Cambridge, MA [95](#)
21. Platt, J.C. 1998. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research, 1998 [95](#)
22. Poggio, T., Mukherjee, S., Rifkin, R., Rakhlin, A., Verri, A., b,CBCL Paper #198/AI Memo# 2001-011, Massachusetts Institute of Technology, Cambridge, MA, 2001
23. Schölkopf, B., Smola, A., Learning with Kernels – Support Vector Machines, Optimization, and Beyond, The MIT Press, Cambridge, MA, 2002 [49](#), [67](#), [96](#)
24. Smola, A., Schölkopf, B. 1997. On a Kernel-based Method for Pattern Recognition, Regression, Approximation and Operator Inversion. GMD Technical Report No. 1064, Berlin [77](#)
25. Smola, A., T.T. Friess, B. Schölkopf, 1998, Semiparametric Support Vector and Linear Programming Machines, NeuroCOLT2 Technical Report Series, NC2-TR-1998-024, also in *In Advances in Neural Information Processing Systems 11*, 1998 [96](#)
26. Steinwart, I., 2003, Sparseness of support vector machines. *Journal of Machine Learning Research* 4 (2003), pp.1071–1105 Support Vector Machines Web Site: <http://www.kernel-machines.org/> [96](#)
27. Suykens, J.A.K., T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, 2002, Least Squares Support Vector Machines, World Scientific Pub. Co., Singapore [96](#)
28. Vapnik, V.N., A.Y. Chervonenkis, 1968. On the uniform convergence of relative frequencies of events to their probabilities. (In Russian), Doklady Akademii Nauk USSR, 181, (4) [50](#), [56](#), [57](#)
29. Vapnik, V. 1979. Estimation of Dependences Based on Empirical Data [in Russian]. Nauka, Moscow. (English translation: 1982, Springer Verlag, New York)
30. Vapnik, V.N., A.Y. Chervonenkis, 1989. The necessary and sufficient condititons for the consistency of the method of empirical minimization [in Russian], yearbook of the Academy of Sciences of the USSR on Recognition, Classification, and Forecasting, 2, 217–249, Moscow, Nauka, (English transl.: The necessary and sufficient condititons for the consistency of the method of empirical minimization. *Pattern Recognitio and Image Analysis*, 1, 284–305, 1991) [57](#), [67](#)
31. Vapnik, V.N., 1995. The Nature of Statistical Learning Theory, Springer Verlag Inc, New York, NY [49](#), [51](#), [57](#), [67](#), [95](#)
32. Vapnik, V., S. Golowich, A. Smola. 1997. Support vector method for function approximation, regression estimation, and signal processing, In *Advances in Neural Information Processing Systems* 9, MIT Press, Cambridge, MA [85](#)

33. Vapnik, V.N., 1998. Statistical Learning Theory, J.Wiley & Sons, Inc., New York, NY **49, 51, 77**
34. Vogt, M., V. Kecman, 2005, Chapter “Active-Set Methods for Support Vector Machines’, in a Springer-Verlag book, “Support Vector Machines: Theory and Applications’, Ed. Lipo Wang **80, 95**

Supporting Deep Learning in an Open-ended Domain

A. Weerasinghe and A. Mitrovic

Abstract. Self-explanation has been used successfully in teaching Mathematics and Physics to facilitate deep learning. We are interested in investigating whether self-explanation can be used in an open-ended, ill-structured domain. For this purpose, we enhanced KERMIT, an intelligent tutoring system that teaches conceptual database design. The resulting system, KERMIT-SE, supports self-explanation by engaging students in tutorial dialogues when their solutions are erroneous. The results of an evaluation study indicate that self-explanation leads to improved performance in both conceptual and procedural knowledge.

1 Introduction

Many Intelligent Tutoring Systems (ITS) have shown significant learning gains for students [24] particularly in the domain of Mathematics [14], Physics [16] and Computer Science [18, 23]. However, some empirical studies indicate that students acquire shallow knowledge even in the most effective systems [1]. As a result, students have difficulties in transferring knowledge to novel situations, even though they obtain passing grades on tests. Researchers are therefore interested in finding methods that overcome the shallow learning problem. Self-explanation is described as an “*activity of explaining to oneself in an attempt to make sense of new information, either presented in a text or in some other medium*” [9], and has been shown to facilitate the acquisition of deep knowledge [10]. Since explaining instructional material to oneself facilitates the integration of new information into existing knowledge, self-explanation can be viewed as a knowledge construction activity [9, 31]. According to the Chi’s study [9] self-explanation facilitates the process of repairing one’s own mental model of the domain, promoting reflection, a metacognitive skill lacked by many students [20]. Self-explanation studies also show that the majority of students do not spontaneously self-explain. However, students begin self-explaining more frequently when they are guided [7] or even simply prompted to do so [11]. These results suggest that it may be greatly

beneficial to integrate computer-based support to problem solving through self-explanation.

KERMIT (Knowledge-based Entity Relationship Modelling Intelligent Tutor) [28, 29] is a problem-solving environment that supports students learning database (DB) modelling. In this paper, we describe how we enhanced KERMIT to support self-explanation. Section 2 describes related work. KERMIT is briefly introduced in Sect. 3 and KERMIT-SE, its enhancement that facilitates self-explanation, is presented in the next section. The results of the evaluation study are presented in Sect. 5. The conclusions and directions for future research are given in the final section.

2 Related Work

Self-explanation has been used in several ITSs to enhance learning. These systems are discussed in this section.

Self-explanation with SE-Coach

SE-Coach supports self-explanation by prompting students to explain solved examples [13]. It is implemented within ANDES [16], a tutoring system that teaches Newtonian Physics. The first level of scaffolding in the SE-Coach's interface is provided by a masking mechanism that covers different parts of the example with grey boxes, each corresponding to a unit of information. When the student moves the mouse over a box, it disappears, revealing the content underneath. The second level of scaffolding produces specific prompts to self-explain. Students are prompted to self-explain only when the tutor decides it is beneficial. To determine when to intervene, SE-Coach relies on a probabilistic student model, that monitors how well the student understands the domain by capturing both implicit self-explanations and self-explanations generated through the interface [12]. The results of the empirical evaluation of SE-Coach reveal that the structured scaffolding of self-explanation can be more beneficial in the early learning stages.

ANDES has been further enhanced by incorporating ATLAS [30], a module to conduct self-explanation in a natural language. When ATLAS recognizes an opportunity to encourage deep learning, it initiates a natural language dialogue with the student. The main objective of these dialogues is to facilitate knowledge construction; hence, the dialogues are known as knowledge construction dialogues (KCDs). KCDs provided by ATLAS are currently limited to teaching domain principles. A limited study (with ten participants) revealed that the students interacting with ATLAS learnt significantly more than students who interacted with ANDES [17]. However, larger studies are needed to obtain more reliable results.

PACT Geometry Tutor

Aleven and Koedinger [1] investigated self-explanation in the PACT Geometry Tutor. The students in the experimental group were expected to provide correct explanations for solution steps by citing definitions and theorems used. A glossary of knowledge in the form of definitions and theorems was provided in order to help students to explain the solution steps. The study revealed that explaining reasoning steps results in improved problem solving skills. Students who explained solution steps attempted significantly fewer problems than their peers who provided only the answers, although both groups spent the same amount of time with the tutor. However, there was evidence that self-explainers required fewer problems to reach the tutor's mastery level criterion.

PACT Geometry Tutor has been further enhanced to facilitate self-explanation through natural language dialogue [3]. In Geometry Explanation Tutor, students explain in natural language, and the system evaluates their explanations and provides feedback. The system contains a hierarchy of 149 explanation categories [4], which is a library of common explanations, including incorrect/incomplete ones. The system matches the student's explanation to those in the library, and generates feedback, which helps the student to improve his/her explanation. An empirical study was carried out to investigate whether self-explanation facilitated through natural language enhances learning better than self-explanation through menu selection [2]. The results revealed that students who explained in natural language did not learn better than those who self-explained through menu selection.

AUTOTUTOR

AUTOTUTOR [18] is used in an introductory course in computer literacy. The system improved the students' learning by 0.5 standard deviation units when compared with a control group of students who read the same chapters from a book. AUTOTUTOR requires students to provide lengthy explanations for the *How*, *Why* and *What-if* type of questions. This approach encourages students to articulate lengthier answers that exhibit deeper reasoning instead of short answers, which may lead to shallow knowledge. A continuous multi-turn dialogue between the tutor and the student takes place throughout the session. The tutor attempts to understand the student input by segmenting contributions into speech acts and matching them with the expectations. Latent semantic analysis (LSA) [19] is used to compute these matches.

NORMIT

NORMIT [21] is designed for university level students and provides a problem-solving environment for database normalisation. Students are expected to

self-explain while solving problems. In contrast to other ITSs that support self-explanation, NORMIT requires an explanation when an action is performed for the first time. For the subsequent actions of the same type, explanation is required only if the action is performed incorrectly. This approach was utilised because it tends to reduce the strain on more able students by not asking them to provide the same explanation every time an action that is performed correctly and also to enable the system to provide enough situations for students to develop and improve their self-explanation skills. Similar to the PACT Geometry Tutor and SE-Coach, NORMIT supports self-explanation by prompting the student to explain by selecting one of the offered options. For instance, one of the tasks in database normalisation is to specify the candidate keys of a relation. When a student specifies a candidate key for the given problem, the student is asked to explain why the specified attribute(s) make a candidate key, if that is the first time he/she is specifying candidate keys. If the student selects an incorrect explanation, the system will then ask for another explanation. In contrast to the first question, which was problem-specific, the second question is generic. The student will be asked to define a candidate key by selecting one of the options given. An evaluation study has been conducted with second year students at the University of Canterbury to assess the effects of self-explanation support in the domain of data normalisation. Due to the small number of participants sound conclusions cannot be made, but the results do indicate that the student's problem solving skills and declarative knowledge increases with self-explanation [21].

Discussion

These systems use different approaches to facilitate self-explanation, depending on the domain and the target student group. Problem solving activities in these domains are well structured, and the types of self-explanations expected from students can be clearly defined. For example, in Mathematics and Physics, students are expected to explain the theorems that they have used. In computer literacy, the students are expected to explain the definitions, meanings of terms and how a certain task is being carried out. However, it is challenging to incorporate self-explanation in the domain of database design, as it is an open-ended task: there is an outcome defined in abstract terms, but there is no procedure to find that outcome. It is not sufficient to ask the students to explain the concepts of database modelling, as the database design skills can only be developed through extensive practice.

3 KERMIT: A Knowledge-based ER Modelling Tutor

KERMIT is an ITS aimed at the university-level students learning conceptual database design. The architecture of the system is illustrated in Fig. 1.

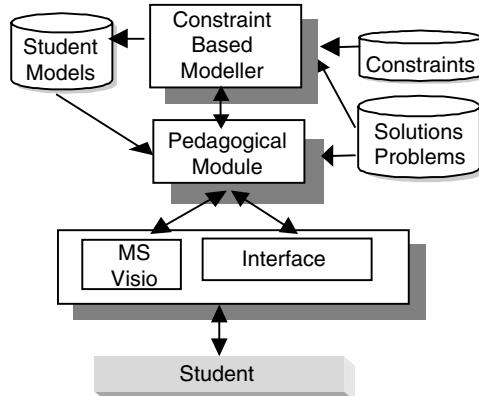


Fig. 1. The architecture of KERMIT

For a detailed discussion of the system, see [27]; here we present some of its basic features. KERMIT is a problem-solving environment in which students practise database design using the Entity Relationship (ER) data model. The system is intended to complement traditional instruction, and assumes that students are familiar with the ER model. The system consists of an interface, a pedagogical module, which determines the timing and content of pedagogical actions, and a student modeller, which analyses student answers and generates student models.

KERMIT contains a set of problems and the ideal solutions to them, but has no problem solver. In order to check the correctness of the student's solution, KERMIT compares it to the correct solution, using domain knowledge represented in the form of more than 90 constraints. It uses Constraint-Based Modelling [22, 25] to model the domain and student's knowledge.

Constraint based modelling [22], based on the theory of learning from performance errors [26], is a student modelling approach that reduces the complexity of the task. CBM focuses on correct knowledge rather than on describing the student's knowledge as in model tracing [6]. The key assumption in CBM is that the diagnostic information is in the problem state at which the student arrives and not in the sequence of his/her actions. This assumption is supported by the fact that a correct solution cannot exist for a problem that traverses a problem state which violates fundamental ideas or concepts of the domain. Since the space of incorrect knowledge is much greater than correct knowledge, knowledge about a domain is represented as a set of state constraints. A state constraint is an ordered pair (C_r, C_s) , where C_r is the relevance condition and C_s is the satisfaction condition. The relevance condition identifies the states in which the constraint is relevant and the satisfaction condition identifies a subset of relevant states in which the constraint is satisfied. For instance, violation of constraint 11 (described below) initiated the first feedback message (i.e. *Check whether all the entities are necessary. Check*

whether some of your regular entities should be represented using some other type of construct), which appears on the bottom left window of Fig. 2. This is caused by the error *CHAPTER should not be modelled as a regular entity.* Constraint 11 is implemented as follows:

Relevance Condition: For each regular entity in the student solution

Satisfaction condition: there should be a matching regular entity in the ideal solution

In this example, the relevance condition is satisfied by the regular entity CHAPTER. However, it violates the satisfaction condition because CHAPTER is modelled as a weak entity in the ideal solution.

CBM does not require a runnable expert module to generate pedagogical actions. This is an important advantage as it is very difficult to design an expert module for many domains. CBM is also computationally very simple as student modelling is reduced to pattern matching. During the evaluation of a problem state all relevance patterns are matched against the problem state. In the case where the problem state matches the relevance pattern, it is then checked against the satisfaction condition. If the satisfaction condition is not met, then the constraint is violated, which indicates errors. Moreover, CBM also does not need extensive bug libraries, which model students' misconceptions about the domain.

Students have several ways of selecting problems in KERMIT. They may work their way through a series of problems, arranged according to their complexity. The other option is a system-selected problem, when the pedagogical module selects a problem for the student on the basis of his/her student model.

The interface is composed of three windows tiled horizontally. The top window displays the current problem and provides controls for stepping between problems, submitting a solution and selecting feedback level. The middle window is the main working area. In this window the student draws ER diagrams. Figure 2 represents the interface of KERMIT-SE, which is very similar to that of the original of KERMIT. The only difference is that bottom window of KERMIT has only one section in original KERMIT.

The feedback from the system is grouped into six levels according to the amount of detail: *Correct, Error Flag, Hint, Detailed Hint, All Errors* and *Solution.* The first level of feedback, *Correct*, simply indicates whether the submitted solution is correct or incorrect. The *error flag* indicates the type of construct (e.g. entity, relationship, etc.) that contains the error. For example, when the solution in Fig. 2 is submitted, *error flag* provides the message *Check your entities, that's where you have some problems.* This is associated with the error CHAPTER being modelled as a regular entity instead of a weak entity. *Hint* and *Detailed Hint* offer a feedback message generated from the first violated constraint. In the case of solution in Fig. 2, hint provides a general message *Check whether all the regular entities are necessary. Check whether some of your regular entities should be represented using some other type of construct.* On the other hand, detailed hint provides a more specific

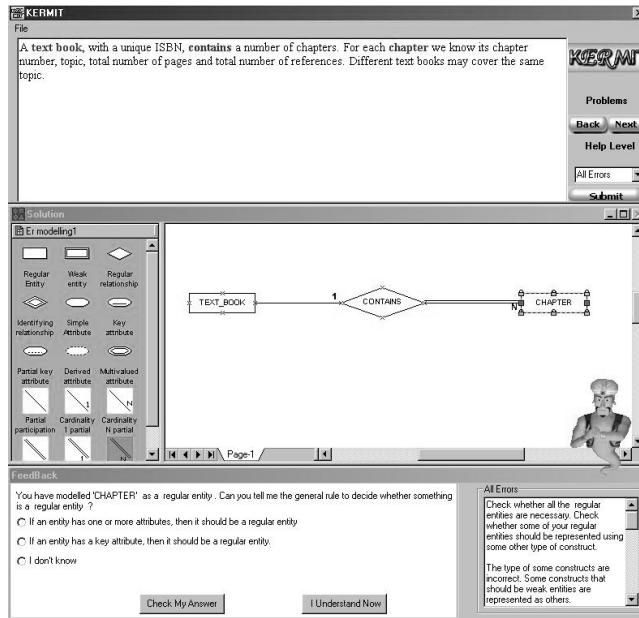


Fig. 2. Interface of KERMIT-SE

message *CHAPTER* should not be an entity. It may be extra or you may want to represent it using some other type of construct, where the details of the erroneous object are given. Not all detailed hint messages give the details of the construct in question, since giving details on missing constructs would give away solutions. A list of feedback messages on all violated constraints is displayed at the all errors level (as indicated in the bottom left hand in Fig. 2). The ER schema of the complete solution is displayed at the final level (solution level).

Initially, when the student begins to work on a problem, the feedback level is set to the correct level. As a result, the first time a solution is submitted, a simple message indicating whether or not the solution is correct is given. The level of feedback is incremented with each submission until the feedback level reaches the detailed hint level. In other words, if the student submits the solutions four times the feedback level would reach the detailed hint level, thus incrementally providing more detailed messages. Automatically incrementing the levels of feedback is terminated at the detailed hint level to encourage the student to concentrate on one error at a time rather than all the errors in the solution. The system also gives the student the freedom to manually select any level of feedback according to their needs. In the case when there are several violated constraints, and the level of feedback is different from “all errors”, the system will generate the feedback on the first violated constraint.

The constraints are ordered in the knowledge base by the human teacher, and that order determines the order in which feedback would be given.

KERMIT maintains two kinds of student models: short-term and long-term ones. Short-term models are generated by matching student solutions to the knowledge base and the ideal solutions. The student modeler iterates through each constraint in the constraint base, evaluating each of them individually. For each constraint, the modeller initially checks whether the current problem state satisfies its relevance condition. If that is the case, the satisfaction component of the constraint is also verified against the current problem state. Violating the satisfaction condition of a relevant constraint signals an error in the student's solution.

The short-term student model consists of the relevance and violation details of each constraint, discovered during the evaluation of the problem state. The short-term model is only dependent on the current problem state and does not account for the history of the constraints such as whether a particular constraint was satisfied during the student's last attempt. The pedagogical module uses the short-term student model to generate feedback to the student.

The long-term student model of KERMIT is implemented as an overlay model. In contrast to the short-term model, the long-term model keeps a record of each constraint's history. It records information on how often the constraint was relevant for the student's solution and how often it was satisfied or violated. The pedagogical module uses these data to select new problems for the student. The long-term student model is saved in a file when the student logs out.

4 Design and Implementation

All systems that facilitate self-explanation prompt students to explain most of the problem-solving steps, requiring students to point out the definitions/theorems used. We believe this approach puts too much burden on able students. Therefore, our tutor prompts for self-explanation only when the student violates a constraint, which indicates missing/erroneous knowledge, or a slip. The tutor is thus able to customise self-explanation based on the student model so that the knowledge construction is facilitated for students who have misconceptions or gaps in their knowledge without disrupting others [8]. For a detailed discussion on how KERMIT was enhanced to support self-explanation see [32]. Some of the important details are discussed here.

Since a student can make several errors (i.e. several constraints can be violated) in a single submission, the pedagogical module (PM) needs to decide on which error to initiate the self-explanation process. The constraints in the knowledge base are ordered according to traditional ER modelling procedure, starting with modelling entities first, then relationships and finally the attributes. This ordering alone is not sufficient to select an error to prompt for self-explanation, as most solutions violate more than one constraint. At

the same time, semantic constraints are not specific enough to guide self-explanation effectively. For instance, the constraint *A construct that should be a regular entity has been represented by another type* is violated when a regular entity is represented either as a simple attribute or a weak entity. Different approaches are required in these two cases. Self-explanation in the first case should help the student to clarify the definitions of entities and attributes so that the student understands when to use them. In the latter case, the student should understand the differences between regular and weak entities, which will enable the error to be corrected and the correct design decisions made in future. Also, the pedagogical module should enable students to build a more comprehensive mental model of the domain knowledge by giving them an opportunity to learn basic concepts before complicated ones. For instance, students need to understand the reason for a certain component to be modelled as a regular entity before understanding the relationships that this entity participates in or attributes of that entity.

4.1 Development of an Error Hierarchy

We analysed the constraint base of KERMIT and students' answers for an assignment in an introductory database course to develop a list of errors that can occur in the domain of database modelling. Students' responses to pre- and post-tests of a previous evaluation study conducted for KERMIT [29] were also analysed to make the list comprehensive. The list of errors compiled was then used to develop a hierarchy of errors that can be used to introduce a high-level structure of constraints to the constraint base of KERMIT. The overall view of the hierarchy of errors is shown in Fig. 3.

As indicated in Fig. 3, all types of errors in this domain can be divided into two categories: (i) Syntax errors (ii) Semantic errors. Violated constraints (which are specified by the constraint numbers in Figs. 3 and 4) for each type of error are represented as the leaves of the hierarchy. Constraints for the nodes in Fig. 3 are given in separate lines to indicate that each constraint deals with a specific type of error.

Our experience in teaching database design in a traditional classroom setting at the University of Canterbury indicates that it is easier for students to understand and correct syntax errors than semantic ones. Therefore, the error hierarchy focuses on dealing with the syntax errors before the semantic ones. This is achieved by placing the node *Syntax errors*, which deals with all the syntax errors before the node *Semantic errors*, which deals with all the semantic errors (Fig. 3).

Nodes in this hierarchy are ordered from the basic domain principles to more complicated ones so that a student can be guided systematically towards building and repairing his/her mental model of the domain. For example, constraint 1 is violated when more than one ER construct is used to model

ALL ERRORS**Syntax Errors****Errors dealing with simple connections**

1
2
3
4
5
6
101
7
8
9
10
21

Errors dealing with other syntactic concepts

22
23
25
33
34
39
40
41
44
45
46
47
48
72
73

Semantic Errors

Using an incorrect construct type
Extra constructs
Missing constructs
Connecting an attribute to an incorrect construct
Errors dealing with cardinalities and participation

Fig. 3. Overall view of the error hierarchy

a phrase of the problem statement whereas constraint 21 is violated when a relationship does not have at least two participating entities. Thus, constraint 1 deals with a relatively simpler error than 21. Therefore, when the hierarchy is traversed from left-to-right, top-to-bottom, the simplest error type for a given student solution can be found.

Syntax Errors

The node *Syntax errors* (Fig. 3) deals with all syntactic errors that can be divided into two categories: (i) Errors dealing with simple connections (ii) Errors dealing with other syntactic concepts. The node *Errors dealing with simple connections* deals with basic syntax errors such as *An Entity cannot be directly connected to another entity*, *Attributes can only be connected using single lines with no cardinality* and so on. The node *Errors dealing with other syntactic concepts* deals with more complicated syntax errors such as *Participation between the identifying relationship and the weak entity should be total*, *A regular entity should have a candidate key attribute* and similar.

Semantic Errors

The node *Semantic errors* (Fig. 3) deals with all types of semantic errors that are identified by comparing the student solution with the ideal solution. As indicated in Fig. 3, semantic errors are divided into five categories.

All these categories are further divided into sub categories as indicated in Fig. 4. For instance, the node *Using an incorrect construct type* is divided into two child nodes: (i) *Using a completely different type of construct*, (ii) *Using a different variation of the correct construct*. In ER modelling design decisions need to be made using a two-step process. In the first step, a student needs to decide whether a certain phrase in the problem statement can be modelled as an entity, relationship or an attribute. In the second step, he/she needs to decide whether it is regular or weak in the case of an entity; regular or identifying in that of a relationship etc. The error types associated with the first step are classified under the node *Using a completely different type of construct*. The node *Using a different variation of the correct construct* deals with the errors of the second step.

For example, the CHAPTER regular entity solution in Fig. 2 violates constraints 11 and 14 because it should be modelled as a weak entity. Constraint 11 deals with extra regular entities in the solution which should be modelled using some other type of construct. Constraint 14 deals with weak entities which are represented as some other type of construct in the solution. This situation is described by the node *Using a regular entity to represent a weak entity* which is a child node of *Using a different variation of the correct construct* (Fig. 4). The two constraints (i.e. 11 and 14) are combined with “and” to state the requirement that both constraints need to be violated by the same constraint. i.e. CHAPTER is an extra regular entity (constraint 11) and needs to be modelled as a weak entity (constraint 14).

4.2 Effectiveness of the Hierarchy

Selection of an error to initiate the self-explanation process is crucial for enhancing learning because if error selection is not effective, it might be difficult

- Using an incorrect construct type
 - Using a completely different type of construct
 - Using an entity to represent another type of construct
 - Using an entity to represent a relationship
(11 or 200) and (27 or 28)
 - Using an entity to represent an attribute
(11 or 200) and 203
 - Using another type of construct to represent an entity
 - Using a relationship to represent an entity
(13 or 14) and (19 or 211)
 - Using an attribute to represent an entity
(13 or 14) and 202
 - Other representations
 - Using an attribute to represent a relationship
(27 or 28) and 202
 - Using a relationship to represent an attribute
(19 or 211) and 203
- Using a different variation of the correct construct
 - Entity
 - Using a regular entity to represent a weak entity
(11 and 14)
 - Using a weak entity to represent a regular entity
(13 and 200)
 - Relationship
 - Using a regular relationship to represent an identifying relationship
(28 and 19)
 - Using an identifying relationship to represent a regular relationship
(27 and 211)
 - Attribute
 - Using a different type of attribute
(54 or 55 or 56 or 57 or 58 or 59 or 80 or 83 or 84 or 85)

Fig. 4. Detailed view of the node *Using an incorrect construct type*

for students to systematically develop a comprehensive mental model of the domain concepts. Selection of the dialogues depends on the student solution and the error hierarchy. We conducted an informal study to investigate the effectiveness of the error hierarchy before implementing it in the system.

Undergraduate students who have either completed or were currently enrolled in an introductory database course at the University of Canterbury were asked to participate in this study. A total of six students participated in the study. We were unable to conduct the study with a larger group of students because it was performed in July 2001, during a term break at the university. Each participant was given two problems (arranged in increasing order of complexity) and was asked to design ER models for these problems on paper. The two problems used in the study are given in Part I of Appendix 6.

When they finished designing the ER models, their solutions were evaluated. If there were errors, participants were given feedback on the first error, which was selected using the error hierarchy. The feedback messages were identical to the feedback provided by KERMIT, when a *Hint* is requested. If

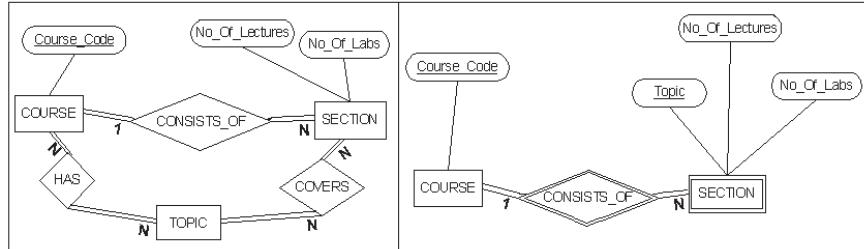


Fig. 5. A student’s solution (*left*) and an ideal solution (*right*)

they requested more help, more specific feedback was given, which was identical to the *Detailed Hint* provided by KERMIT. Participants were shown the ideal solution on request.

Most students were able to solve the first problem with very little feedback. However, some students were confused by the feedback given for the second problem. For example, consider the second problem used in this study (Fig. 5). Figure 5 shows a student’s solution and the ideal solution. The student’s solution contains several errors: The relationships HAS and COVERS are not needed, TOPIC entity should be represented as an attribute, SECTION entity should be represented as a weak entity, and its partial key is missing and CONSISTS_OF should be represented as an identifying relationship. Since TOPIC is modelled as a regular entity the constraint that checks whether each regular entity has at least one attribute is violated. According to the hierarchy, the first error that was selected to provide feedback for is *TOPIC entity does not have attributes*.

Upon receiving this feedback, many students tried to find attributes for the entity type *TOPIC* without realising that *TOPIC* should not be an entity in this model. The error *TOPIC entity does not have any attributes* was caused by the error *TOPIC should not be an entity. It should be represented using some other construct*. However, the pedagogical module fails to identify that it will be more effective if the student is guided to correct the latter error, because the hierarchy focuses on guiding students to understand and correct the syntactic errors first. Since many students had difficulties in understanding the feedback in similar situations, we refined the \to overcome such problems as described in the following section.

4.3 Refining the Error Hierarchy

The previous hierarchy focused on dealing with all the syntactic errors associated with the existing constructs in the student solution before checking for semantic errors. This happened because the node *Errors dealing with other syntactic concepts* (Fig. 3), which deals with the syntactic errors of the student solution was visited before the node *Semantic errors* (Fig. 3) which deals

with the semantic errors, when the error hierarchy was traversed in left-to-right, top-to-bottom manner. Therefore, the node *Errors dealing with other syntactic concepts* was removed and the hierarchy was restructured to include the constraints associated with the removed node.

Figure 6 represents the overall view of the refined hierarchy. According to the refined hierarchy, the syntactic details of the existing constructs of a student solution are checked after ensuring their accuracy. As a result, the first error selected to provide feedback for the student solution in Fig. 5, now becomes *TOPIC should not be an entity. It needs to be represented as some other construct.*, which is handled by the refined node *Using an entity to represent an attribute* (Fig. 6).

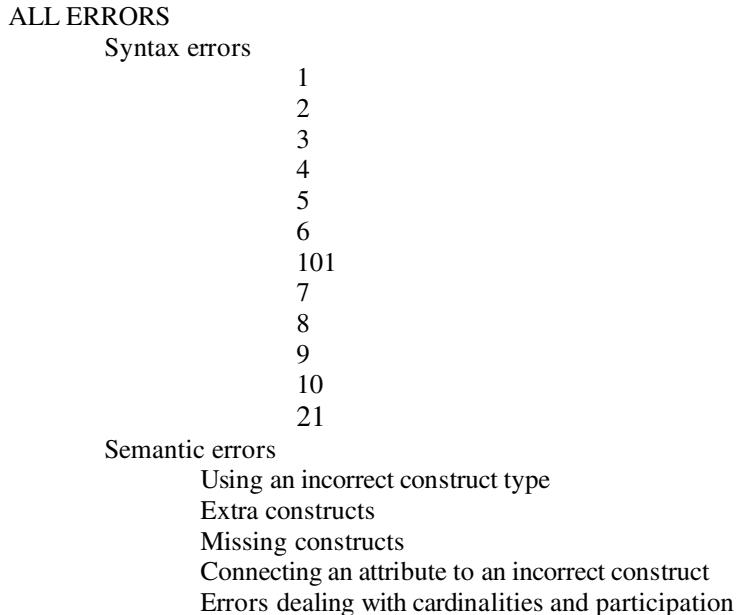


Fig. 6. Overall view of the refined error hierarchy

We reviewed the error selection process manually, using the refined error hierarchy for all the solutions collected during the informal study. The manual error selection process indicated that the refined error hierarchy is able to select the most suitable error to initiate the self-explanation process so that student is guided towards the ideal solution systematically.

4.4 Design of Dialogues

Self-explanation is facilitated through tutorial dialogues and a dialogue is developed for each error in the error hierarchy. For some simple errors, the

dialogue is limited to a detailed feedback message, as it is sufficient to explain the error. For other errors, a tutorial dialogue that consists of a series of questions is necessary to guide students to self explain both domain concepts and solution steps. Therefore, dialogues can be classified into two categories depending on the number of nodes in the dialogue: (i) Dialogues with a single node (ii) Dialogues with several nodes.

Dialogues with a Single Node

These are the simplest type of dialogues, which handle the basic errors associated with simple connections. In other words, this type of dialogues is mainly used to handle the errors associated with the node *Errors dealing with simple connections*. There are 12 such dialogues. An example of a single node dialogue is *You have connected an entity A to entity B directly. Entities cannot be directly connected to each other.*

Dialogues with Several Nodes

These dialogues consist of a series of questions beginning with general questions and moving to more specific ones. Most dialogues consists of four main stages:

- (i) Inform the student about the incorrect modelling decision and prompts to interpret his/her decision
- (ii) Guide the student to understand why his/her incorrect decision contradicts the problem statement
- (iii) Prompt the student to specify the correct modelling decision
- (iv) Prompt the student to review the domain concept learned through a review question

For example, consider the problem statement shown in Fig. 2. The student's solution contains several errors: the *CHAPTER* entity should be represented as a weak entity and its partial key is missing, *TEXT_BOOK* is missing the key attribute, *CONTAINS* should be an identifying relationship and its participation is wrong. When this student solution is submitted, the student modeller evaluates it against the constraint base and identifies the violated constraints. The pedagogical module then searches for the first tutorial dialogue for the same violated constraints. Since the error types and the corresponding dialogues are ordered according to the complexity of the domain concepts, the pedagogical module selects the dialogue by traversing the hierarchy in a top-to-bottom, left-to-right manner selecting the some or all violated constraints. In this situation, the pedagogical module selects the dialogue corresponding to the error *CHAPTER* represented as a regular entity instead of as a weak entity to start with. Figure 7 presents a sample dialogue

Tutor-1: You've modelled *CHAPTER* as a regular entity. What does that mean in terms of properties of a regular entity?

Student-1: I don't know

Tutor-2: Ok! Let's try a simple thing. What is used to uniquely identify instances of a regular entity?

Student-2: The key attribute

Tutor-3: Great! Now tell me whether modeling *CHAPTER* as a regular entity contradicts with the problem statement?

Student-3: I'm not sure

Tutor-4: Let me help you a bit. Check carefully whether any of the attributes of *Chapter* can uniquely identify the instances of *Chapter*? Please press Done when you finish.

{When Done button is pressed} Please tell me whether you managed to find any key attributes

Student-4: No

Tutor-5: Fine. It doesn't have any attributes that can uniquely identify instances of *Chapter*. So now can you tell me why modeling *Chapter* as a regular entity contradicts with the problem statement?

Student-5: Because none of the attributes can be a key attribute.

Tutor-6: Great! Now tell me the best way to model *Chapter*?

Student-6: A weak entity

Tutor-7: Well Done! Now select the correct pair of words to fill in the blanks so that you'll remember what you learnt.

A is used to uniquely identify instances of a entity.

Student-8: key attribute, regular

Tutor-9: Excellent, Go ahead and do the changes. Submit the solution when you're done.

Fig. 7. A sample tutorial dialogue

that occurs between the student and the tutor. In the first stage of the dialogue, the tutor informs the student about the incorrect modelling decision and prompts him/her to interpret his/her decision (*tutor-1*).

Because self-explanation is initiated by making the student aware of an incorrect action and prompting them to interpret their decision, the student is given an opportunity to reflect on their problem-solving procedure, which is one of the important meta-cognitive skills. A list of possible answers is provided. Sometimes it might be difficult for students to select the correct explanation even from a list of possible explanations provided by the tutor. In order to provide an opportunity for students to request self-explanation, the tutor provides phrases like *I'm not sure* and *I need more help*. If the student selects one of these phrases or an incorrect explanation, there is evidence that he/she has difficulties in providing the correct explanation. In such situations, the student is given more help by explaining the domain concept or by guiding him/her through simpler questions.

For example, if the student fails to interpret his/her modelling decision correctly, he/she is guided to simple, straightforward questions (*tutor-2*). If he/she still fails to understand the mistake through simple prompts provided by the tutor, the correct answer is provided as a last resort.

In the second stage of the dialogue, the student is guided to explain why his/her modelling decision contradicts the problem statement. First, the student is given an opportunity to understand that his/her modelling decision leads to a contradiction (*tutor-3*) and then he/she is asked how the modelling decision under discussion contradicts the problem statement (*tutor-5*).

If it is difficult for the student to understand that his/her decision contradicts with the problem statement, the tutor guides the student to search for

more information in the problem statement. For instance, when the student fails to understand that there is a contradiction between his/her modelling and the problem statement, he/she is asked to search for any key attributes for the entity type CHAPTER (*tutor-4*). As students need to be actively engaged in finding out whether there are any key attributes for CHAPTER, the possible answers are not given for the question immediately. Instead, the list of possible answers is displayed only when the student informs the tutor that he/she is ready to continue with the self-explanation process. By delaying the display of possible answers when students are expected to search for information in the problem statement, they are encouraged to actively engage in understanding why their modelling decision is incorrect. Then he/she is given the opportunity to understand that CHAPTER cannot be modelled as a regular entity because it does not have a key attribute.

In the third stage, he/she is asked to specify the correct modelling decision. Again, if he/she fails to specify the correct the modelling decision, the tutor provides the correct answer with an explanation (*tutor-6*).

In the final stage of the dialogue, the student is given a review question to provide another opportunity to understand and learn the concept discussed by the tutor (*tutor-7*). Various types of review questions are used to increase the student's motivation to go through the review question. The review questions can be simple questions, fill-in-the-blanks, or true-false questions. If the student has difficulty in answering the review question correctly, the correct answer is provided.

There is evidence that immediate feedback on students' responses has the potential to enhance learning in ITS [4]. Therefore, it is important to provide feedback on the accuracy of the self-explanations provided by students during problem-solving. In addition to providing feedback on students' responses, phrases such as *Well done*, *Great* and *Good job* are used in the dialogues to encourage students to provide more self-explanations. Furthermore, when a student fails to provide the correct self-explanation he/she is encouraged by using phrases such as *Let me help you a bit* and *Think again* instead of phrases like *Your answer is incorrect. Please try again*.

Even though the dialogues contain a series of questions, students can correct a mistake as soon as they realise it without going through the entire dialogue. Therefore, knowledge construction is facilitated for students who have misconceptions or gaps in their knowledge without disrupting others [6].

5 Evaluation

As evaluation is fundamental in all stages of developing an ITS, we conducted two evaluation studies, described in this section. A pilot study was carried out to investigate the effectiveness of the self-explanation support provided by KERMIT-SE in enhancing learning. The system was then refined based

on the results of the pilot study. The second study was conducted in a real classroom setting using the refined version of the system.

5.1 Pilot Study

The pilot study was conducted to explore the effects of self-explanation support provided by KERMIT-SE. It also aimed to discover students' perceptions about the various aspects of self-explanation support incorporated in the tutor, such as the prompts and the number of levels used in the tutorial dialogues. It was carried out as a think-aloud protocol [13], in May 2002 at University of Canterbury. The participants were postgraduate students enrolled in a course titled *Cognitive Modelling and Intelligent Tutoring Systems* (COSC420) in the first semester of 2002. Even though the target population for KERMIT-SE is undergraduates who are familiar with conceptual database design, it was not possible to gain access to this population at the time of the pilot study because the database course was taught in the second semester of 2002.

Participants had completed 60% of the ITS course lectures, and were expected to have a good understanding of ITS. All participants except one were familiar with ER modelling. Twelve postgraduate students participated in the study.

The system prototype that was used in the study supported self-explanation only for the constraints that deal with entities and relationships. The student model in the system prototype was not enhanced to record the self-explanation ability of users because the study aimed to discover only whether self-explanation components incorporated were effective in enhancing learning.

Procedure

Two versions of the system were used in the pilot study, referred to as A and B. The two versions were identical except for problem set. The problem sets used for versions A and B are given in Part II of Appendix 6. Each version had 3 problems arranged according to increasing complexity. Initially, one of members in the group performed the role of the participant and interacted with version A of the system. The participant was asked to verbalise his/her thoughts while performing a database modelling task using KERMIT-SE. Participants were able to skip the problems without completing them and to return to previous problems. Upon completion of the interaction, the roles were interchanged and the new participant interacted with version B. One of the researchers observed the sessions with the consent of participants. Participants were expected to provide feedback during informal discussion after the think-aloud session. Data was collected from four sources: video footages of think-aloud sessions, participants' logs, informal discussions after the session and researcher's observations.

Learning

A total of twelve students participated in the pilot study. Information on important events such as logging in, submitting a solution and requesting help which could occur while interacting with the system, were recorded in a log specific to each participant. The date and time of each event were also recorded. Table 1 summarises the data extracted from the participants' logs.

Table 1. Details of the system interaction

| | Mean | Standard Deviation |
|---|-------|--------------------|
| Time spent on problem-solving (min.) | 47:06 | 26:01 |
| No. of attempted problems | 2.75 | 0.62 |
| No. of completed problems | 1.67 | 0.49 |
| Time spent per attempted problem (min.) | 14:11 | 8:10 |

The interaction time has a very high standard deviation because the time required by individual participants respond to self-explanation dialogues varied significantly. There were only three problems to complete, so we considered a mean value of 2.75 to be quite high. This may be due to the fact that participants tried to complete all three problems, as it was part of their assessment for the course that they were enrolled in.

The completion rate of 60.7% demonstrates that many participants were able to complete the first two problems. Most participants did not show sufficient interest in completing the third problem, which was more complicated than the first two, because they felt that they had gathered sufficient data for their assignment after completing the first two problems.

The domain knowledge in KERMIT-SE is represented by constraints [27]. If these constraints represent psychologically appropriate units of knowledge, then learning should follow a smooth curve when plotted in terms of constraints [5]. In order to evaluate this expectation, the participants' logs were analysed, and each problem-state in which a constraint was relevant was identified. These identified occasions are referred to as *occasions of application* [22]. Each constraint relevance occasion was ranked 1 to R. For each occasion we recorded whether a relevant constraint was satisfied or violated. We then calculated the probability of violating a constraint on the first occasion of application, the second occasion and so on, for each participant. The probabilities were then averaged across all participants and plotted as a function of the number of occasions when C was relevant, as shown in Fig. 8. At n (occasion number) = 1, all participants had a constraint relevant whereas only six participants had a constraint relevant at n = 19. This indicates that the number of constraints that were relevant decreases as occasion number increases. Hence, a single failure at the end of the learning curve will have a much larger

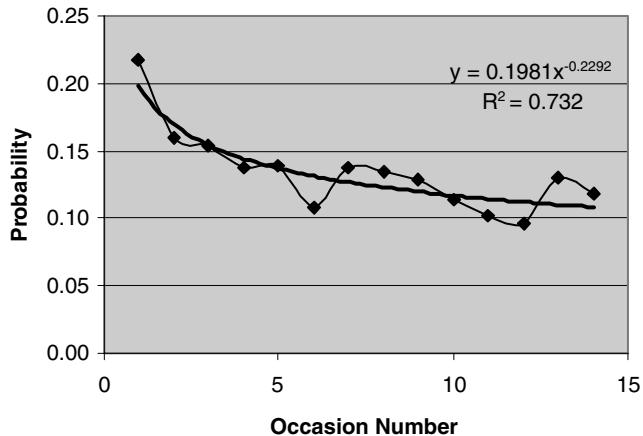


Fig. 8. Probability of violating a constraint as a function of number of occasions when that constraint was relevant, cut-off = 14

effect on the probability than an error at the start of the curve. In order to reduce this effect, an arbitrary cut-off point at $n = 14$ was selected, at which at least two thirds of the participants were still active.

The power curve displays a close fit with an R^2 fit of 0.732. The probability of 0.21 for violating a constraint at its first occasion of application decreased to 0.11 at its fourteenth occasion of application displaying a 47.6% decrease in the probability. The results of the mastery of constraints reveal that students appear to learn ER modelling concepts by interacting with KERMIT-SE.

Subjective Analysis

This section summarises the subjective data collected from participants during the informal discussions and the observations recorded by the first author. For a detailed discussion see [32]. Some of the important suggestions are discussed here.

Tutorial Dialogues Facilitating Self-explanation

The majority of the participants felt that the dialogues helped them to understand the domain concepts that they found difficult. Some of them even understood that the dialogues attempted to cover both problem-specific help and domain concepts associated with errors in the student solution. However, it was difficult for many participants to understand the tutor's prompts to interpret their modelling decisions. For instance, if a participant indicated a cardinality of N between entity A and relationship R instead of 1, the tutor

asked the student *You have indicated the cardinality between entity A and relationship R as N. How can it be interpreted according to ER modelling?*

In such situations, most participants read the list of answers provided by the tutor in order to understand the question better. This difficulty might be due to the fact that the participants are not generally expected to explicitly interpret their modelling decisions in a traditional classroom environment. Some of them felt that the tutor would be easier to understand if the tutor used a simple prompt like *You have indicated cardinality between entity A and relationship R as N. Can you tell me why it is wrong?*

Some participants felt that some review questions were too easy and thus did not greatly assist them to improve their understanding as expected. One example of such a question was:

*Select the correct answer before starting to make the changes.
The cardinality of the regular entity (owner) of an identifying relationship should always be*

This question belongs to the dialogue that discusses the concept *Cardinality of an owner entity of an identifying relationship should be 1.*

Levels of the Tutorial Dialogues

Each dialogue consists of a series of questions, providing more support for self-explanation with each successive question. However, users have the opportunity to correct the mistake as soon as they realise the mistake. In other words, they do not have to go through the entire dialogue for a particular error before modifying their solutions. However, all participants except two went through the entire dialogue before attempting to correct their solutions. The two participants who tried to correct mistakes without completing the dialogue reported that they were able to do that because they became familiar with the system by observing their partner before using the system prototype themselves. Some participants did not pay attention to the prompts provided by the tutor after realising their mistakes.

During the discussion, many participants suggested having an explicit interface feature, such as a button labelled *Fix it now*, or *I understand now*, to make it easier for users to understand that they can correct mistakes soon after they are aware of them, without having to go through the entire dialogue.

Students' Impressions on Feedback

When a solution is evaluated by the system, feedback is provided in textual form, as well as through an animated pedagogical agent. All participants suggested to incorporate an error list, which contains a list of all the errors of

the student solution. They felt that such a list would be beneficial to obtain an overall view of the system's evaluation of the solution.

The majority of the participants agreed that the animated pedagogical agent tends to make the learning process more enjoyable. However, they felt displaying feedback as the agent's speech was a bit distracting, as the rate at which the feedback was displayed is much slower than that of the feedback window.

Participants also wanted more control over the agent's feedback. For instance, they wanted to be able to access the previous feedback whenever necessary. One participant felt that if the agent could move around the diagram and point to the errors, it would be much easier for users to understand the errors when the ER models become complicated.

Refining KERMIT-SE

This section how KERMIT-SE was further enhanced based on the outcomes of the pilot study.

Refining Tutorial Dialogues

As discussed in Sect. 5.1, participants found it difficult to interpret their modelling decisions. Therefore, questions that prompted users to interpret their modelling decisions were replaced by questions that assist users to apply domain concepts to the specific problem. For instance the previous prompt is replaced with: *You have indicated that the cardinality between entity A and relationship R as N. Can you tell me the correct question to use to decide the cardinality between entity A and relationship R?*

Consider another situation where a student has modelled a weak entity as a regular entity. The first question in the corresponding dialogue in the system prototype used for the pilot study was: *You have modelled entity A as a regular entity. What does that mean in terms of the properties of a regular entity?*

In the refined version, this has been replaced by *You have modelled entity A as a regular entity. Can you tell me the general rule to decide whether something is a regular entity?*

The new questions are more closely related to the cognitive activities occurring during the ER modelling process, and thus are more effective in guiding students to self-explain.

Another instance that participants found difficult to respond to is when the tutor asked them to specify whether their modelling decisions contradict the problem statement and, if so, to specify the reason for the contradiction. For instance, many participants had difficulty in understanding prompts like: *Can you tell me whether modelling entity A as a regular entity contradicts*

the problem description? or Why does modelling entity A as a regular entity contradicts the problem description?

Therefore, these prompts were replaced by simpler ones, which ask users to specify why their decisions were incorrect. For instance, both prompts mentioned above were replaced by *Now can you tell me why modelling entity A as a regular entity is incorrect?* This type of question is common in a traditional learning environment, so students are likely to find it easier to understand and respond.

Table 2 represents a comparison of the four main stages of a dialogue used in the system prototype and in the refined version. Only the first two stages were modified to enhance the support for self-explanation in the refined version of KERMIT-SE.

Table 2. Comparison of the layout of the dialogues in the system prototype used for the pilot study and the refined version of KERMIT-SE

| Stage | System Prototype Used for the Pilot Study | Refined KERMIT-SE |
|-------|--|--|
| 1 | Informs the student about the incorrect modelling decision and asked to interpret his/her decision | Informs the student about the incorrect modelling decision and asked to specify the reason (general rule) for his/her decision |
| 2 | Guides the student to understand that the incorrect action contradicts the problem statement | Prompts the student to understand why his/her decision is incorrect |
| 3 | Prompts the student to specify the correct modelling decision | Prompts the student to specify the correct modelling decision |
| 4 | Prompts the student to review the domain concept learned using a review question | Prompts the student to review the domain concept learned using a review question |

Some review questions (belongs to stage 4 in Table 2) were modified to increase their effectiveness. For instance, the review question of the dialogue that discusses the domain concept *Cardinality of an owner entity of an identifying relationship should be 1* was

The cardinality of the regular entity (owner) of an identifying relationship should always be

The question was modified to the following:

Why does the cardinality of the regular entity (owner) of an identifying relationship always have to be 1?, which has greater potential to facilitate deeper understanding of the domain concept discussed by the dialogue.

Figure 9 represents the refined version of the sample dialogue given in Fig. 7. The complete dialogue is given in Appendix 6.

Tutor-1: You've modelled *CHAPTER* as a regular entity. Can you tell me the general rule to decide whether something is a regular entity ?

Student-1: *I don't know*

Tutor-2: Ok! Let's try a simple thing. What is used to uniquely identify instances of a regular entity?

Student-2: *The key attribute*

Tutor-3: Great! Now tell me why modeling *CHAPTER* as a regular is incorrect?

Student-3: *I'm not sure*

Tutor-4: Let me help you a bit. Check carefully whether any of the attributes of *Chapter* can uniquely identify the instances of *Chapter*? Please press *Continue if you want more clarifications*.

{When Continue button is pressed} Can you tell me the attribute that you've identified as a key attribute for *CHAPTER*?

Tutor-4: *There's no attribute that can be a key for CHAPTER.*

Tutor-5: Great! Now tell me the best way to model *Chapter*?

Student-5: *A weak entity*

Tutor-6: Well Done! Now select the correct pair of words to fill in the blanks so that you'll remember what you learnt.

A is used to uniquely identify instances of a entity.

Student-6: *key attribute, regular*

Tutor-7: Excellent, Go ahead and do the changes. Submit the solution when you're done.

Fig. 9. Refined version of the dialogue given in Fig. 7

Incorporating an Error List

All the participants of the pilot study thought it was beneficial to have an overall view of the solution, so KERMIT-SE was modified to include a list of all errors upon submission of a student solution. The feedback window of KERMIT-SE (Fig. 2) was divided into two sub-windows. The right-hand side window is used to display the list of all errors when a student solution is submitted and the other window is used to display the tutorial dialogue. As the error list is used to provide an overall view of the student solution, it is sufficient to provide a brief description of each error. The feedback level *Hint* of KERMIT provides general feedback for each error, so we decided to use the messages on that level in the error list.

Figure 2 represents the interface of the refined version of KERMIT-SE when a student solution is submitted. The right-side pane of the feedback window displays the list of all errors, which provides an overall view of the student solution. When the student solution is submitted for the first time, the left-side window displays the following message:

OK! Let's look at the first error.

Check whether all the regular entities are necessary. Check whether some of your regular entities should be represented using some other type of construct.

A new button called *More Help* was added to the left-side window and is used to inform the tutor to initiate the self-explanation process of the selected dialogue.

The error list remains unchanged until the next submission of the student solution, as users should be able to refer to the error list whenever necessary. When a student decides to submit the solution again, the pedagogical module evaluates the solution and displays the new error list.

Some students may be able to correct the student solution using the error list alone. When they decide to request more help, they are guided to self-explain through the appropriate tutorial dialogue. Students have the opportunity to decide when to initiate the self-explanation process. Thus, the tutor does not force the natural self-explainers to go through the dialogues, compromising their motivation to use the tutor. By having a high-level overview of the student solution, the tutor is able to provide less feedback for students who are natural self-explainers. On the other hand, the tutor is able to provide detailed explanations for students who need guidance to self-explain.

Refining the Interface

The majority of participants of the pilot study did not understand that they could correct mistakes immediately without having to complete the dialogue. Thus a new button *I Understand Now* was added to the left-side window in which the tutorial dialogues are displayed (Fig. 2). Although students are not expected to explicitly imply that they understand their mistakes by pressing this button, the existence of the button may make it easier for users to realise that they do not have to complete the dialogue to correct mistakes.

The *OK* button, which was used to inform the tutor to evaluate the self-explanations selected by a student, as changed to *Check my answer* because then the purpose of the button is clearer (Fig. 2).

Discussion

The analysis carried out to investigate how participants acquired domain knowledge while interacting with KERMIT-SE indicated that the self-explanation components incorporated in KERMIT assisted participants in learning database design concepts. More importantly, interactions with the system indicated that participants' learning was not hampered by expecting them to self-explain when there were errors in their solutions. Furthermore, participants did not feel that it was overwhelming to solve problems as well as provide self-explanations, even though both are highly demanding cognitive

tasks. Participants' behaviour and suggestions provided insightful information on the usability aspects of the system, such as the questions, levels and support provided by the tutorial dialogues. For example, all the participants requested an overview of errors upon submission. As a result, a list of errors is now included in the system and is updated upon each submission of a solution. As many participants had difficulty in understanding some of the prompts in the dialogues, they were changed to make them similar to the problem-solving process generally followed in database design.

Although some suggestions made by the participants had the potential to improve the motivation to use the system, they were not considered feasible due to the time constraints of the research. For instance, one participant suggested incorporating functionality where the animated pedagogical agent could move around and point to the errors in the student solution. Even though this suggestion was technically feasible, it was not implemented due to time limitations.

Participants experienced a number of difficulties interacting with the system. The video footage was useful in identifying the bugs in the system. All the bugs identified were fixed to make the system more robust.

5.2 Evaluation Study

An evaluation study was conducted in July 2002 with students enrolled in an introductory database course at the University of Canterbury. We wanted to test the hypothesis that self-explanation facilitates acquisition of both procedural and conceptual knowledge. The experimental group used KERMIT-SE, while the control group used a cut down version of KERMIT. Both groups received the list of all errors for their solutions, and could ask for the ideal solution. Even though there were five different levels of feedback in the original KERMIT, only *Detailed Hint* was available to the control group to make it comparable with the experimental group. As the name suggests, *Detailed Hint* provides a detailed feedback message on a single error in a student solution.

Procedure

The experiment was carried out during normal lab hours over the duration of two weeks, and consisted of four phases: pre-testing, system interaction, post-testing and subjective system assessment. The pre- and post-tests consisted of two questions each, of equal difficulty (given in Appendix C). The first question required students to design an ER model for the given requirements, whereas the second question required them to explain the design decisions for the given ER model. The tests were rotated between successive sessions to minimise any effect resulting from variation in test difficulty. Ideally, each student was expected to spend a total of 4 hours to complete the four stages of the study.

We developed two different versions of the questionnaire: control group was given 12, and the experimental group 15 questions. Initially, students were asked about their previous experience in database modelling, and their impressions of the system and its interface. The experimental group was additionally asked about their perception of self-explanation support. Some of the questions asked were how easy/difficult the dialogues were to understand, and how useful the dialogues were for understanding errors. Students answered on a Likert scale with five responses ranging from *very poor* (1) to *very good* (5), and were also allowed to give free-form responses.

Pre- and Post-test Performance

Table 3 represents mean pre- and post-test scores and the standard deviation (given in parenthesis) for the different groups. The sizes of control and experimental group differ, as they depended on how many students turned out to corresponding lab sessions. The mean score on the pre-test for all students was 72.18% (SD = 18.72%). The difference in pre-test scores for the two groups is insignificant, confirming that the groups are comparable. Examining the logs of the session, we see that only 19 students in the experimental group have gone through at least one dialogue, (they had control over that via the *More Help* button). We are interested in these students, as the rest of the experimental group has not self-explained, and we summarize statistics for those students separately (column *self-explainers* in Table 3). The mean score on the pre-test for self-explainers is significantly higher than the mean for the control group. Therefore, we cannot directly compare the control group to self-explainers. However, the other students in the experimental group, who have not gone through any of the dialogues (column *non self-explainers* in Table 3), are comparable to the self-explainers, as there is no significant difference for these two groups of students on the pre-test.

Table 3. Mean pre- and post-test scores

| | Control | Experimental | Self-explainers | Non Self-explainers |
|-------------------|---------------|---------------|-----------------|---------------------|
| No of students | 72 | 53 | 19 | 34 |
| No. of post-tests | 59 | 35 | 18 | 17 |
| Pre-test | 70.98 (18.47) | 75.61 (17.33) | 79.32 (13.16) | 73.17 (20.47) |
| Post-test | 79.94 (17.75) | 78.11 (14.35) | 79.76 (12.22) | 77.37(16.76) |
| Gain | 8.96 (24.61) | 2.50 (21.04) | 0.44 (22.00) | 4.97 (20.30) |

The participation in the experiment was voluntary, and not all students completed the study. We report the number of post-tests in Table 3. The difference between the post-test scores of the experimental and control groups is not significant. The control group students improved significantly on the

post-test ($p = < 0.01$). However, these students had lowest existing knowledge (lowest pre-test score) and therefore had more room for improvement. Even though the self-explainers did better in the post-test, the improvement is not significant. As a result, the difference in gain scores for the experimental and the control groups was significant ($t = 1.33, p = 0.09$).

The difference between the post-test scores for the self-explainers and non self-explainers is not significant. Although both groups improved after interacting with the allocated system, the improvements of both groups were not statistically significant. As the result, the difference in gain scores was found to be statistically insignificant ($t = -0.61, p = 0.27$).

We wanted to investigate whether the system is more beneficial to *less* able students within the self-explainers and non self-explainers groups. Within each group, students were divided into *more* and *less* able groups based on their performance in the pre-test. The *more* able group comprised those students who scored above the mean score for all participants (i.e. 72.18%) and the remaining students formed the *less* able group. Table 4 represents the number of students in each group and their mean, gain and standard deviation.

Table 4. Test scores for less and more able students in the self-explainers and non self-explainers groups

| | | N | Pre-test | Post-test | Gain |
|---------------------|-----------|----|---------------|---------------|---------------|
| Self-explainers | Less able | 4 | 61.27 (6.67) | 84.71 (12.49) | 23.43 (17.78) |
| | More able | 14 | 84.48 (9.34) | 78.35 (12.23) | -6.13 (18.74) |
| Non self-explainers | Less able | 7 | 53.61 (16.33) | 68.18 (12.94) | 14.57 (16.04) |
| | More able | 8 | 86.52 (8.26) | 83.09 (17.35) | -3.43 (20.76) |

Less able students in both groups improved significantly ($t = -2.63, p = 0.03$ and $t = -2.4, p = 0.02$ for less able students in the self-explainers and non self-explainers groups, respectively). There was a decrease in performance for more able students in both groups although it was not significant ($t = 1.22, p > 0.1$ and $t = 0.46, p > 0.1$ for more able students in the self-explainers and the non self-explainers respectively). As a result, the gain between the less and more able students is significant for both groups ($t = 2.89, p = 0.01, t = 1.89, p = 0.04$ for self-explainers and non self-explainers groups, respectively). Furthermore, we found that the version of the system used had no significant effect on the student gain for less able or more able students in both groups.

We also analysed the students' performance on questions 1 and 2 on the pre- and post-tests separately for the self-explainers and non self-explainers (Table 5). Students had to use their procedural knowledge (problem-solving) skill to answer question 1 in the questionnaire. The performance for question 2

Table 5. Mean pre- and post-test scores for questions 1 and 2 for self-explainers and non self-explainers

| Group | Question 1 (procedural question) | | | Question 2 (conceptual question) | | |
|---------------------|-------------------------------------|------------------|------------------|-------------------------------------|------------------|-------------------|
| | Pre-test | Post-test | Gain | Pre-test | Post-test | Gain |
| Self-explainers | 76.82 (19.63) | 77.02 (16.60) | 0.20 (31.75) | 83.70 (15.21) | 84.81 (15.09) | 1.11 (23.49) |
| Non self-explainers | 68.36 (22.14) | 84.61 (15.76) | 16.24 (30.20) | 76.22 (32.29) | 60.00 (32.22) | -16.22 (28.03) |

can provide an indication of their self-explanation ability, as they are expected to explain the design decisions for a given ER model.

When a paired *t*-test was used for the pre- and post-test scores for question 1 for self-explainers, the results indicate that the improvement was not statistically significant. Although self-explainers did improve their performance on the conceptual question, the improvement was not significant.

Non self-explainers improved significantly on the procedural question ($t = -2.08, p = 0.02$) while their performance decreased significantly on the conceptual question ($t = 2.24, p = 0.02$). These results suggest that procedural knowledge can be improved by guiding students through general hint messages. Non self-explainers used only the general hint messages given in the error list, although it is difficult to improve their self-explanation ability using the same type of guidance. As a result of the significant difference between the pre- and post-test scores for non self-explainers, the gain for questions 1 and 2 was found to be significant ($t = 3.05, p = 0.002$).

When the gain for question 1 is compared for both groups, we found it was statistically significant ($t = 1.48, p = 0.07$). Furthermore, the performance for question 2 for both groups is also statistically significant ($t = -1.90, p = 0.03$).

The difference in pre-test scores for the procedural question is insignificant confirming that the groups are comparable. The non self-explainers achieved a much higher score on the same question in the post-test. The others improved only slightly but the difference in performance between the two groups on this question is not significant.

The difference in pre-test scores for the conceptual question is not significant confirming that the groups are comparable. However, self-explainers performed significantly better on the conceptual question in the post-test ($t = 2.74, p < 0.01$) than their peers. It is interesting to note that the performance of the non self-explainers decreased after using the cut down version of KERMIT which did not provide any self-explanation support. As the test used in the study were rotated between successive sessions any effect resulting from variation in test difficulty is minimised. Therefore, these results suggests that the self-explanation support provided by KERMIT-SE has

significantly contributed towards improving the conceptual knowledge of the students.

Learning

There is no significant difference between the problem solving time for the control and the experimental groups (Table 6). However, self-explainers spent significantly more time on problem solving ($t = 5.01, p < 0.001$) than non self-explainers. This might be due to the self-explanation dialogues, as student needed time to answer the questions. However, the self-explainers also attempted and solved significantly more problems than the rest of the experimental group. Therefore, self-explanation supports problem-solving.

Table 6. Mean system interaction details

| | Control | Experimental | Self-explainers | Non self-explainers |
|-----------------------------|----------------|---------------|-----------------|---------------------|
| No of students | 72 | 53 | 19 | 34 |
| Problem solving time (min.) | 105:21 (44:19) | 98:37 (49:34) | 133:21 (30:44) | 79:13 (47:41) |
| No. of attempted problems | 7.08 (2.69) | 6.34 (3.22) | 8.21 (2.42) | 5.29 (3.17) |
| No. of completed problems | 5.25 (2.43) | 4.62(2.63) | 6.36 (2.31) | 3.65 (2.29) |
| No. of post-tests | 59 | 35 | 18 | 17 |
| Pre-test | 70.98 (18.47) | 75.61 (17.33) | 79.32 (13.16) | 73.17 (20.47) |
| Post-test | 79.94 (17.75) | 78.11 (14.35) | 79.76 (12.22) | 77.37 (16.76) |

The student logs were also used to analyse the mastery of constraints, as explained in Sect. 5.1. Figure 10 illustrates the probability of violating a constraint plotted against the number of occasions when constraints were relevant averaged over all the participants. All 31 non self-explainers had a constraint relevant at $n = 1$, and only 16 of them had a constraint relevant at $n = 19$. Initially (i.e. $n = 1$), all 19 self-explainers had at least one constraint relevant whereas at $n = 19$, only 17 participants had a constraint relevant. i.e. as the occasion of application increases, the set of constraints that were relevant on that occasion decreases in size, and the impact of a single failure on the probability increases. To reduce the impact of a single failure, we decided to use a cut off point of $n = 15$ at which at least two thirds of the participants had a constraint relevant.

The power curve for the non self-explainers has a better fit than for the self-explainers. This might be due to the fact that existing knowledge of non self-explainers is lower than that of the self-explainers and as a result, they have more room for improvement

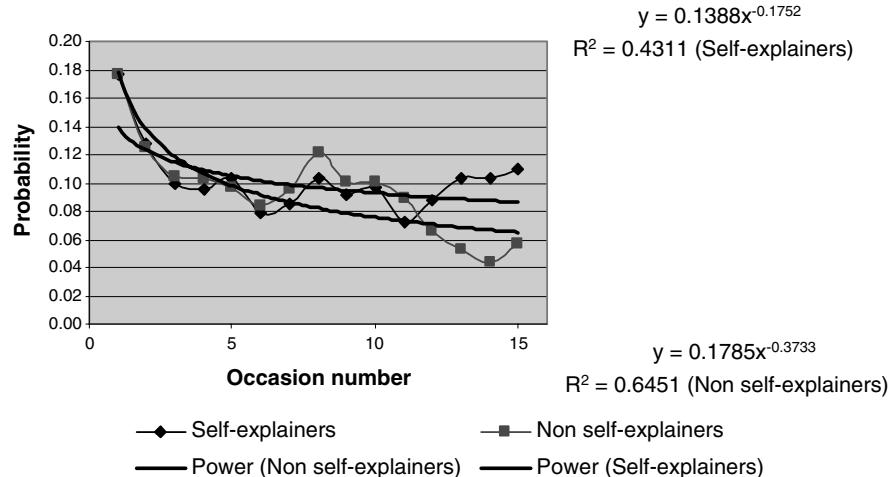


Fig. 10. Probability of violating a constraint as a function of the occasion when that constraint was relevant for the self-explainers and non self-explainers

Self-explanation Performance

The self-explainers on average went through 6.95 dialogues, ranging from 1 to 21. On average, students completed 78.25% of the dialogues, with an average of 57.61% of correct responses to the questions in the dialogues. To test the second part of our hypothesis (self-explanation results in improved conceptual knowledge), we analysed the student answers to the first question of a chosen dialogue, which prompts students to explain domain concepts. Figure 11 illustrates the correctness of students' explanations. The probabilities of correct answers on the first and subsequent occasions were averaged over all error types and over all students. The fit to the power curve is very good, indicating that students do learn by explaining.

We also analysed students' answers to the second question in a chosen dialogue, which expects students to provide problem-specific explanations. Figure 12 illustrates the correctness of students' problem-specific explanations. On the first occasion, 15 students provided a problem-specific explanation, whereas only a single student did so at $n = 6$. To reduce impact of a single failure in the probability of providing the problem-specific explanations correctly, we selected an arbitrary cut-off point of $n = 3$. A good fit to the power curve indicates that students learn by providing problem-specific explanations. The R^2 values for the conceptual question and the problem-specific question were 0.79 and 0.61, respectively, revealing a better fit for the conceptual question. This suggests students learn better by explaining domain concepts rather than providing problem-specific explanations.

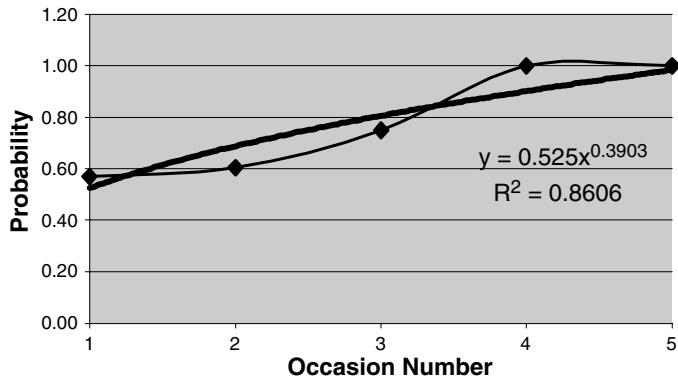


Fig. 11. Performance on the first question in the dialogues

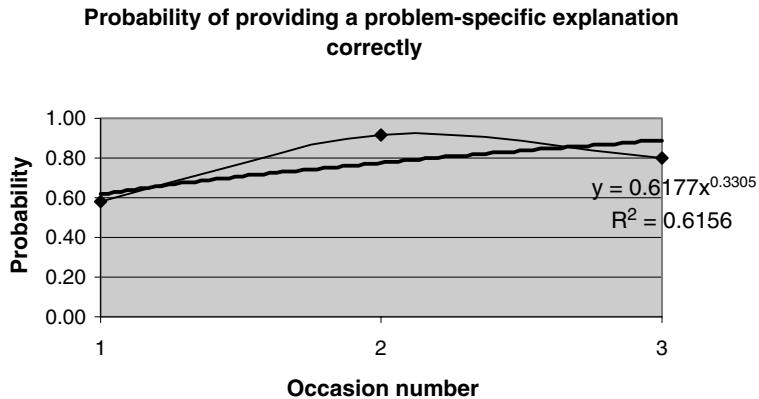


Fig. 12. Probability of providing a problem-specific explanation correctly as a function of the occasion when the student chose to provide the explanation

Subjective Analysis

A summary of the responses to the user questionnaires is given in Table 7. Students in both groups required approximately the same time to learn the interface. This was expected, as there was not much difference between the two interfaces. Students in the experimental group found it significantly easier to use the interface ($t = 2.17, p = 0.01$). We were encouraged to see that students in the experimental group felt that the interface was easier to use even though KERMIT-SE has more features than the version used by the control group. This might be due to the fact that students in the experimental group had more features in their version due to the self-explanation support provided. The difference in mean responses on the amount learnt and the enjoyment were not significant.

Table 7. Means for the questionnaire responses

| | 1 to 5 on Likert Scale | Control Group | Experimental Group | Self- explainers | Non Self- explainers |
|--|----------------------------|------------------|-----------------------|---------------------|-------------------------|
| Time to learn the interface | N/A | 14.27 (13.99) | 11.82 (11.64) | 11.67 (13.39) | 12.00 (9.59) |
| Amount learnt | Nothing to Very much | 3.35 (0.82) | 3.33 (0.78) | 3.22 (0.73) | 3.40 (0.83) |
| Enjoyment | Very much to Not at all | 2.84 (0.98) | 2.72 (1.14) | 3.12 (1.11) | 2.27 (1.03) |
| Ease of using interface | Very much to Not at all | 2.78 (0.98) | 2.34 (0.87) | 2.47 (0.94) | 2.27 (0.79) |
| Usefulness of feedback/ dialogues | Not at all to Very much | 3.45 (1.09) | 3.50 (0.82) | 3.41 (0.94) | 3.61 (0.65) |
| Ease of understanding the questions in the dialogues | Very easy to Not at all | N/A | 2.65 (0.95) | 2.50 (1.21) | 2.86 (0.53) |

Control group student found feedback to be useful, even though they only had limited feedback. Although student on average find questions in the dialogue difficult to understand, they rated the usefulness of the dialogues higher than the rank of feedback by the control group.

The user responses to the questionnaires were compared for self-explainers and non self-explainers. As expected, there was no significant difference between the time needed by the two groups to learn the interface. When asked to rate the amount learnt on a scale 1 to 5 (*Nothing* to *Very much*), non self-explainers claimed that they learnt more (the ranking of non self-explainers is higher by 0.18). Though the non self-explainers' claim seems surprising without them going through at least one dialogue, the difference was not statistically significant. Non self-explainers' claim of learning more suggests that students think problem solving using general hint messages helped them to acquire a robust knowledge of the domain concepts. This is consistent with the control groups' claim that they have learnt more than the experimental group even though they received limited feedback (Table 5.2).

When asked to rate the ease of using the interface on a scale 1 to 5 (Very much to Not at all), the non self-explainers found the interface easier to use than the self-explainers. This can be expected, because non self-explainers did not interact with some of the interface controls, as they did not use the self-explanation support. However, the difference in the mean rating was found to be statistically insignificant ($t = 0.66, p = 0.25$).

Even though non self-explainers did not go through any of the dialogues, they gave a ranking indicating the ease in understanding the questions in the dialogues and the usefulness of the dialogues to understand mistakes.

As they gave rankings for two aspects of the self-explanation support that they did not use, it is not meaningful to compare the mean ratings. However, self-explainers gave a mean rating of 3.41 for the usefulness of dialogues to understand mistakes.

68% of the control group students indicated that they would recommend KERMIT to other students while the percentage of experimental group students who had the same opinion was lower (60%). 60% of the control group students preferred more details in the feedback whereas, only 17% indicated that they do not want more details. As it is not effective to consider the response given by the non self-explainers in the experimental group about the effects of the dialogues, we consider only the response of the self-explainers.

When asked how many questions they had to go through on average to realise a mistake in their student solution, the majority (66%) of the self-explainers indicated that 2 to 3 questions were needed. Only 5% of the self-explainers indicated that they needed to go through the entire dialogue. This suggests that being able to resume problem solving in the middle of a dialogue might have increased the usability of KERMIT-SE. Moreover, 56% of the self-explainers felt that the questions in the dialogues assisted them in understanding the domain concepts.

Discussion

Results of the evaluation study indicate that students performance improved by self-explaining although not significantly. Furthermore, results reveal that students learn by explaining domain concepts and providing problem-specific explanations. As students have the freedom to initiate the self-explanation dialogues (via the *More Help* button), the experimental group cannot be considered a true experimental group. This factor can be considered an experimental flaw. However, if students were forced to go through self-explanation dialogues, it might have had a detrimental effect on their motivation to use the system for a considerable period of time. Students' use of the dialogues suggest that they were willing to self-explain while problem solving. As many students did not go through the entire dialogue before resuming problem-solving, giving the freedom to resume problem-solving at any point within the self-explanation dialogue might have improved their motivation to use the system. Students who self-explained attempted and solved significantly more problems, thus the self-explanation process seems to support problem solving.

It was encouraging to note that many students from both control and experimental groups requested access to use the system after the study to practice for exams. The systems were made available for students until the end of that semester.

6 Conclusions and Future Work

Self-explanation is an effective learning strategy to facilitate deep learning. This research focuses on incorporating self-explanation into a tutor that teaches the open-ended task of ER modelling. KERMIT-SE supports self-explanation by engaging students in tutorial dialogues about errors they make. Students are asked problem-specific and general questions, and can select answers from menus.

An evaluation study was conducted, to investigate whether guided self-explanation would facilitate acquisition of both procedural and conceptual knowledge in the domain of database modelling. The experiment involved second-year university students enrolled in an introductory database course. The experimental group used KERMIT-SE, while the control group used a cut down version of KERMIT. Both groups received the list of all errors for their solutions, and could ask for the ideal solution. Both groups received similar feedback. In addition, the experimental group had the freedom to initiate the self-explanation process. The pre- and post-tests consisted of two questions each, of equal difficulty. The first question required students to design an ER model for the given requirements, whereas the second question required them to explain the design decisions for the given ER model. The first question was used to measure their problem-solving capabilities and the second question their self-explanation abilities. The tests were rotated between successive sessions to minimize any effect resulting from variation in test difficulty.

The results showed that performance of both experimental and control groups improved. Furthermore, a significant improvement was achieved only by the control group. However, their pre-knowledge (pre-test score) was lower than that of the experimental group so they had room for improvement. Analysis of the participants' logs indicated that only 35% of the students in the experimental group have gone through at least one dialogue. We compared the performance of these students (*self-explainers*) with the other students who have not self-explained (*non self-explainers*). Although both groups improved after interacting with KERMIT-SE, the improvement was not statistically significant.

We measured the pre and post-test performance on the procedural and the conceptual questions separately for self-explainers and non self-explainers. The results indicated that the improvement in procedural knowledge of non self-explainers is better than their peers although it is not significant. This suggests that guiding students towards the ideal solution through general feedback messages provided by the cut-down version of KERMIT help them to significantly improve their procedural knowledge. However, self-explainers performed significantly better in the conceptual question in the post-test. It is interesting to note that the performance of the non self-explainers decreased after using the cut down version of KERMIT which did not provide any self-explanation support. Therefore, these results suggests that the

self-explanation support provided by KERMIT-SE has significantly contributed towards improving the conceptual knowledge of the students.

We wanted to investigate whether the system is more beneficial to less able students. We divided the self-explainers and the non self-explainers into *more* and *less* able groups based on their performance in the pre-test. The *more able* group comprised of those students who scored above the mean score for all participants and the remaining students formed the less able group. The performance of less able students in both groups improved significantly whereas there was a decrease in the performance of more able students. Therefore, the system is more beneficial to less able students.

Self-explanation has been successfully used to facilitate deep learning in several well-structured domains. This research is the first attempt to facilitate self-explanation in an open domain like database domain. The evaluation study revealed that self-explanation improved both the procedural and conceptual knowledge of students in database design. Therefore this research suggests that self-explanation can be successfully used to facilitate deep learning even in open-ended domains although it is highly challenging to define the self-explanation for open-ended tasks.

There are a number of future avenues that can be explored to further improve the effectiveness of KERMIT-SE. Currently, the system provides the same dialogue for the same error for two different students who could possibly have different self-explanation skills. We believe that the effectiveness of the dialogues could be greatly enhanced by incorporating adaptive self-explanation support for students. Currently the short-term model student model of KERMIT consists of lists of satisfied and violated constraints for the student's last submission, while the long-term model records the history of each constraint (how often a constraint has been relevant, and how often it has been satisfied/violated). The current student model describes the knowledge level of each student using constraints. As a result, these models can be used to understand how each student acquires domain knowledge, which is represented as a set of constraints. In order to facilitate adaptive self-explanation skill, the current student model needs to be extended to record students' self-explanation behaviour. The enhanced student model can also be used to provide additional support in acquiring domain knowledge to students who have difficulty in understanding domain concepts.

Providing the opportunity to self-explain in a natural language might potentially enhance the quality of learning as natural language is more expressive than a restricted interface of an ITS used to obtain self-explanations from a student. In addition, it also provides an opportunity to express partial knowledge using natural language. However, a recent evaluation study revealed that students who explained in natural language did not learn better than those who self-explained through menu selection [2]. Incorporating natural language capabilities to KERMIT-SE might be an avenue worth exploring to investigate whether learning can be significantly enhanced using natural language to self-explain in an open-ended domain like database design.

Acknowledgements

We thank Pramuditha Suraweera and Danita Hartley for their help in implementing KERMIT-SE. This research was made possible by the NZODA postgraduate scholarship awarded to the first author.

References

1. V. Aleven, K. R. Koedinger, and K. Cross, Tutoring Answer Explanation Fosters Learning with Understanding, *Artificial Intelligence in Education* (1999), 199–206. [105](#), [107](#)
2. V. Aleven, K. Koedinger and O. Popescu, A Tutorial Dialogue System to Support Self-Explanation: Evaluation and Open Questions, in: U. Hoppe, F. Verdejo and J. Kay ed., Proc. AIED 2003, IOS Press, pp. 39–46. [107](#), [140](#)
3. V. Aleven, O. Popescu, and K. R. Koedinger, Towards Tutorial Dialogue to Support Self-Explanation: Adding Natural Language Understanding to a Cognitive Tutor, *Int. Journal on Artificial Intelligence in Education* 12 (2001), 246–255. [107](#)
4. V. Aleven, O. Popescu and K. R. Koedinger, Pilot-Testing a Tutorial Dialogue System that Supports Self-Explanation, in: Proc. Int. Conf. Intelligent Tutoring Systems, Biarritz, France, 2002, pp. 344–354. [107](#), [121](#)
5. J. R. Anderson, Rules of the mind, 1993. [123](#)
6. J. R. Anderson, A. T. Corbett, K. R. Koedinger, and R. Pelletier, Cognitive Tutors: Lessons Learned. *The Journal of Learning Sciences* 4 (1995), 167–207. [109](#), [121](#)
7. K. Bielaczyc, P. Pirolli, and A. L. Brown, Training in Self-Explanation and Self-Regulation Strategies: Investigating the Effects of Knowledge Acquisition Activities on Problem-Solving. *Cognition and Instruction* 13 (2) (1995), 221–252. [105](#)
8. A. Bunt, and C. Conati, Modeling Exploratory Behaviour, in: M. Bauer, P. J. Gmytrasiewicz, and J. Vassileva, ed., Proc. of 8th International Conference, User Modeling, Sonthofen, Germany, 2001, pp. 219–221. [112](#)
9. M. T. H. Chi, Self-explaining Expository Texts: The Dual Processes of Generating Inferences and Repairing Mental Models, *Advances in Instructional Psychology*, (2000) 161–238. [105](#)
10. M. T. H. Chi, M. Bassok, W. Lewis, P. Reimann, and R. Glaser, Self-Explanations: How Students Study and Use Examples in Learning to Solve Problems. *Cognitive Science*, 13 (1989), 145–182. [105](#)
11. M. T. H. Chi, N. De Leeuw, M. Chiu and C. Lavancher, Eliciting self-explanations improves understanding. *Cognitive Science*, 18 (1994), 439–477. [105](#)
12. C. Conati and K. VanLehn, Toward Computer-Based Support of Meta-Cognitive Skills: a Computational Framework to Coach Self-Explanation, *Int. J. Artificial Intelligence in Education*, 11 (2000), 389–415. [106](#)
13. C. Conati and K. VanLehn, Providing Adaptive Support to the Understanding of Instructional Material, in Proc. IUI 2001 Sante Fe, New Mexico. [106](#), [122](#)
14. A. T. M. Corbett, H. J. Trask, K. C. Scarpinatto and W. S. Handley, A Formative Evaluation of the PACT Algebra II Tutor: Support for Simple Hierarchical Reasoning, Proc. ITS'98, pp. 374–383. [105](#)
15. K. A. Ericsson and H. A. Simon, Protocol Analysis: Verbal Reports as Data, 1984.

16. A. S. Gertner and K. VanLehn, ANDES: A Coached Problem-Solving Environment for Physics, in: Proc. ITS 2000, G. Gauthier, C. Frasson, and K. VanLehn, ed., Montreal 2000, New York : Springer, pp. 133–142. [105](#), [106](#)
17. A. C. Grassler, K. VanLehn, C. P. Rose, P. W. Jordan and D. Harter, Intelligent Tutoring Systems with Conversational Dialogue, *AI Magazine*, Winter (2001), 39–51. [106](#)
18. A. C. Grassler, K. Wiemer-Hastings, P. Wiemer-Hastings and R. Kreuz,, Tutoring Research Group 1999. AUTOTUTOR: A Simulation of a Human Tutor. *Journal of Cognitive Systems Research* 1(1) (1999), 35–51. [105](#), [107](#)
19. T. K. Landauer, P. W. Foltz and D. Laham, An Introduction to Latent Semantic Analysis. *Discourse Process* 25 (2–3) 1998, 259–284. [107](#)
20. A. Mitrovic, Investigating Students' Self-assessment Skills, in: Proc. UM 2001, M. Bauer, P. J. Gmytrasiewicz and J. Vassileva, ed., Berlin Heidelberg, 2001, Springer-Verlag, pp. 247–250. [105](#)
21. A. Mitrovic, NORMIT, a Web-enabled Tutor for Database Normalization, in: Proceedings of the Eighth International Conference on Computers in Education, Kinshuk, R. Lewis, K. Akahori, R. Kemp, T. Okamoto, L. Henderson, and C. H. Lee, ed., Auckland, 2002, IEEE Computer Society, pp. 1276–1280. [107](#), [108](#)
22. A. Mitrovic and S. Ohlsson, Evaluation of a Constraint-Based Tutor for a Database Language *International Journal on AIED* 10 (3–4), 1999, 238–256. [109](#), [123](#)
23. A. Mitrovic, P. Suraweera, B. Martin, and A. Weerasinghe, DB-suite: Experiences with Three Intelligent, Web-based Database Tutors, *Journal of Interactive Learning Research (JILR)*, 15 (4), 409–432. [105](#)
24. M. Negoita, C. D. Neagu and V. Palade, Computational Intelligence-Based Engineering of Intelligent Hybrid Systems, Springer-Verlag, 2005. [105](#)
25. S. Ohlsson, Constraint-based Student Modelling, in: Proc. of Student Modelling: the Key to Individualized Knowledge-based Instruction, J.E. Greer and G. McCalla, ed., Springer-Verlag Berlin, 1994, pp.167–189. [109](#)
26. S. Ohlsson, Learning from Performance Errors. *Psychological Review* 103 (2), (1996), 241–262. [109](#)
27. P. Suraweera, An Intelligent Teaching System for Database Modelling, M.Sc. Thesis, University of Canterbury, 2001. [109](#), [123](#)
28. P. Suraweera and A. Mitrovic, KERMIT: a Constraint-based Tutor for Database Modelling. in: Proc. ITS'2002, S. Cerri, G. Gouarderes and F. Paraguacu eds., Biarritz, France, LCNS 2363, 2002, pp. 377–387. [106](#)
29. P. Suraweera, and A. Mitrovic, An Intelligent Tutoring System for Entity Relationship Modelling, *Int. J. Artificial Intelligent in Education* 14 (3–4) (2004), 375–417. [106](#), [113](#)
30. K. VanLehn, et al. Fading and Deepening: The Next steps for ANDES and Other Model-Tracing Tutors, in: Proc. ITS 2000, G. Gauthier, C. Frasson, and K. VanLehn, ed., Montreal (2000), pp. 474–483. [106](#)
31. K. VanLehn, R. M. Jones, and M. T. H. Chi, A Model of the Self-Explanation Effect. *The Journal of Learning Sciences* 2 (1) 1992, 1–59. [105](#)
32. A. Weerasinghe, Exploring the Effects of Self-Explanation in the Context of a Database Design Tutor, M.Sc. Thesis, University of Canterbury, 2003. [112](#), [124](#)

Appendix A

Part I – Problem Set Used in the Informal Study

1. For each course a student has taken, we need to know the final grade. Each course has a unique course code.
2. Each course with a unique course code, consists of several sections. For each section, we know the topic it covers and the no of lectures and labs it contains. The same topic can be covered in several courses, but the number of lectures and labs will differ in that situation.

Part II – Problem Sets Used for the Pilot Study

Problems Used in Version A of the System Prototype

1. For each course a student has taken, we need to know the final grade. Each course has a unique course code and a student has his/her student id.
2. A text book, with a unique ISBN, contains a number of chapters. For each chapter we know its chapter number, topic, the number of pages and the number of references. Different textbooks may cover the same topic.
3. You have been asked to design a database for a small library. The database needs to store data about various branches of the library and about books the library holds. There are only three branches and their ids are ENG for the Engineering Branch, CEN for the Central Branch and SCI for the Science branch. Branch names and addresses must also be known. The database needs to record the book id, title, publisher, and the year the book was published for each book in the library. A book may have several authors. Each copy of a book is given a serial number. The number of copies of a book held in a specific branch of the library should also be known. The maximum number of copies of a book in the library is 20.

Problems Used in Version B of the System Prototype

1. Sometimes student work in groups. Each group has a unique number and students have their student ids. A student may have different roles in various groups he/she belongs to.
2. Each course, with a unique course code consists of several sections. For each section, we know the topic it covers, number of lectures it covers and labs it contains. The same topic can be covered in several courses, but the number of lectures and labs will differ in that situation.

3. You are chosen to design a database for the University Accommodation office. Each hall of residence has a name, number, address, phone number and a hall manager. The halls provide only single rooms, each with a room number (unique within a hall) and a weekly rent. The total number of rooms should also be available. For each student currently renting a room, we store ID number, name, home address, date of birth and the category of student (for example, 1UG for the first year undergraduate student). Whenever possible, information on one or more next-of-kin related to a student is stored, which includes the name, relationship, address and contact phone.

Appendix B: Tutorial Dialogue that is Initiated When a Weak Entity is Modelled as a Regular Entity

Consider the dialogue given in Figs. 13(a), 13(b) and 13(c). This dialogue aims to discuss the error *The CHAPTER entity type should be represented as a weak entity* in the student solution in Fig. 2. The dialogue is presented in tree form, in which each distinct node has a number, a question and a list of possible explanations. The numbers are assigned by traversing the tree in a top-to-bottom, left-to-right manner. The possible explanations are given in italic font and the correct one is given in bold italic font. The explanations are separated by semi-colons and phrases that need to be separated within an explanation are separated by commas (Node (7) in Fig. 13(b)). Some nodes have only a number to indicate that particular node is identical to the node specified by the number. The tutorial dialogue ideally starts from the root node and finishes at one of the leaf nodes.

The specific path taken by the pedagogical module depends on the correctness of the explanation selected by the student for each prompt. In other words, if the explanation selected is correct, then the pedagogical module selects the question associated with the left child of the current node. If it is incorrect, the pedagogical module selects the question associated with the right child of the current node.

(1) You've modelled CHAPTER as a regular entity. Can you tell me the general rule to decide whether something is a regular entity?

If an entity has one or more attributes, then it should be a regular entity; If an entity has a key attribute then it should be a regular entity; I don't know

Selected answer is incorrect

(2) Good! Now can you tell me why modelling CHAPTER as a regular entity is incorrect?

Because it doesn't have a key attribute; Because it doesn't have attributes; I'm not sure

Selected answer is correct

(4) Great! Now can you tell me the best way to model CHAPTER?
a weak entity; an attribute; regular relationship

(2) Now can you tell me why modelling CHAPTER as a regular entity is incorrect?

Ok! Let's try a simple thing. What is used to uniquely identify instances of a regular entity? attributes; The key attribute; I want more help

Selected answer is correct

(3) Ok! Let's try a simple thing. What is used to uniquely identify instances of a regular entity? attributes; The key attribute; I want more help

Selected answer is incorrect

(2)

(6) This is a fundamental concept you need to remember.
 A regular entity should have at least one candidate key attribute OK! now tell me why modelling CHAPTER as a regular entity is incorrect?

CONTINUE
button is pressed

[Continued in Figure 13(b)]

(9) Now can you tell me the attribute that you have identified as a key attribute for 'CHAPTER'?

(4)

(5)

Chapter_No; Topic; There's no attribute that can be a key attribute for CHAPTER

[Continued in Figure 1(c)]

Fig. 13(a). Part I of the tutorial dialogue that is initiated when a weak entity is modelled as a regular entity

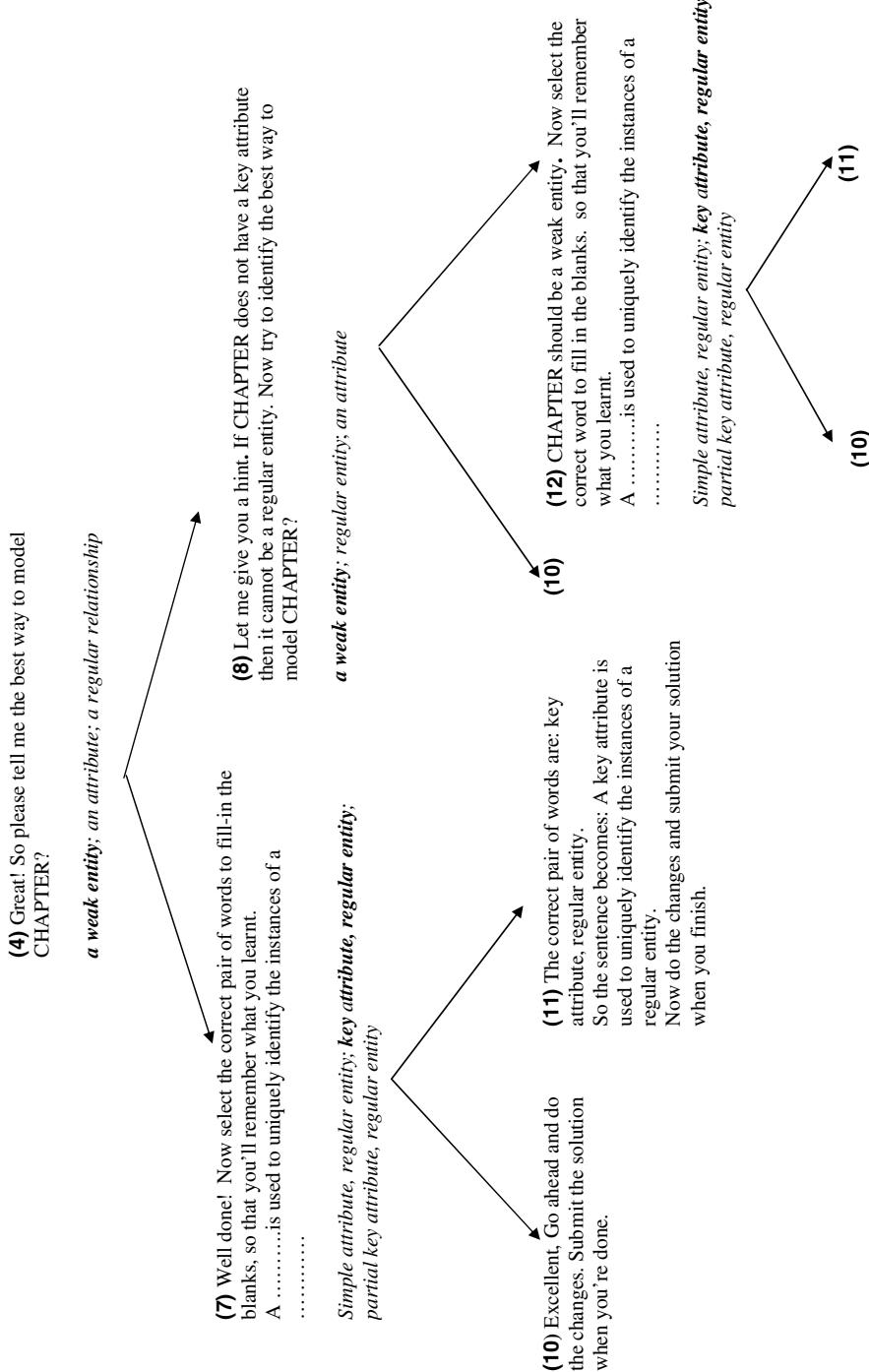


Fig. 13(b). Part II of the tutorial dialogue that is initiated when a weak entity is modelled as a regular entity

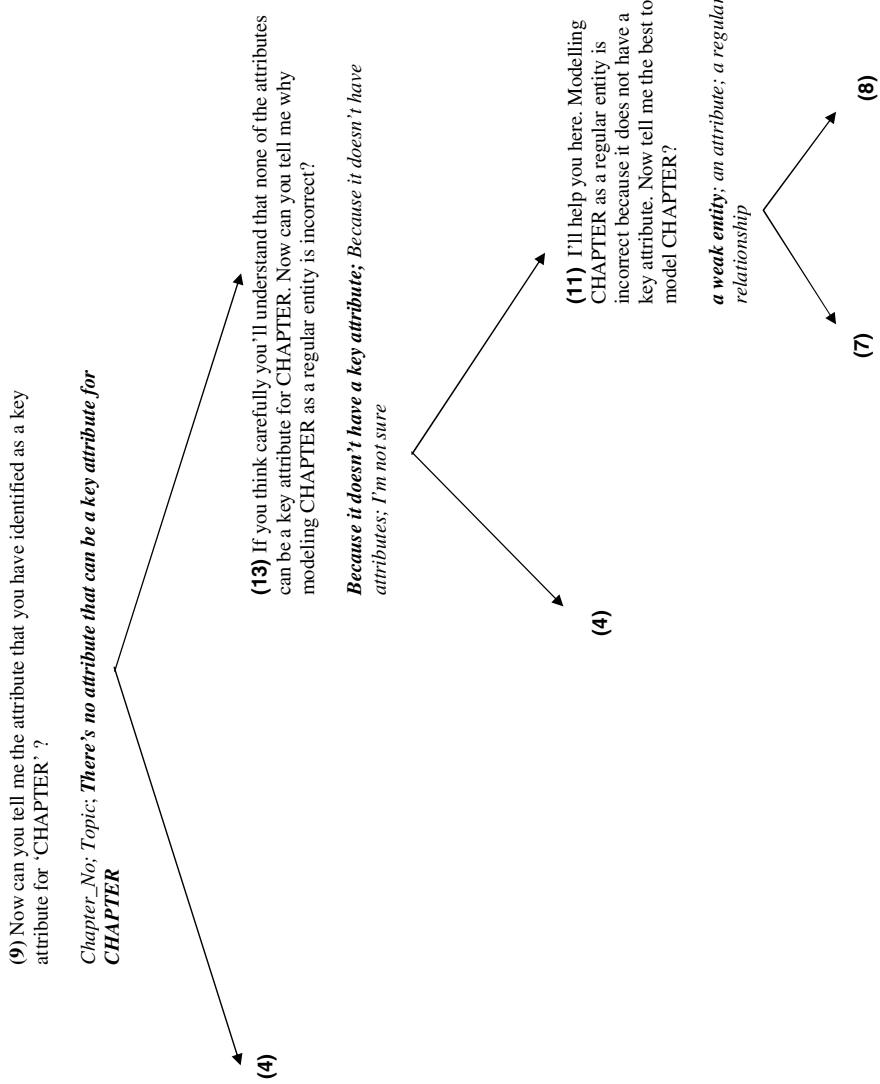


Fig. 1.3(c). Part III of the tutorial dialogue that is initiated when a weak entity is modelled as a regular entity

Appendix C: Pre- and Post-tests Used in the Evaluation Study

Test A

Post – Test

Your user code :

Answer all Questions

1. Please draw an ER diagram that captures the information given below.
2. For each course, a student has taken, we need to know the final grade. Each course has a unique course code and a student has his/her student id.
3. The ER diagram, which captures the information below, is given in Fig. 1A. Please answer *all* the questions given below based on the ER diagram and the information.

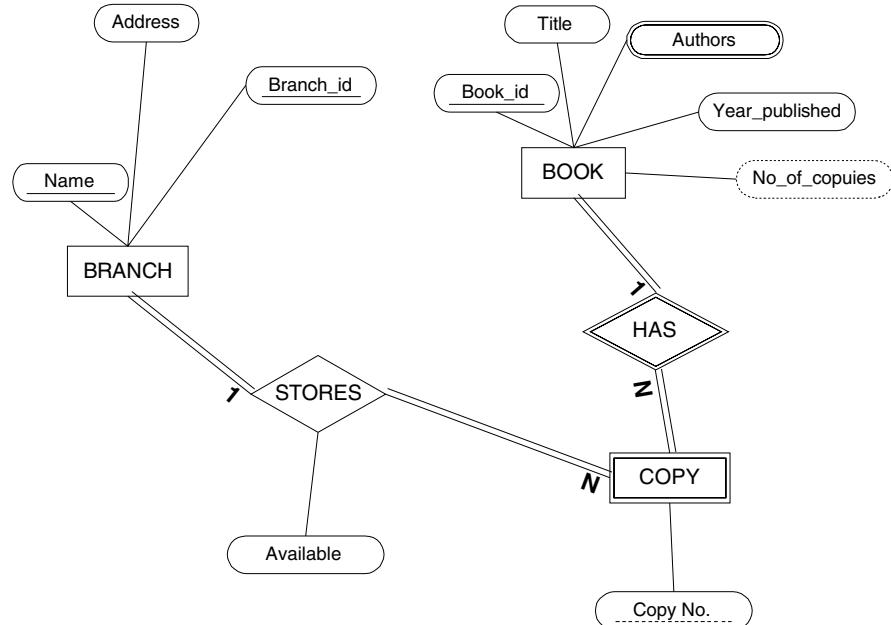


Fig. 1A.

You have been asked to design a database for a small library. The database needs to store data about various branches of the library, and about books the library holds. There are only three branches and their ids are ENG for

the Engineering branch, CEN for the Central branch ad SCI for the Science branch. Branch names and address must also be known. The database needs to record the book id, title, publisher and the year the book was published for each book in the library. A book may have several authors. Each copy of a book is given a serial number. The number of copies of a book held in a specific branch of the library and the availability of each copy should also be known. The maximum number of copies of a book in the library is 20.

1. Explain why Library is **not** modelled as an entity.

.....
.....
.....
.....

2. Explain why the *Authors* attribute (connected to the *Book entity*) is modelled as a multi-valued attribute?

.....
.....
.....
.....

3. Explain why **both** *Branch_id* and *Name* are represented as key attributes of the entity *Branch*.

.....
.....
.....

4. Explain why *N* has been indicated with the *Copy* entity in the *Stores* relationship?

.....
.....
.....
.....

5. Explain why the participation of the *Book entity* in the *Has* relationship is total?

.....
.....
.....
.....

Test B

Pre-Test

Your user code :

Answer all Questions

1. Please draw an ER diagram that captures the information given below.

Each course with a unique course code, consists of several sections. For each section, we know the topic it covers, and the number of lectures and labs it contains. The same topic can be covered in several courses, but the number of lectures and labs will differ in that situation.

Your user code :

2. The ER diagram, which captures the information below, is given in Fig. 1B. Please answer ***all*** the questions given below based on the ER diagram and the information.

Design the HOTELS database based on the following requirements. Each hotel has a name, code, address, and a category (the number of stars). The total number of beds is also known. Each hotel has many rooms, each with a

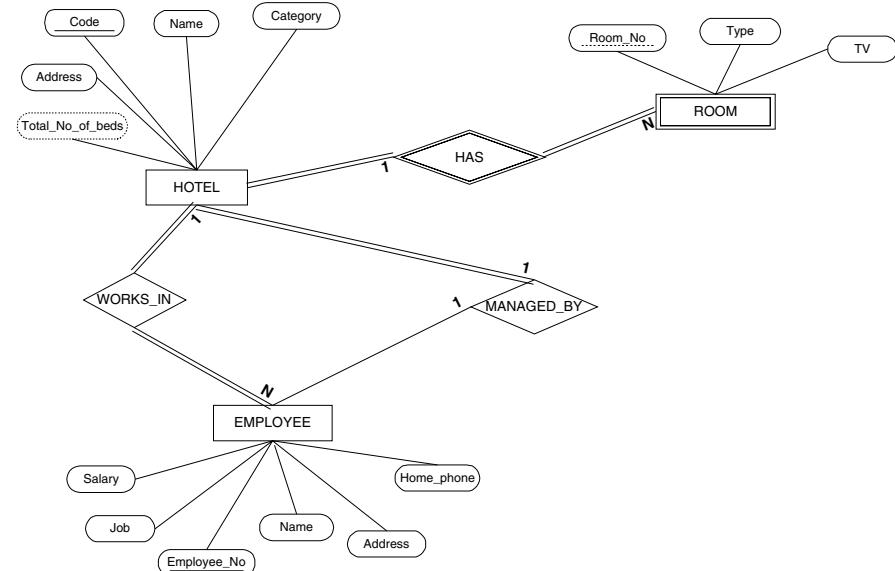


Fig. 1B.

room number. Rooms may be of two types (single or double). In some rooms there are TVs. For each employee, we know his/her employee number, name, address, home phone number, job position and salary. Each employee works for just one hotel, and each hotel is managed by one employee.

1. Explain why *Room* is modelled as a weak entity.

.....
.....
.....
.....

2. Explain why the *Total_No_of_beds attribute* (connected to the *Hotel* entity) is modelled as a derived attribute?

.....
.....
.....
.....

3. (i) Name a partial key in the given ER diagram

- (ii) What is the difference between a partial key and a key attribute?

.....
.....
.....
.....

4. Explain why 1 has been indicated with the *Hotel* entity in the relationship *Has*?

.....
.....
.....

5. Explain why the participation of the *Employee* entity in the *Managed_by* relationship is partial?
-
-
-
-

Data Driven Fuzzy Modelling with Neural Networks

C. Moraga

Abstract. Extraction of models for complex systems from numerical data of behavior is studied. In particular, systems representable as sets of fuzzy if-then rules where the premises are not connected by t -norms, but by a compensating aggregation operator are discussed. A method is presented to extract this kind of fuzzy rules with support of neural networks. Finally it is shown that it is possible to extract compensating fuzzy if-then rules from a great number of already existing feedforward neural networks.

1 Introduction

The extraction of models for systems of the real world from data of behaviour may be considered as a knowledge acquisition problem. The goal is to obtain models that are *both* understandable *and* accurate enough for a given application. The subject has received much attention from the research community in the last 15 years with the aim of combining the learning capability of neural networks with the expressiveness of fuzzy if-then rules using linguistic variables. This kind of modeling has a two-fold purpose: First, to provide a model that allows to control and possibly to forecast the behaviour of the underlying unknown system. Second, to provide a model to better understand the system. The former, sets demands on accuracy to meet the requirements of a given application. The latter, demands that the rules be simple to interpret (see e.g. [5]). Emphasis in obtaining a high numerical accuracy may deteriorate the interpretability of the rules (see e.g. [22, 27]). Much work done in this area has been strongly influenced by the pioneering work of Jang [11], who introduced the ANFIS system. It is fair to mention however, that in that same period of time, several other research groups were developing similar ideas (see e.g. [8, 9, 10, 12, 13, 25, 28]). The important contribution of ANFIS is the idea of expressing as net architecture, the main components of a fuzzy inference system: fuzzification, implication and (if needed) defuzzification. Nodes of the first hidden layer realize the linguistic terms of the linguistic variable

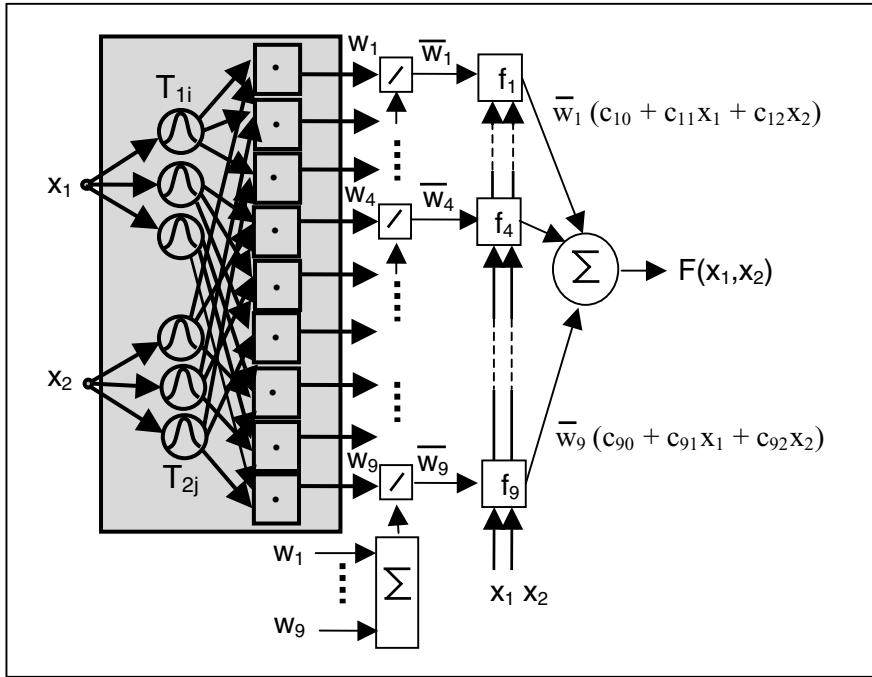


Fig. 1. Structure of an ANFIS system to extract rules of type Takagi-Sugeno. *Shaded* in grey is shown the part of the network to learn the membership functions of the linguistic terms

associated to a given external input. For this purpose they have a bell-shaped (or triangular/trapezoidal shaped) activation function, where the parameters to be adjusted (from the neural net point of view) determine the width and centre of the bell representing a linguistic term (from the fuzzy sets point of view), respectively. Figure 1 shows the front end of an ANFIS system with two input variables having three linguistic terms each. The system extracts if-then fuzzy rules of type Takagi-Sugeno [26]. It has been shown [3] that other back ends for ANFIS are possible in order to obtain if-then rules of other types.

If the training of the neural network is done with a gradient descent algorithm, then all elementary functions along an input-output path have to be differentiable with respect to the parameters under adjustment. Particularly, nodes at the second hidden layer, which represent the conjunction of the degrees of satisfaction of the premises, must have a differentiable t -norm as transfer function. Almost all applications of fuzzy logic use continuous t -norms. It is well known [14, 19] that if t is a continuous t -norm, then t is the Minimum operation, t belongs to the family of the Product or to the family of Lukasiewicz or is an ordinal sum. From the t -norms used in fuzzy logic,

only t -norms of the family of the Product are differentiable. Any t -norm of the family of the Product is obtained as follows: $\forall x, y \in [0, 1]$

$$t_\varphi(x, y) = (\varphi(x)) \cdot (\varphi(y))$$

where φ is an order automorphism of $([0,1], <)$.

In ANFIS the product operation was chosen to connect the degrees of satisfaction of the premises. After proper training of the free parameters of the system as a neural network, their values will specify the corresponding parameters of the associated fuzzy if-then-rule. Even though this kind of system has proven to be able to deliver very accurate models, this is often achieved by giving up the understandability of the rules. Fuzzy sets that should represent linguistic terms exhibit after training strong and complex overlaps -(possibly due to an unfortunate initial choice with too few linguistic terms)- impairing a simple interpretation. A *constrained* training algorithm should be used to tune the neural networks with controlled overlap. This is a question that continues to be an area of research (see e.g. [21, 24]).

Even though ANFIS deserves recognition as a milestone in the development of neuro-fuzzy systems, its limitations must be also clearly stated. ANFIS-like systems require from the user an a priori decision on the number of linguistic terms per variable, they only extract rules where the premises are connected through *differentiable* t -norms and generate as many rules as the product of the number of linguistic terms per variable. For instance if a system has 5 input variables and the associated linguistic variables have only 3 terms each, then any ANFIS-like system would generate $3^5 = 243$ rules. It becomes apparent that this amount of rules is a great burden on the understandability of the model. The number of rules may be strongly reduced if a cluster-oriented neuro-fuzzy system is used, (see e.g. [1]). In this case the understandability of each rule may become more difficult, since a possibly non-linear combination of input values is associated to a linguistic term; but this is compensated by the reduced number of rules needed to model a system. It is interesting to mention that this strategy may be related back to work done by Jang and Sun [12], who proved the functional equivalence between RBF neural networks and fuzzy inference systems, however not following this idea to integrate it in ANFIS.

2 Compensating Systems

Consider the following rule of decision for people who like to read, with respect to buying a book:

*"If a book has received an important award
and the price is acceptable then buy it!"*

Furthermore, assume that the following simple fuzzy logic model is used. Linguistic variables representing the premises are assigned degrees of membership between 0.5 and 1 when they are increasingly getting close to satisfy the assertion and between 0.5 and 0 as they get increasingly far from the acceptable “reasonable minimum” (which is graded with 0.5). As a conjunction for the premises the *t-norms product* and *minimum* will be used, since they represent “everyone’s first choice”. Finally, to avoid defuzzification problems, the simplest of the Takagi-Sugeno systems is chosen: the joint “strength” of the premises is taken as the strength of the conclusion.

Now, consider the case of two books: the first one has been awarded a Pulitzer prize, from where $\mu_{\text{award}}(1^{\text{st}} \text{ book}) = 0.98$ meanwhile the second only has the recommendation of a local newspaper in a middle-size town, i.e., $\mu_{\text{award}}(2^{\text{nd}} \text{ book}) = 0.6$. The price of both books is however the same, and very close to the limit of the available budget: $\mu_{\text{good-price}}(1^{\text{st}} \text{ book}) = \mu_{\text{good-price}}(2^{\text{nd}} \text{ book}) = 0.51$.

Let $\sigma(\text{book})$ denote the strength with which a book satisfies the conditions of the rule. Then

$$\sigma(1^{\text{st}} \text{ book}) = t(\mu_{\text{award}}(1^{\text{st}} \text{ book}), \mu_{\text{good-price}}(1^{\text{st}} \text{ book}))$$

and

$$\sigma(2^{\text{nd}} \text{ book}) = t(\mu_{\text{award}}(2^{\text{nd}} \text{ book}), \mu_{\text{good-price}}(2^{\text{nd}} \text{ book}))$$

If t is taken to be the *t-norm product*,

$$\begin{aligned}\sigma(1^{\text{st}} \text{ book}) &= \mu_{\text{award}}(1^{\text{st}} \text{ book}) \cdot \mu_{\text{good-price}}(1^{\text{st}} \text{ book}) = 0.98 \cdot 0.51 = 0.4998 \\ \sigma(2^{\text{nd}} \text{ book}) &= \mu_{\text{award}}(2^{\text{nd}} \text{ book}) \cdot \mu_{\text{good-price}}(2^{\text{nd}} \text{ book}) = 0.6 \cdot 0.51 = 0.306\end{aligned}$$

Since in both cases the value of σ is below 0.5, the rule suggests that no book should be bought. This is obviously not the way an interested reader would use the rule.

If the *t-norm minimum* had been chosen, it is fairly simple to see that the rule would be suggesting a “blind choice”, since in this case $\sigma(1^{\text{st}} \text{ book}) = \sigma(2^{\text{nd}} \text{ book}) = 0.51$. This is also not the way a human being would use the rule.

Even more, since for any *t-norm* holds that

$$\begin{aligned}t(\mu_{\text{award}}(1^{\text{st}} \text{ book}), \mu_{\text{good-price}}(1^{\text{st}} \text{ book})) \\ \leq \text{minimum}(\mu_{\text{award}}(1^{\text{st}} \text{ book}), \mu_{\text{good-price}}(1^{\text{st}} \text{ book}))\end{aligned}$$

and similarly for the second book, the situation would not change by trying to use a “more appropriate” *t-norm*. (There is no “more appropriate” *t-norm*)

Why does the model fail in this case, meanwhile there are so many other cases (particularly in applications of fuzzy control) in which this very same model leads to quite convenient decisions?

The example stated above represents a special kind of situation, which here will be called *compensating*, where the user is open to compromises and may

even use implicit priorities when applying the rule. This cannot be modeled with t -norms, which are “too drastic” for this situation.

Compensating rules may be modeled by using *aggregations* to connect the premises. A function $a: [0,1]^n \rightarrow [0,1]$ that satisfies the following conditions:

- (i) a is continuous in all variables
- (ii) $a(0, \dots, 0) = 0$ and $a(1, \dots, 1) = 1$
- (iii) if $x_1 \leq y_1, \dots, x_n \leq y_n$, then $a(x_1, \dots, x_n) \leq a(y_1, \dots, y_n)$

is an aggregation [4, 7].

One of the earliest proposals to use aggregation operators in fuzzy rules may be traced back to [29], where the “ γ -operator” was introduced, with the parameter γ controlling a linear combination of a t -norm and a t -conorm (see below). Considering the associativity of t -norms and t -conorms, let for a while $t(x_1, \dots, x_n)$ denote $t(\dots t(t(x_1, x_2), x_3) \dots, x_n)$. It is simple to see that t -norms, t -conorms and any normalized linear combination thereof satisfy the three conditions given above and are therefore aggregations. t -norms and t -conorms are, however, *extreme* aggregations. Compensating rules use aggregations between t -norms and t -conorms.

The γ -operator a_γ is defined as follows for all x_1, x_2 in $[0,1]$

$$a_\gamma(x_1, x_2) = \gamma t(x_1, x_2) + (1 - \gamma)t^*(x_1, x_2)$$

where t denotes a t -norm, t^* its dual t -conorm and $\gamma \in [0, 1]$.

Notice that the γ -operator satisfies the following boundaries:

$$t(x_1, x_2) \leq a_\gamma(x_1, x_2) \leq t^*(x_1, x_2)$$

In [29], t was chosen as the product and t^* as the so called algebraic sum: $t^*(x_1, x_2) = x_1 + x_2 - x_1 x_2$.

In order to apply the γ -operator to the rule for buying a book, some preliminary calculations will be done:

$$\begin{aligned} \mu_{\text{award}}(\text{1}^{\text{st}} \text{ book}) \cdot \mu_{\text{good-price}}(\text{1}^{\text{st}} \text{ book}) &= 0.98 \cdot 0.51 = 0.4998 \\ \mu_{\text{award}}(\text{1}^{\text{st}} \text{ book}) + \mu_{\text{good-price}}(\text{1}^{\text{st}} \text{ book}) \\ - \mu_{\text{award}}(\text{1}^{\text{st}} \text{ book}) \cdot \mu_{\text{good-price}}(\text{1}^{\text{st}} \text{ book}) &= 0.98 + 0.51 \\ - 0.98 \cdot 0.51 &= 1.49 - 0.4998 = 0.9902 \\ \mu_{\text{award}}(\text{2}^{\text{nd}} \text{ book}) \cdot \mu_{\text{good-price}}(\text{2}^{\text{nd}} \text{ book}) &= 0.6 \cdot 0.51 = 0.306 \\ \mu_{\text{award}}(\text{2}^{\text{nd}} \text{ book}) + \mu_{\text{good-price}}(\text{2}^{\text{nd}} \text{ book}) \\ - \mu_{\text{award}}(\text{2}^{\text{nd}} \text{ book}) \cdot \mu_{\text{good-price}}(\text{2}^{\text{nd}} \text{ book}) \\ = 0.6 + 0.51 - 0.6 \cdot 0.51 &= 1.11 - 0.306 = 0.804 \end{aligned}$$

Let γ be chosen to be 0.5. Then:

$$\begin{aligned}
\sigma(1^{\text{st}} \text{ book}) &= a_\gamma(\mu_{\text{award}}(1^{\text{st}} \text{ book}), \mu_{\text{good-price}}(1^{\text{st}} \text{ book})) \\
&= 0.5 t(\mu_{\text{award}}(1^{\text{st}} \text{ book}), \mu_{\text{good-price}}(1^{\text{st}} \text{ book})) \\
&\quad + 0.5 t^*(\mu_{\text{award}}(1^{\text{st}} \text{ book}), \mu_{\text{good-price}}(1^{\text{st}} \text{ book})) \\
&= 0.5 \cdot (0.4998 + 0.9902) = 0.5 \cdot 1.49 = 0.745
\end{aligned}$$

and

$$\begin{aligned}
\sigma(2^{\text{nd}} \text{ book}) &= a_\gamma(\mu_{\text{award}}(2^{\text{nd}} \text{ book}), \mu_{\text{good-price}}(2^{\text{nd}} \text{ book})) \\
&= 0.5 t(\mu_{\text{award}}(2^{\text{nd}} \text{ book}), \mu_{\text{good-price}}(2^{\text{nd}} \text{ book})) \\
&\quad + 0.5 t^*(\mu_{\text{award}}(2^{\text{nd}} \text{ book}), \mu_{\text{good-price}}(2^{\text{nd}} \text{ book})) \\
&= 0.5 \cdot (0.306 + 0.804) = 0.5 \cdot 1.11 = 0.555
\end{aligned}$$

Now the rule expresses a clear recommendation to buy the Pulitzer-Prize book instead of the second one. The “moderate enthusiasm” of the recommendation may be interpreted as a consequence of the high price of the book (close to the budget limit).

Notice that in this case the aggregation satisfies tighter boundaries, i.e.:

$$t(x_1, x_2) < \min(x_1, x_2) < a_\gamma(x_1, x_2) < \max(x_1, x_2) < t^*(x_1, x_2)$$

where x_1 represents $\mu_{\text{award}}(\text{book})$ and x_2 , $\mu_{\text{good-price}}(\text{book})$. This is a common feature of compensating rules.

Is it possible to use a neuro-fuzzy system to learn from data of behaviour the appropriate value of γ besides tuning the parameters that will specify the linguistic terms of the linguistic variables?

Figure 2 shows a simple modification of the front-end of an ANFIS-like system, thus allowing to learn γ from data of behaviour.

The modification comprises the following changes:

- (i) The neurons of the second hidden layer compute both the product and the sum of the degrees of satisfaction of the linguistic terms of both input variables. There are no parameters to be adjusted and both operations are differentiable.
- (ii) A new hidden layer with an adder-subtractor is included. This computes the algebraic sum of the degrees of satisfaction of the linguistic terms of both input variables. There are no parameters to be adjusted. The operation is differentiable.
- (iii) One adder-subtractor is introduced in an additional hidden layer, to add the product of the degrees of satisfaction of the inputs weighted by γ and the algebraic sum of the same degrees of satisfaction weighted by $(1 - \gamma)$, thus completing the aggregation. The operation is differentiable and γ is the parameter that will be adjusted to minimize an error function, based on the training data.

A similar approach has been discussed in [18] to learn uninorms and generalized λ -means.

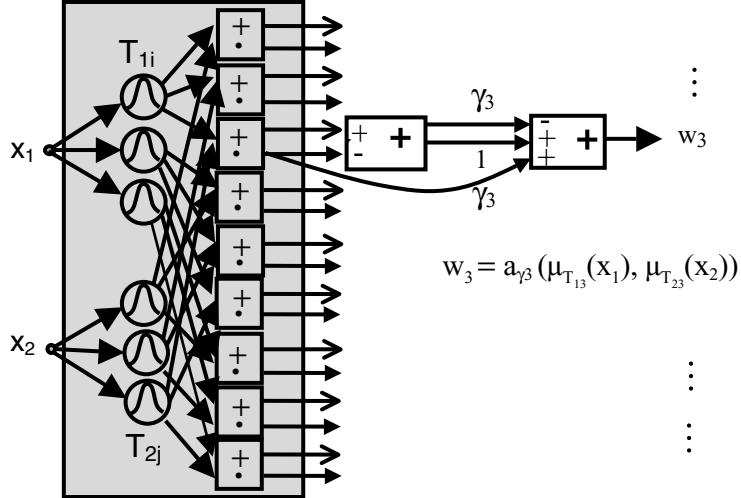


Fig. 2. Modified front-end of an ANFIS-like system to learn the γ parameter of the γ -operator for a compensating rule (Only the case of the third rule is shown)

In what follows it will be shown that a standard feedforward neural network with one hidden layer with neurons having a sigmoide as activation function and a linear output neuron is a neuro-fuzzy system to learn fuzzy if-then rules based on the symmetric summation. This method was disclosed in [2] and extended to other similar activation functions in [20].

Consider a single neuron with n inputs x_1, x_2, \dots, x_n , n input weights w_1, \dots, w_n and a sigmoide f as activation function. Let h denote the transfer function of the neuron. Then:

$$\begin{aligned} h(x_1, x_2, \dots, x_n) &= f \left(\sum_{i=1}^n w_i x_i \right) = \frac{1}{1 + e^{-(\sum_{i=1}^n w_i x_i)}} \\ &= \frac{1}{1 + \prod_{i=1}^n e^{-w_i x_i}} \end{aligned} \quad (1)$$

$$\begin{aligned} \text{From } f(y) = \frac{1}{1 + e^{-y}} \text{ it is simple to see that } e^{-y} \\ &= \frac{1}{f(y)} - 1 = \frac{1 - f(y)}{f(y)} \end{aligned} \quad (2)$$

Introducing (2) in (1) leads to:

$$h(x_1, x_2, \dots, x_n) = \frac{1}{1 + \prod_{i=1}^n \frac{1 - f(w_i x_i)}{f(w_i x_i)}} = \frac{\prod_{i=1}^n f(w_i x_i)}{\prod_{i=1}^n f(w_i x_i) + \prod_{i=1}^n (1 - f(w_i x_i))} \quad (3)$$

Define

$$f(w_1x_1) \otimes f(w_2x_2) \otimes \dots \otimes f(w_nx_n) = \frac{\prod_{i=1}^n f(w_i x_i)}{\prod_{i=1}^n f(w_i x_i) + \prod_{i=1}^n (1 - f(w_i x_i))} \quad (4)$$

From (1) and (4) it may be seen that

$$f\left(\sum_{i=1}^n w_i x_i\right) = f(w_1x_1) \otimes f(w_2x_2) \otimes \dots \otimes f(w_nx_n) \quad (5)$$

Operation \otimes is known as *symmetric sum* [23]. It has been shown [6] that this operator maps $(0,1) \times (0,1)$ into the open interval $(0,1)$, it is associative, commutative, strict monotone in $(0,1) \times (0,1)$ and continuous (except at the points $(0,1)$ and $(1,0)$). Furthermore it has been shown [23] that $((0,1), \otimes)$ is an Abelian group where the neutral element is $1/2$ and the inverse of an element x of $(0,1)$ is $1 - x$. It becomes apparent that inversion in the group corresponds to a strong negation (actually, to the most used negation) in the context of fuzzy logic. Finally, for all x, y in $[1/2, 1]$ the \otimes -operator behaves like a t -conorm and for all x, y in $(0, 1/2]$, the \otimes -operator behaves like a t -norm [15].

It becomes apparent that the closer to $1/2$ is the satisfaction degree of a premise, the weaker will be its influence on the aggregation of the premises with the \otimes -operator. In other words, the weaker the statement of a premise, the less it will affect the final conclusion.

In the case of the books considered in the opening example,

$$\begin{aligned} \mu_{\text{award}}(\text{1st book}) \otimes \mu_{\text{good-price}}(\text{1st book}) &= 0.98 \otimes 0.51 \\ &= \frac{0.98 \cdot 0.51}{0.98 \cdot 0.51 + 0.02 \cdot 0.49} = \frac{0.4998}{0.4998 + 0.0098} = \frac{0.4998}{0.5096} = 0.98076 \\ \mu_{\text{award}}(\text{2nd book}) \otimes \mu_{\text{good-price}}(\text{2nd book}) &= 0.6 \otimes 0.51 \\ &= \frac{0.6 \cdot 0.51}{0.6 \cdot 0.51 + 0.4 \cdot 0.49} = \frac{0.306}{0.306 + 0.196} = \frac{0.306}{0.502} = 0.6107 \end{aligned}$$

It may be seen that since $\mu_{\text{good-price}}$ was for both books very close to 0.5 the final recommendation follows strongly the award quality of the books.

Notice that (5) indicates that the output of a neuron (of the hidden layer) of a feedforward neural network with a sigmoide activation function, is equivalent to the symmetric sum of the partial results obtained by applying the sigmoide to every single weighted input. Furthermore the curve representing a sigmoide may be interpreted as a soft trapezoidal fuzzy set.

At the right hand side of (5), every w_i in $f(w_i x_i)$, $1 \leq i \leq n$, affects the slope of the corresponding sigmoide (at the inflection point).

Define $f^{(w_i)}(x_i) = f(w_i x_i)$. Then (5) may be written as:

$$h(x_1, \dots, x_n) = f\left(\sum_{i=1}^n w_i x_i\right) = \bigotimes_{i=1}^n f^{(w_i)} x_i \quad (6)$$

Equation (6) may be given the following interpretation:

*“if x_1 is in T_1 and ... and x_n is in T_n
then h is given by the symmetric sum of their degrees of satisfaction”*

where “is in T_i ” denotes “at least” if the corresponding weight w is positive or “at most” if w is negative, and T_j is the linguistic label associated to the fuzzy set represented by $f^{(wj)}$. A reference threshold e.g. 0.5 will be agreed upon to interpret the conditions “at least” or “at most”, respectively.

It becomes apparent that the extracted rule has a complexity-(understandability)-similar to that of the system introduced in Takagi and Sugeno [26], except that the conclusion is not a *linear*, but a *compensating* combination of the inputs based on the symmetric summation. If like in the majority of feedforward neural networks the hidden nodes are connected to a linear output node, from the fuzzy logic point of view, the networks represents a fuzzy additive system [16].

Figure 3 shows a neural network with one hidden layer with sigmoidal activation functions and a linear output node. The rule extraction at the right hand side. It becomes apparent that the network represents a *compensating* fuzzy additive system, i.e. The output rule is a linear combination of the rules generated at the hidden nodes.

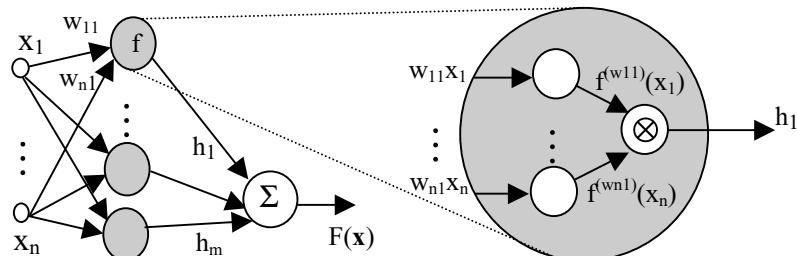


Fig. 3. (Left) The neural network view. (Right) The fuzzy logic interpretation

A feedforward neural network was reported in [17] to model the enthalpy of vaporization for three different classes of chemical compounds obtaining better results than with known established methods of chemical engineering. Training data was prepared for the study of three and five different functional groups and the corresponding neural networks had 4–4–1 and 6–5–1 structures, respectively. The neural networks presented a very high accuracy with an average mse of 0.000037. With the above method, the models may be expressed with 4 and 5 fuzzy if-then rules, containing 4 and 6 premises, respectively. An ANFIS-like system used to solve the same problem and using only *three* linguistic terms per variable would require $3^4 = 81$ and $3^6 = 729$ rules, respectively.

It should be emphasized that, as shown at the left hand side in Fig. 3, there are no restrictions on the feedforward neural network, other than using sigmoide activation functions at the hidden layer and a linear output. This means that a large number of *already existing* neural networks may be given a proper interpretation as compensating additive fuzzy systems.

3 Conclusions

It has been shown that there are systems that are not appropriately modeled with the help of t -norms, but with compensating aggregation operators. Furthermore a one to one relation between neural networks using a sigmoide activation function and rules combinig premises with a symmetric summation has been presented. This allows not only a convenient way of neuro-fuzzy modeling compensating systems, but also to extract (compensating) rules from existing neural networks. If the hidden layer has been minimized either by means of a pruning strategy or an evolutionary algorithm, this will lead to a correspondingly reduced number of reasonably understandable rules.

References

1. Bersini, H., Bontempi, G.: Now comes the time to defuzzify neuro-fuzzy models. *Fuzzy Sets and Systems* **90**, 161–169, 1997 [155](#)
2. Benítez J.M., Castro J.L., Requena I.: Are neural networks black boxes? *IEEE Trans. on Neural Networks* **8**, 1156–1163, 1997 [159](#)
3. Brahim K.: Neuro-Fuzzy Inferenz-Systeme. In: *Fuzzy Logic. Theorie und Praxis* (B. Reusch, Ed.), 176–186, Springer, Heidelberg, 1993 [154](#)
4. Calvo T., Kolesárová A., Komorníková M., Mesiar R.: *A Review of Aggregation Operators*. Handbook of AGOP'2001. University of Alcalá Press, Alcalá de Henares, Spain, 2001 [157](#)
5. Craven M.: Extracting comprehensible models from trained neural networks. PhD Thesis, University of Wisconsin, Madison Wisconsin, 1996 [153](#)
6. Dombi J.: Basic concepts for a theory of evaluation: The aggregative operator. *European Jr. Operation Research* **10**, 282–293, 1981 [160](#)
7. Dubois D., Prade H.: A review of fuzzy set aggregation connectives. *Information Sciences* **36**, 85–121, 1985 [157](#)
8. Glorenne P.Y., Barret C., Brunet M.: Application of Neuro-Fuzzy Networks to identification and control of nonlinear dynamic systems. *Proc. Int. Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU)*, 507–510, Palma de Mallorca, 1992 [153](#)
9. Han J., Moraga C.: Parametric Feedforward Network based Adaptive Fuzzy Modeling. *Proc. Int. Symp. Intelligent Industrial Automation and Soft Computing*, B-159–165, Reading UK, ICSC Academic Press, 1996 [153](#)
10. Horikawa S.I., Furuhashi T., Uchikawa Y.: A new type of Fuzzy Neural Network for Linguistic Fuzzy Modeling. *Proc. 2nd. Int. Conference on Fuzzy Logic and Neural Networks*. 1053–1056, Iizuka, Japan, 1992 [153](#)

11. Jang J.S.R.: ANFIS: Adaptive Network based Fuzzy Inference System. *IEEE Trans. on Systems, Man and Cybernetics* **23**, (3), 665–685, 1993 **153**
12. Jang J.S.R., Sun C.T.: Neuro-fuzzy Modeling and Control. *Proceedings IEEE* **83**, (3), 378–406, 1995 **153, 155**
13. Keller J.M., Tager R.R., Tahani H.: Neural Network implementation of fuzzy logic. *Fuzzy Sets and Systems* **45**, (1), 1–12, 1992 **153**
14. Klement P., Mesiar R., Pap E.: *Triangular Norms*. Kluwer Acad. Publishers, Dordrecht, 2000 **154**
15. Klement P., Mesiar R., Pap E.: On the relationship of associative compensatory operators to triangular norms and conorms, *Int'l Jr. of Uncertainty, Fuzziness and Knowledge-based Systems* **4** (2) 129–144, 1996 **160**
16. Kosko B.: Fuzzy systems as universal approximators. *IEEE Trans. Computers* **43**, (11), 1324–1333, 1994 **161**
17. Mandischer M., Geyer H., Ulbig, P.: Comparison of neural networks, evolutionary techniques and thermodynamic group contribution methods for the prediction of heats of vaporization, Research Report CI-70/99, SFB 531, Universität Dortmund, ISSN 1433–3325, 1999 **161**
18. Moraga C.: Neuro-evolutionary systems for learning parametric fuzzy connectives from examples of behaviour. *Proc. Workshop Minería de Datos y Aprendizaje de IBERAMIA 2002*, 85–92. University of Seville, Spain, 2002 **159**
19. Moraga C., Pradera A., Trillas E.: Evolutionary tuning of fuzzy if-then rules at the level of operations: A proposal. *Proc. II Congreso Español sobre Metaheurísticas, Algoritmos evolutivos y bioinspirados*, 530–537. Press Universidad de Oviedo, ISBN 84–607–65–26–1, 2003 **154**
20. Moraga C., Temme K.-H.: Functional equivalence between S-neural networks and fuzzy models. In: *Technologies for Constructing Intelligent Systems 2*. (B. Mouchon-Meunier, J. Gutiérrez, L. Magdalena, R.R. Yager, Eds.), 355–364. Physica Verlag, Heidelberg, 2002 **159**
21. Pal S.K., Mitra S.: Multilayer perceptron, fuzzy sets, and classification, *IEEE Trans. Neural Networks* **3** (5) 683–697, 1992 **155**
22. Rotshtein A.: Design and tuning of fuzzy rule-based systems for medical diagnosis. In: *Fuzzy and Neuro-Fuzzy Systems in Medicine* (H.-N. Teodorescu, A. Kandel, L.C. Jain, Eds.), 243–289. CRC Press, Boca Raton, FLA., 1999 **153**
23. Silvert W.: Symmetric summation: A class of operations on fuzzy sets. *IEEE Trans. on Systems, Man and Cybernetics* **9**, 659–667, 1979 **160**
24. Su M.-C., Chang H.-T.: Application of neural networks incorporated with real-valued genetic algorithms in knowledge acquisition, *Fuzzy Sets and Systems* **112** (1) 85–98, 2000 **155**
25. Takagi H., Hayashi I.: NN-driven fuzzy reasoning. *Int. Journal of Approximate Reasoning* **5**; (3), 191–212, 1991 **153**
26. Takagi H., Sugeno M.: Fuzzy identification of systems and its application to modeling and control. *IEEE Trans. on Systems, Man and Cybernetics* **15**, (1), 116–132, 1985 **154, 161**
27. Vergara V., Moraga C.: Optimization of fuzzy models by global numeric optimization. In: *Fuzzy Model Identification* (H. Hellendorn, D. Driankov, Eds.), 251–278. Springer, Berlin, 1997

28. Yi H.J., Oh K.W.: Neural Network based Fuzzy Production Rule Generation and its application to an Approximate Reasoning Approach. *Proceedings 2nd. Int. Conference on Fuzzy Logic and Neural Networks.* 333–336, Iizuka, Japan, 1992 [153](#)
29. Zimmermann H.-J., Zysno P.: Decisions and evaluations by hierarchical aggregation of information, *Fuzzy Sets and Systems* **10** 243–266, 1983 [157](#)

Hybrid Computational Intelligence Systems for Real World Applications

L. Hildebrand

Abstract. This chapter covers two topics. An introduction into the field of the main disciplines used in computational intelligence and detailed description of real-world applications which make use of computational intelligence. The introduction describes fuzzy logic and evolutionary algorithms in detail, whereas the artificial neural networks are only described briefly. For a more detailed introduction see the chapter of Moraga in this book. The application explained in the later part of this chapter are based on projects carried out by the author and funded by the European community (EC) or the German research society (DFG).

1 Computational Intelligence

Computational intelligence covers three main disciplines of computer science: fuzzy logic, artificial neural networks, and evolutionary algorithms. All three disciplines differ from the classical artificial intelligence in the way information and data is expected. Classical artificial intelligence uses mainly symbolic expressions and different types of logic, like first order logic, non-monotonic logic, or default logic. In contrast computational intelligence uses information and data in form of numbers directly. All disciplines of the computational intelligence are in need of high computing power. Although the roots of all computational intelligence disciplines are dating back many decades this demand for computing power explains the late entering of computational intelligence in real-world applications. The use of a single method of computational intelligence has resulted in well suited applications. More specialized abilities emerge from the combination of two or more methods of the computational intelligence, the so called hybrid methods. The next part of this chapter describes the main methods of computational intelligence.

1.1 Fuzzy Logic

In recent years fuzzy logic has become widely acknowledged – apart from its other applications – as an important and useful methodology in the design

of rule based systems. It allows the representation of imprecise or incomplete knowledge and offers various mechanisms for reasoning with fuzzy data. In comparison to “classical” rule based systems, only very few rules are needed to describe difficult problems [6, 7, 8, 25].

To characterize the semantics of the verbal statements representing the fuzzy knowledge and to make it usable for the calculus it is essential to introduce the concepts of fuzzy sets and linguistic variables. Both are due to Zadeh. Crisp sets bear a one-to-one correspondence to characteristic functions

$$\mu_M : U \rightarrow \{0, 1\} \quad (1)$$

with

$$\mu_M(x) = \begin{cases} 1 & \text{if } x \in M \\ 0 & \text{if } x \notin M \end{cases} \quad (2)$$

where μ denotes the universe of discourse. An extension of the image set of these functions from $\{0, 1\}$ to the closed unit interval $[0, 1]$ leads to the concept of fuzzy sets. The mappings $\mu : U \rightarrow [0, 1]$ correspond to fuzzy sets just as characteristic functions correspond to ordinary crisp sets. Such a mapping is called membership function and provides a membership value for each element of its definition set. With fuzzy sets various shades of grey are introduced between the classical cases of black and white. The meaning of linguistic terms – we restrict ourselves to adjectives in what follows – for the verbal characterization of the input and output variables is fixed by appropriately defined membership functions.

Although there exist no restrictions on the form of membership functions, in the following we always use piecewise linear function, like triangles or trapezoids. We suppose all occurring universes to be finite intervals of real numbers.

The knowledge base of a system consists of rules of the form

$$\text{if } X_1 \text{ is } A_1 \text{ and } \dots \text{ and } X_m \text{ is } A_m \text{ then } Y \text{ is } B \text{ with priority } p \quad (3)$$

where X_1, \dots, X_m and Y denote input variables and an output variable, respectively; the symbols A_1, \dots, A_m and B stand for linguistic terms which characterize the variables and which are interpreted using membership functions as mentioned above, and p denotes a numerical value in the interval $[0, 1]$ characterizing the relative importance of the rule.

The inference mechanism for the evaluation of single rules consists of two steps:

1. First find out the degree of fulfilledness of the premise; for simple premises ($m = 1$) just calculate the function value of the membership function in the case of numerical inputs. In the case of fuzzy inputs (namely a membership function) use the supremum of the minima of the input function and the membership function of the premise. For complex premises ($m > 1$) recursively calculate the degrees for the distinct parts of the premise and then use the min-operator to interpret the “and”.

2. Then evaluate the conclusion; for this aim multiply the membership function of the conclusion with the product of the priority value of the rule and the degree of fulfilledness calculated in the step 1.

This scheme is to be gone through for all rules. If a variable occurs on the right hand side of several rules, the different results are to be accumulated disjunctively by point wise summation and then normalization on [0, 1]. Using this algorithm, one application of the inference mechanism supplies membership functions for all output variables. Crisp results can be obtained by projecting the centre of gravity of the areas limited by the membership functions onto the corresponding definition set [3, 4, 5].

1.2 Artificial Neural Networks

Artificial neural networks are a model for the (human) brain and its learning behavior. The information processing core of an artificial neural network is a neuron, which can only add weighted information and response, if the sum is large than a given threshold value. Many aspects that are important for biological neurons and the interaction of neuron cells are not modeled in the simple artificial neuron. The following figure shows the simplification of the biological neuron towards a mathematical oriented model (see Fig. 1).

The fact that artificial neural networks can process complex input data and behave as if they were intelligent lies in the interconnections of a vast number of single artificial neurons. These neurons are organized in layers. One layer is the input layer and accepts the raw information. The following layers are information processing layers, the last one is the output layer, which gives the result of the networks for the input information.

Standard feed forward neural networks have been shown to be general function approximators. With back propagation they possess a very simple method to learn from examples. Hence they present themselves for the given problem of creating a controller based on a table of example values.

See the chapter written by Moraga in this book for more details and explanation on the various types of artificial neural networks.

1.3 Evolutionary Algorithms

Evolutionary algorithms form a class of probabilistic optimization techniques motivated by the observation of biological systems. Although these algorithms

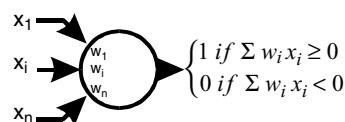


Fig. 1. Principle working of a single artificial neuron

are only crude simplifications of real biological processes, they have proved to be very robust and due to their simple parallelizability even efficiently implementable. The basic idea of evolutionary algorithms is the use of a finite population of individuals, any one of which represents exactly one point in search space. After its initialization the population develops in a self organizing collective learning process constituted by the (stochastic) subprocesses of selection, mutation and recombination towards better and better regions of the search space. The selection process favors those individuals with a high fitness value, the mutation operator supplies the population with new genetic information, and the recombination operator controls the mixing of this mutated information when it is passed from one generation to another.

According to T. Bäck [1], three types of evolutionary algorithms which differ in the procedures of initialization, selection, mutation, and recombination they use can be distinguished today: evolutionary programming (EP) was first developed by L.J. Fogel, A.J. Owens, and M.J. Walsh in the US [10], evolution strategies (ES) were invented in Germany by Rechenberg [18] and refined by Schwefel [20], and genetic algorithms (GA) are a product of the work of J. Holland during the sixties in the US [17]. Before presenting evolution strategies in detail, we'd like to characterize the notation of a parameter optimization problem first:

Given a function $f : M \subseteq R^n \rightarrow R$ a parameter optimization problem is the task of finding a parameter vector $x_{opt} \in M$ for which f has its global minimum. The definition set M of f can equal, but may also be restricted to a proper subset $M = \{x \in R^n | g_i(x) \geq 0 \forall i \in \{1, \dots, q\}\}$ by constraints of the form $g_i : R^n \rightarrow R$, with $i = \{1, \dots, q\}$.

Depending on the structure of the optimization problem – which means especially the properties of the function to be optimized – different optimization techniques are in use. evolutionary algorithms have shown to be valuable when other optimization fail as for example in the case of non-continual or non-differentiable problems and even when the problem changes in time.

We will present real-valued real-world applications in this work. From the choice of different evolutionary algorithms the evolution strategies are best suited for this kind of problems. The most commonly used variant of the evolution strategies is the so called (μ, λ) -evolution strategy. We will present the basic algorithm in the next section and show a powerful extension in the section after the next one.

(μ, λ) – Evolution Strategies

The (μ, λ) -evolution strategy is probably the most powerful and modern type of evolution strategies which has the great advantage of selfadaption of strategy parameters. (μ, λ) – evolution strategies are described by a octuple $(P^{(0)}, \mu, \lambda, rec, mut, sel, \tau, \tau', f, g, \kappa)$ whose components remain to be explained.

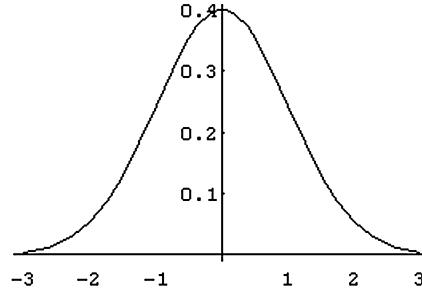


Fig. 2. Probability density function $N(0, \sigma)$

| | |
|---|-----------------------------------|
| $P^{(0)} = \{a_1^{(0)}, \dots, a_\mu^{(0)}\} \in I^\mu$ | initial population at $t = 0$ |
| $\mu \in N$ | population size |
| $\lambda \in N$ | number of ancestors |
| $rec : I^\mu \rightarrow I$ | recombination operator |
| $mut : I \rightarrow I$ | mutation operator |
| $sel : I^\lambda \rightarrow I$ | selection operator |
| $\tau, \tau' \in R$ | step size meta control parameters |
| $f : R^n \rightarrow R$ | objective function |
| $g_i : R^n \rightarrow R$ | constraints, |
| $\kappa : I^\mu \rightarrow \{0, 1\}$ | termination criterion |

For the following explanations let $N(0, \sigma)$ be the realization of Gaussian random variable with mean 0 and standard deviation $\sigma > 0$ as shown in Fig. 2.

Fitness and Population

An individual $a = (a_1, \dots, a_n, a_{n+1}, \dots, a_{2n}) \in I$ consists of an object component and a component which contains internal strategy parameters. The object component ($a_1 = x_1, \dots, a_n = x_n$) determines the position in the search space R^n , and the other component ($a_{n+1} = \sigma_1, \dots, a_{2n} = \sigma_n$) contains standard deviations which control the mutation process for the object variables. The fitness value of an individual is defined to be its value under the objective function. The best individuals are those, for which the objective function yields the smallest values.

Initialization

The start population $P^{(0)} = \{a_1^{(0)}, \dots, a_\mu^{(0)}\} \in I^\mu$ originates by mutation from a start point which may be chosen either at random or – as in our case – may be determined roughly by an approximate knowledge of the region in which the solution is situated.

Mutation

In evolution strategies the mutation operator *mut* is the main search operator and introduces something like innovation into the population. It first mutates the standard deviations of an individual a according to

$$\sigma'_i = \sigma_i \cdot e^{(N(0, \tau') + N_i(0, \tau))}, \quad \text{for } i = 1, \dots, n \quad (4)$$

and then modifies the object variable component using the new standard deviations according to

$$x'_i = x_i + N(0, \sigma'_i), \quad \text{for } i = 1, \dots, n \quad (5)$$

The modification of its own strategy parameters by the mutation operator is termed selfadaptation by Schwefel.

Recombination

In current literature a great number of recombination operators is proposed. Principally, *rec* is applied to the object component of an individual as well as to its strategy component; following empirical results it may be useful, though, to use different recombination procedures for the different components. For details see [1].

The operators may be distinguished regarding the selection of the parents – either two parents are chosen for all object variables and strategy parameters of a new individual, or two parents are selected anew for each quantity (this is called global recombination) – and the way of combination of parental information – discrete recombination which chooses one parent to pass the information to its offspring is in use as well as intermediate recombination in which an average value of the information of both of the parents is computed.

Selection

The selection operator *sel* works deterministically; it just chooses the best μ individuals from the complete offspring of size of the size λ .

Applying the algorithm to a population $P^{(t)} = \{a_1^{(t)}, \dots, a_\mu^{(t)}\} \in I^\mu$ yields a population $P^{(t+1)} = \{a_1^{(t+1)}, \dots, a_\mu^{(t+1)}\} \in I^\mu$ which has moved in search space towards a solution of the optimization problem. The iteration $P^{(t)} \rightarrow P^{(t+1)}$ stops when the termination criterion $\kappa(a_1^{(t)}, \dots, a_\mu^{(t)})$ is satisfied. The function κ is implementation dependent and may consider the CPU time already consumed, the number of generations, the absolute or relative progress per generation, or any other useful criterion.

1.4 Asymmetrical Evolution Strategies

Evolution strategies are a powerful variant of the evolutionary algorithms, which themselves are probabilistic optimization methods. Many sophisticated methods have been developed to increase the convergence of evolution strategies. Selfadaptation is one of these methods and allows an evolution strategy to adapt to the goal function. Nevertheless most real world applications of evolution strategies do not make use of the selfadaptation. In this section we analyze the reasons for this and introduce a new type of selfadaptation that overcomes the disadvantages of the known types [1, 12, 16].

Selfadaptation is an inherent feature of evolution strategies, that enables an adaptation of the optimization process towards a specific goal function. This means, that an evolution strategy can adjust itself between more global or more local search strategies. This adjustment is done by updating the step sizes according to the fitness of the individuals. The selfadaptation is a continuous process, so that an evolution strategy can escape from local optima by increasing the step size and focus on potential solution by decreasing it.

The first selfadaptation was introduced by Rechenberg and called the “1/5 success rule”. During an optimization process, a new vector $\vec{\sigma}'$ (which consists only of one step size) for the step size is computed using the deterministic rule:

$$\vec{\sigma}' = \sigma' = \begin{cases} \sigma/c, & \text{if } p > 1/5 \\ \sigma \cdot c, & \text{if } p < 1/5 \\ \sigma, & \text{if } p = 1/5 \end{cases} \quad (6)$$

The parameter p must be reevaluated for each generation, and gives the ratio of successful mutations to the total number of mutations. The parameter c is derived from the sphere model and constantly set to $c = 0.817$. The position vector x is mutated by applying the new step size to each component of x in the following step. At regular intervals, the survival behavior (measured as ratio p) of the parents and offspring are checked. If the fitness value of the offspring is too high ($p > 1/5$) then the search space is enlarged. It is assumed here that the reason for the large number of successful mutations is that we are far away from the optimum. If, however, a parent individual survives too few generations ($p < 1/5$), the step size is decreased in order to reduce the size of the search space. One disadvantage of this rule is that for optimization problems which do not achieve a reasonable success rate due to their topology, the step size will constantly decrease upon each application of the deterministic operation, so that the evolution strategy will eventually stagnate at a local optimum.

More sophisticated methods have been developed to overcome this disadvantage. They rely more on the vectors $\vec{\sigma}$ and $\vec{\alpha}$ of an individual $\vec{I} = (\vec{x}, \vec{\sigma}, \vec{\alpha})^T$. To benefit from the selfadaptation the inner working of the mutation operator $mut_\lambda()$ is important. An individual \vec{I} is changed by mutating each

of its components, first the strategy parameters $\vec{\sigma}$ and $\vec{\alpha}$, and then the object variable \vec{x} using the new values σ' and α' :

$$\begin{aligned}\vec{I}' &= (\vec{x}', \vec{\sigma}', \vec{\alpha}')^T \\ &= \text{mut}(\vec{x}; \vec{\sigma}', \vec{\alpha}') \\ &= \text{mut}(\vec{x}'; \text{mut}(\vec{\sigma}), \text{mut}(\vec{\alpha}))\end{aligned}\tag{7}$$

The idea is, that good strategy parameters result in better offspring, and the quality of the strategy parameters can indirectly be measured using the fitness of their offspring. Depending on the amount of strategy parameters used, three basic types of selfadaptation can be distinguished. The first type uses only the step size vector $\vec{\sigma}$ and all elements of $\vec{\sigma}$ are identical. That reduces $\vec{\sigma}$ to a scalar σ . The probability of the position of a new individual I' regarding the position of the parent individual I is shown in the next figure, where the position of I is in the center of the diagram. All shown diagrams are for 2-dimensional object vectors, to retain a descriptive figure.

Due to one single step size, the shape of the probability space is circular, with the highest probability at the center being the position of the parent individual. This type of selfadaptation is fast, only one parameter has to be adjusted. If the coordinates have different gradients this selfadaptation can only poorly adjust to it.

If each coordinate direction has its own step size a better adjustment can be achieved. The circular shape of the probability space changes into an elliptic shape oriented along the coordinate axes. Figure 3 shows an example.

Evolution strategies using this type of selfadaptation can adjust themselves in a more flexible way. Different gradients along the coordinate axes

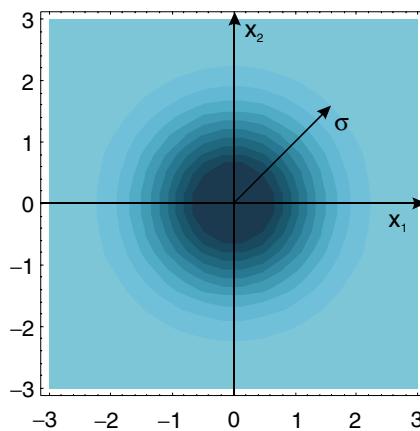


Fig. 3. Simple selfadaptation using one step size

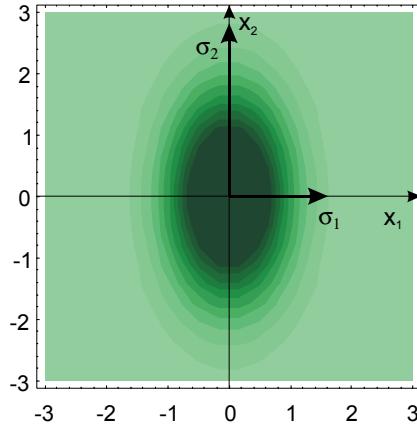


Fig. 4. Linear selfadaptation using n step sizes

can be taken into account. Disadvantage is the increasing number of strategy parameters that have to be adjusted. Evolution strategies using this type of selfadaptation are usually slower than the former one, but achieve a better result, because of their better adaptability.

The last and most complex type of selfadaptation is the correlated adaptation. This type uses the step size vector $\vec{\sigma}$ and the correlation vector $\vec{\alpha}$. $\vec{\alpha}$ corresponds to the correlation matrix, that is able to allow arbitrary rotation of the probability space with respect to the coordinate axes. The resulting probability space has an arbitrary oriented, elliptical shape. The next figure shows an example of this:

An evolution strategy using correlated selfadaptation has to adjust the step sizes $\vec{\sigma}$, as well as the correlation matrix $\vec{\alpha}$. The number of step sizes is linear increasing with the dimension of the optimization problem, whereas the number of rotation angles in $\vec{\alpha}$ is increasing quadric. Even for low dimensional optimization problems the number of rotation angles highly exceeds the number of step sizes or object variables. This results in slow convergence rates. This slow rate is the main reason for the lack of real world applications of correlated mutation, though the adaptability of this type is high.

Although the flexibility of the different type of selfadaptation allows a fine adjustment of the mutation operator to the goal function two main problems arise. The first one is the mentioned increase of the number of strategy parameters. The second one depends on the topology of the goal function and is caused by the symmetrical structure of the normal distribution. If the step size is sufficient small where it can be assumed that the goal function in this area is approximate linear, one half of the mutations are worse than the parent individual. This applies to all coordinate directions, so the ratio of better mutations to worse mutations is in the worst case $1/2^n$.

One idea to overcome this problem is to give up the demand for symmetrically distributed probability density functions. Introducing an asymmetrical distributed probability density functions the mutation operator can perform directed mutation by preferring a coordinate direction over another. The presented, directed mutation allows an independent directional adjustment for each coordinate directions, giving an enormous flexibility in adaptation. Before we explain the theoretical results, Fig. 5 is presented that shows the probability space of the directed mutation, similar to the previous figures.

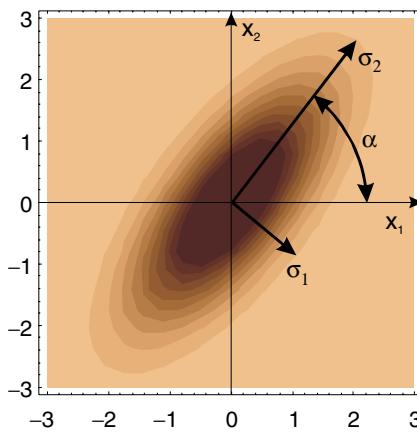


Fig. 5. Correlated selfadaptation using step sizes and orientation angles

The strategy parameters of the directed mutation consist of two parts, n step sizes σ_i for the n dimensions of the optimization problem and n directional values c_i for the preference of the coordinate direction. Using two independent vectors for the step size and the direction allows the directed mutation to behave like the simple or linear selfadaptation, if all directional values are set to 0. Figure 6 shows an example, where the step size of the x_2 axis is larger than the one of the x_1 axis, and the positive directions are preferred to the negative ones.

To develop a directed mutation operator a asymmetrical probability density function $\phi_{\sigma,c}(x)$ is needed. Each component of the object variable is mutated independently, so an 1-dimensional asymmetrical probability density function is needed. We will explain the development for an arbitrary step size σ and a positive directional value c , that prefers the positive direction. All following equations are valid for a negative c' (preference of the negative direction) if c is set to $-c'$ and the conditions are swapped. $\phi_{\sigma,c}(x)$ is based on an exponential function, that uses distinction of cases for the positive and the negative part of the coordinate axis.

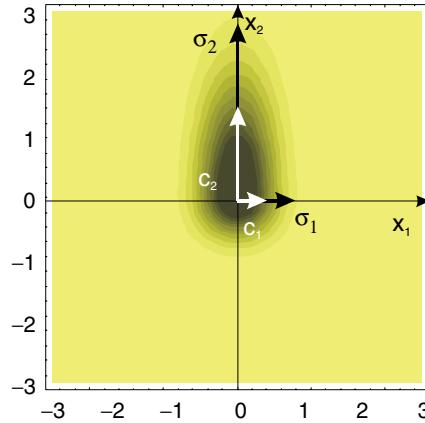


Fig. 6. Asymmetrical selfadaptation using step sizes and skewness

$$\phi_{\sigma,c}(x) = \begin{cases} \frac{\sqrt{2}}{\sqrt{\pi\sigma^\gamma} \left(1 + \sqrt{(1+c)^\gamma}\right)} e^{-\frac{1}{2} \frac{x^2}{\sigma^\gamma}} & \text{für } x < 0 \\ \frac{\sqrt{2}}{\sqrt{\pi\sigma} \left(1 + \sqrt{(1+c)^\gamma}\right)} e^{-\frac{1}{2} \frac{x^2}{(\sigma(1+c))^\gamma}} & \text{für } x \geq 0 \end{cases} \quad (8)$$

$$\Phi_{\sigma,c}(x) = \begin{cases} \frac{1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}\sigma^\gamma}\right)}{1 + \sqrt{(1+c)^\gamma}} & \text{für } x < 0 \\ \frac{1 + \sqrt{(1+c)^\gamma} \operatorname{erf}\left(\frac{x}{\sqrt{2}(\sigma(1+c))^\gamma}\right)}{1 + \sqrt{(1+c)^\gamma}} & \text{für } x \geq 0 \end{cases} \quad (9)$$

where $\operatorname{erf}()$ is the error function

$$\operatorname{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt \quad (10)$$

It can be shown that $\lim_{x \rightarrow \infty} (\Phi_{\sigma,c}(x)) = 1$, $\lim_{x \rightarrow -\infty} (\Phi_{\sigma,c}(x)) = 0$, and $\Phi_{\sigma,c}(x)$ monotone increasing is. The integral $\int_{-\infty}^{\infty} \phi_{\sigma,c}(x) dx$ is

$$\begin{aligned}
\int_{-\infty}^{\infty} \phi_{\sigma,c}(x) dx &= \frac{\sqrt{2}}{\sqrt{\pi\sigma^\gamma} \left(1 + \sqrt{(1+c)^\gamma}\right)} e^{-\frac{1}{2} \frac{x^2}{\sigma^\gamma}} \\
&\quad + \frac{\sqrt{2}}{\sqrt{\pi\sigma} \left(1 + \sqrt{(1+c)^\gamma}\right)} e^{-\frac{1}{2} \frac{x^2}{(\sigma(1+c))}} \\
&= \frac{1}{1 + \sqrt{1+c}} + \frac{\sqrt{1+c}}{1 + \sqrt{1+c}} \\
&= 1
\end{aligned} \tag{11}$$

so that all necessary properties for $\phi_{\sigma,c}(x)$ are fulfilled, to be a probability density function. Figure 7 shows the graph for $\phi_{\sigma,c}(x)$, with varying σ and c .

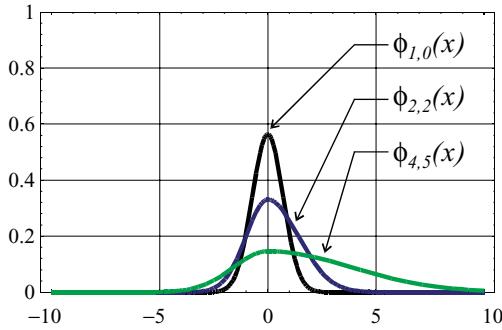


Fig. 7. Probability density of the asymmetrical density function

The selfadaptation works similar to the one of the linear adaptation. In a first step, the strategy parameters $\vec{\sigma}$ and \vec{c} are mutated. The second step mutates the object variable \vec{x} using this new strategy parameters. Again, the idea is that good strategy parameters are generated automatically during the optimization process and may vary over time to adapt to the goal function.

$$\begin{aligned}
\vec{I}' &= (\vec{x}', \vec{\sigma}', \vec{c}')^T \\
&= \text{mut}(\vec{x}; \vec{\sigma}', \vec{c}') \\
&= \text{mut}(\vec{x}'; \text{mut}(\vec{\sigma}), \text{mut}(\vec{c}))
\end{aligned} \tag{12}$$

Apart from the possibility to prefer coordinate directions, the directed mutation only needs $2n$ strategy parameters, where n is the dimension of the optimization problem. This is a great advantage over the correlated mutation, that uses $(n^2 - n)/2$ strategy parameters.

2 CI-based Hybrid Systems

2.1 Fuzzy Color Processing

Resistance spot welding belongs to a class of welding processes, that uses the electrical resistance of metals to join two conductive sheets of metal. Welding heat is generated by the flow of current that is passed through the sheets. The current is applied through electrodes, that also apply simultaneously a force. Advantages of the resistance spot welding process is its economy, speed and robustness. No secondary materials are needed, processing time is only a few seconds. Typical areas are automotive, aerospace, and engineering industry. Resistance spot welding can easily be automated, and in most cases only assistant helpers or robots are needed to supply the material. If automated systems are used the quality inspection process has to be automated, too. Quality testing systems can be divided into two major classes. destructive and non-destructive testing systems. If quality measures like longitudinal and transverse tensile strength, bend strength, or hardness have to be determined, the welding point has to be destroyed. Microscopic and macroscopic examination of the joint also require destructive operations. Non-destructive tests often need large and expensive equipment, like gamma or X-ray tubes, or are too sensitive to be used directly at the welding machine, like most ultrasonic sensors [13, 14, 15, 23].

The quality of a single welding spot is characterized by two criteria types: physical and optical criteria. Most of the physical criteria, like tensile or bend strength have to be determined using methods, that destroy the welding spot. These methods are necessary, because they result in exact assessments of the physical properties. These exact assessments are used by institutional quality testing, and are the base for legal liability. Physical criteria are not covered in this article. The focus of this article is on optical properties. These properties can be measured using non-destructive tests. Examples of these properties are color of the spot or the surrounding metal or the diameter or shape of the spot. Non destructive tests benefit from the fact that the working process in factories doesn't need to be interrupted.

To determine significant criteria of resistance welding points, a knowledge acquisition with welders and welding engineers has to be carried out. As a result, the following criteria can be deduced: size, color and shape of the inner spot, the outer spot and the impact zone.

The definition of these three regions of interest can be found in Fig. 8. The inner spot is the part of the welding joint, that is touched by the electrodes. The outer spot is characterized by a bulge which result from displaced steel. The impact zone is the outermost part of the joint, that is influenced by the welding process [13].

The next two figures show a welding spot with good quality (Fig. 9 left) as well as a welding spot with poor quality (Fig. 9 right).

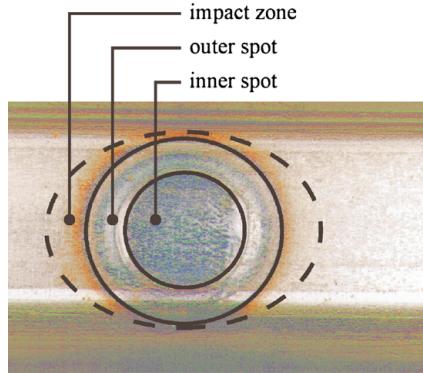


Fig. 8. Significant areas of a resistance welding spot

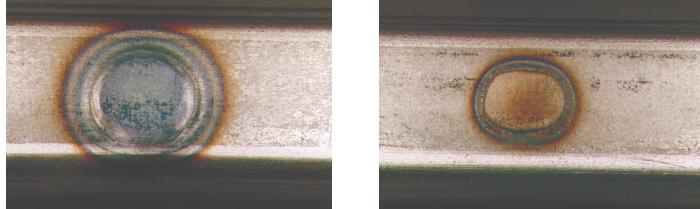


Fig. 9. Example of two welding spots with good (*left*) and poor (*right*) quality

An important optical criteria is the color. The color information has the advantage, that it is robust, because it is not susceptible to a change in electrodes or the material, that changes over time. Nonetheless, the wavelengths description of a certain color is not powerful enough to allow the processing of color information. In addition color is a subjective impression. It is an experience in human perception. If light of a specific wavelength falls on the different cones in the eye's retina this perception is created in the human brain. Three types of cones, that approximately correspond to red, green, and blue lights, and one additional type, that corresponds to the brightness are responsible for the perception [21, 22]. Due to the inadequacy of wavelength descriptions and the very subjective nature of color a more sophisticated description of colors have to be developed.

The mathematics belonging to the presented fuzzy color model is completely described in [13]. Here we will present the basic ideas and the application.

The fuzzy color model is based on the HSI color model. it uses the projection of the HS and HI color submodels on circular area based on a polar coordinate system. The added third dimension is the degree of membership. the resulting coordinate system is a cylindrical coordinate system. The next figure shows an example.

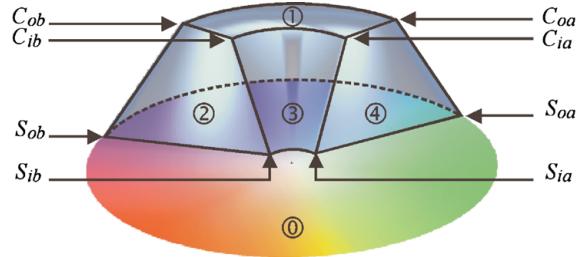


Fig. 10. Cylindrical coordinate system for fuzzy color processing

To increase the speed of computation some restrictions have to be maintained. A fuzzy set over the HS-color space is defined by eight HS-points S_{ia} , S_{ib} , S_{oa} , S_{ob} , C_{ia} , C_{ib} , C_{oa} and C_{ob} . This results in an extension of the polar coordinate system of the HS-color space towards a cylindrical coordinate system, see Fig. 10. Each point represents one corner of the fuzzy set, the letters S and C stand for the terms support and core, the letters i and o for the terms inner and outer, respectively. The eight points must fulfill the following conditions:

- equal angles of S_{oa} and S_{ia} , as well as equal angles of C_{oa} and C_{ia} ,
- equal angles of S_{ob} and S_{ib} , as well as equal angles of C_{ob} and C_{ib} ,
- equal radii of S_{ia} and S_{ib} , as well as equal radii of C_{ia} and C_{ib} , and
- equal radii of S_{oa} and S_{ob} , as well as equal radii of C_{oa} and C_{ob} .

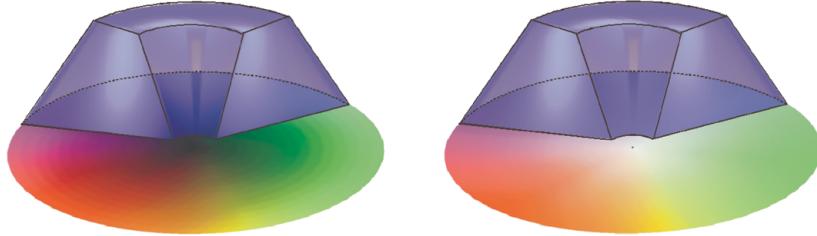
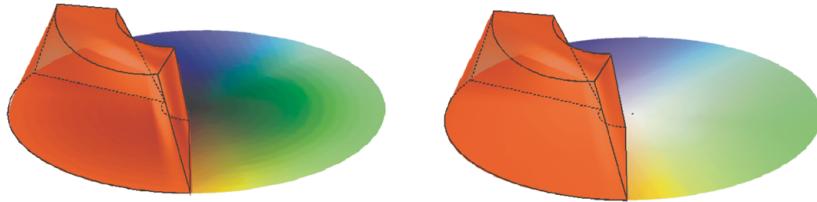
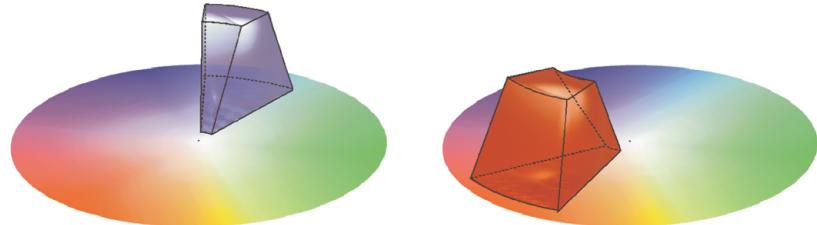
After all colors have been defined using the fuzzy color descriptions, these definitions can be used to add functionality to linguistic rule based systems. As an example, the quality criteria for resistance welding spots can be expressed in the following way [13]:

```

IF color_of_inner_spot IS blue THEN quality IS good
IF color_of_outer_spot IS light_blue THEN quality IS good
IF color_of_impact_zone IS dark_red THEN quality IS good
IF color_of_inner_spot IS light_red THEN quality IS poor
IF color_of_outer_spot IS blue THEN quality IS poor
IF color_of_outer_spot IS red THEN quality IS poor
IF color_of_impact_zone IS light_red THEN quality IS poor
  
```

A visualization of the used fuzzy sets can be found in Figs. 11, 12, and 13. The colors blue and red can be determined by using a fuzzy-or function with respect to the color definitions of the HS- and HI-color model. Due to the definition of the color model no special case is needed in rule (iv) to represent the linguistic term for the colors light grey – light red. By adjusting the channel of the representation of the linguistic term light red, the color light grey can be included (in this case), but also be excluded.

Colors found in the real world are not pure colors in the sense of color models. They are more like distributions of a multitude of different colors,

**Fig. 11.** Definition of color blue**Fig. 12.** Definition of color red**Fig. 13.** Definition of color light blue and light red

where the dominating ones give us an impression of a single, more or less noisy, color. This absence of unicolored objects can make the process of finding the desired fuzzy color definition very complicated. One approach to solve this problem is the following method, which is similar to the generation of fuzzy sets as described in [8].

In many cases, where colors need to be described as fuzzy colors, a sufficient number of sample pictures or photographs exist. These samples can be used to carry out an automated acquisition of the fuzzy color definitions. The idea is to determine the region of interest for a special color and to count the occurrences of all colors. Drawing this number as a third dimension alternatively on the HS- or HI-color space results in a frequency distribution of the existing colors. Some statistical method have to be applied to detect outlier and to fit an adequate shape of a fuzzy set. After this, the fuzzy set is automatically generated according to the samples, and the corresponding color definition can be used. If no regular shape of the fuzzy sets are needed, a normalized frequency distribution can be used directly. Figures 14 and 15 give

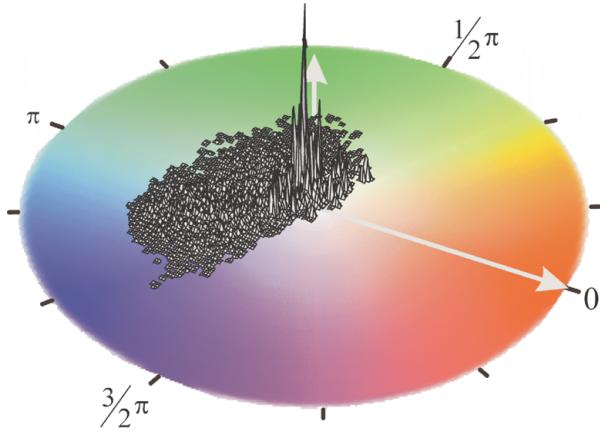


Fig. 14. Frequency distribution of a color (light blue)

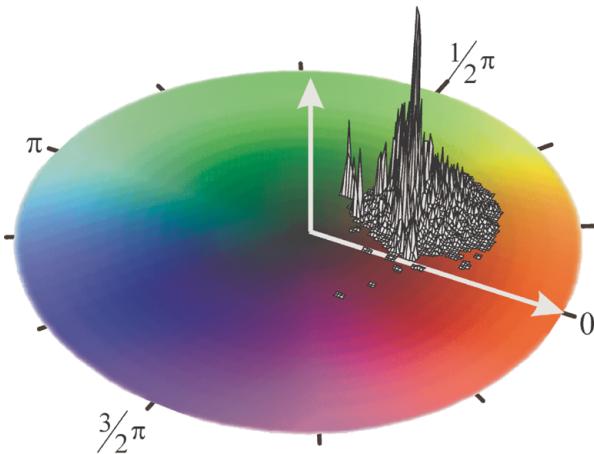


Fig. 15. Frequency distribution of a color (dark red)

examples of these frequency distribution. Figure 14 shows the HS spectrum (pure and pale colors) of the inner point of a good quality sample, Fig. 15 shows the HI spectrum (pure and dark colors) of the inner point of a sample with poor quality.

To generate proper fuzzy sets an evolution strategy can be used. The object parameters are the definition points of the fuzzy set. The whole object vector consists of eight two-dimensional coordinates, yielding a 16-figure vector. The dimension is average so different step sizes for each figure in the vector can be used. The fitness function calculates the difference of the frequency distribution and the surface of the optimized fuzzy set. The smaller the difference,

the better is the fitness value. As a result, fuzzy sets are created which are a good approximation of the sampled color frequency

2.2 Developing Specialized Digital Filters

A common task during image processing is the enhancement of certain image features. This task is performed if a direct analyzing of the image is not possible (arrow in Fig. 16). Well known are the techniques for edge detection, or the use of digital filters to enhance vertical or horizontal lines [11, 19]. These filters yield only poor results, however, if more complex features have to be detected. The following paragraph shows how digital filters in combination with evolution strategies can be used to generate image enhancement methods that are able to detect circular features.

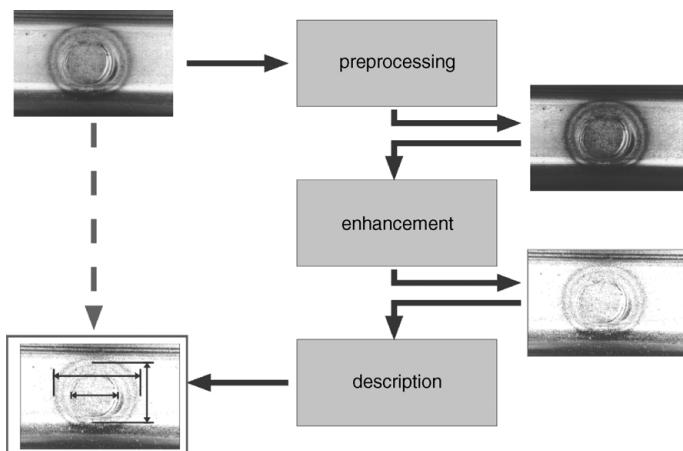


Fig. 16. Processing steps in a technical image processing application

Digital image filters use a matrix of filter coefficients that is applied to each pixel in the input picture. The pixel and its neighbors are used to calculate the value of the output pixel. The whole process is described by the coefficients of the filter matrix. Some matrix filters are well known, the Frei/Chen-, Sobel-, Prewitt-, or Kirsch filter [11, 19]. The result of these filters applied to a welding spot image is shown in the next figures, see Fig. 17, Fig. 18, Fig. 19, Fig. 20 and Fig. 21.

It can be seen that all four filters are not able to enhance the circular structure, that is necessary for the image description. Due to the fact, that the filter matrix can be expressed as a real valued vector, the use of evolution strategies to optimize the filter matrix is possible and allows a faster optimization compared to other types of evolutionary algorithms, e. g. genetic algorithms. The matrix can be rearranged to build a vector. The low dimensionality allows

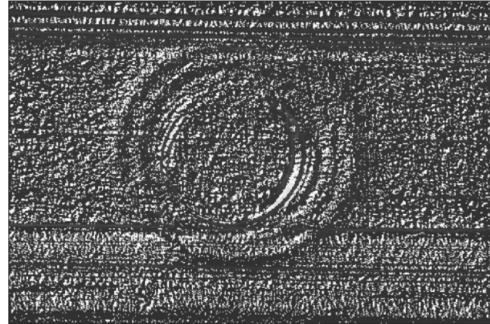


Fig. 17. Result of application of a Frei/Chen digital image filter

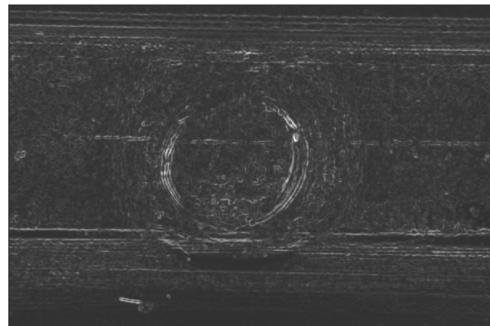


Fig. 18. Result of application of a Sobel digital image filter

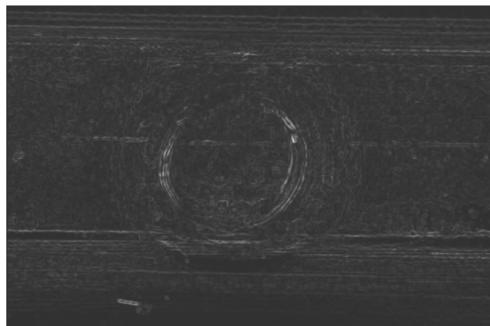


Fig. 19. Result of application of Prewitt digital image filter

the use of all types of self adaptation and evolution strategy variants. The fitness function compares the filtered image with a control image, in which the relevant features are enhanced by hand.

After the optimization a new filter matrix is created that enhances circular features as shown in Fig. 22. The values for the matrix elements after the

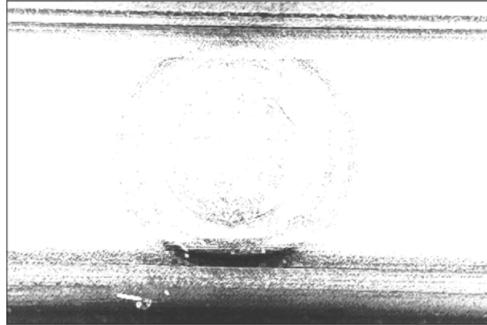


Fig. 20. Result of application of a Kirsch digital image filter

$$\begin{array}{ccc}
 \boxed{f_{11}} & \boxed{f_{12}} & \boxed{f_{13}} \\
 \boxed{f_{21}} & \boxed{f_{22}} & \boxed{f_{23}} \\
 \boxed{f_{31}} & \boxed{f_{32}} & \boxed{f_{33}}
 \end{array} \longrightarrow \begin{array}{l}
 (f_{11}, f_{12}, f_{13}, \\
 f_{21}, f_{22}, f_{23}, \\
 f_{31}, f_{32}, f_{33}) \\
 \hline
 (f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33})
 \end{array}$$

Fig. 21. Rearranging a matrix in vector form

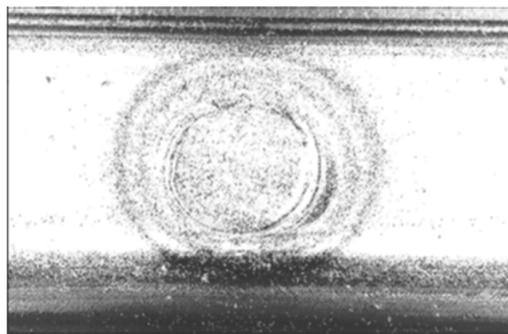


Fig. 22. Result of application of a evolutionary optimized digital image filter

optimization took place are: (7.77, -11.31, 0.13, -8.32, 10.77, 3.52, -6.11, 14.41, -5.35).

2.3 Analyzing GDOES Depth Profiles

The last decade there have been important developments in coating various steel substrates in order to enhance quality performance, comply with stringent environmental legislation and optimize costs.

Moreover, there is a trend to increase the use of pre-primed products, even complete coil-coating paint systems, for various industrial applications. This development will have important consequences for the steel sheet manufacturers.

In any stage of the coil coat production line, the process control is very important. Often an in-line control of a chromatisation layer is done by XRF analysis. The inspection control of the intermediate and final product is mainly based of visual aspects of the surface finish such as color, waviness, gloss, reflectivity and point defects together with its formability, adherence and hardness. At any stage of the production, some unforeseen problems can occur that affects the chemical or mechanical quality of the product without affecting the visual aspects (bad adhesion, corrosion). Often quality problems are encountered after a certain time of production. In order to re-trace possible errors, chemical characterization and quantification of each individual layer is necessary.

At present, the "near-the-line" measurement techniques for control of the coating lines are relatively limited. There exist different analytical probes or techniques, but none of them is really adapted to these tasks, considering the analytical improvement requests particularly with respect to the newly developed Cr-free conversion layers. In particular, fast quantitative measurement of thickness and composition of the different layers of more complex organic/inorganic coating systems has not been realized.

A few promising techniques for chemical analysis, like Glow Discharge Optical Emission Spectrometry (GD-OES), X-Ray Fluorescence Spectrometry (XRF), X-Ray Diffraction Spectrometry (XRD) and Laser Induced Breakdown Spectroscopy (LIBS), have been developed for analysis of industrial coatings. The industrial use of these techniques to date is mainly oriented towards quantitative measurement of inorganic coatings (like ZNi coating) and qualitative measurement of organic coatings. The main uses are in quality control or R&D laboratories. To a limited extent, a few "near-the-line" analytical systems are in industrial use for control of the quality of inorganic coating products. Primarily, these are based on different XRF measurement probes. Development of the other techniques in recent years has largely been focused on data treatment, primarily quantification. A very promising current development is in using computational intelligence, so-called expert systems, for evaluation of data and correlation to product quality criteria.

Since more than ten years, GD-OES has been extensively used in industry as a fast, powerful technique for quantitative determination of metallic coatings of up to 50 micron thickness (1). Recent work has also demonstrated that very thin layers down to a few nm can be profiled. Based on sputtering of the sample surface in an easy-to-use glow discharge source, it is a practical technique for industrial applications. With the introduction of Radio Frequency (RF) powered glow discharge sources, the applicability of GD-OES has been extended to non-conductive materials (2). For the steel and metal industry, the interest in this development consists in depth profile analysis of various

non-conducting surface layers and coatings, primarily organic coatings. In previous ECSC projects, the methodology to quantify depth profiles using RF GD-OES has been developed and successfully tested for a limited number of applications (3). One particularly important improvement over older quantification methods is the capability to correct for the strong influence of hydrogen on the emission intensities of other elements. This correction does not only correct for errors in measured concentrations, due to the nature of the effect even distortion in the shape of depth profiles can be partly removed. The evaluation of the industrial applications also revealed a number of additional artifacts of the technique, but several of these can undoubtedly be resolved, at least for several applications of industrial interest. In this context, it should be emphasized that GD-OES is one of the very few fast analytical techniques available for industrial coatings, of those it is undoubtedly the technique that gives the best resolved in-depth information [2].

In terms of hardware, the last few years has also seen the development of more compact GD-OES systems based on CCD-type spectrometers. Combined with the latest developments in quantification techniques and software, the prospects for development of GD-OES for “near- the-line” analysis is very promising.

Advanced Statistical Investigations

One of the typical problems in the analysis of GDOES-spectra is to find significant information independently of the spectrometers that was used. Some of the measured element signals are the same, no matter what spectrometers are used. Other signals can differ, but the differences can be significant or not. The following figure shows the C-signal of two different machines, see Fig. 23.

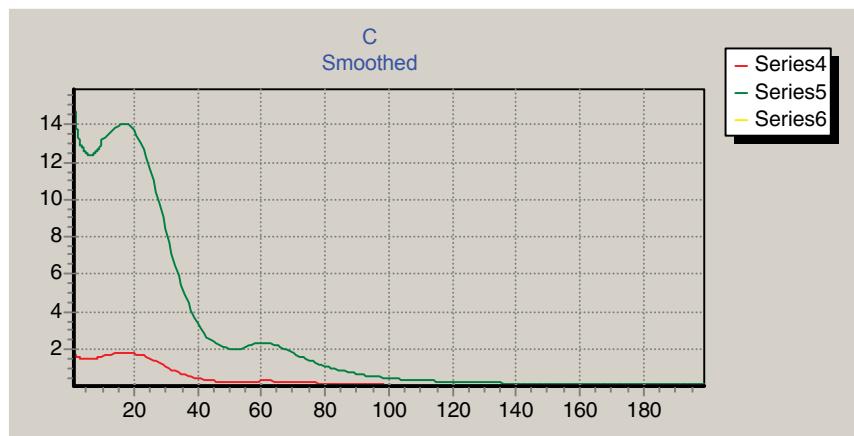


Fig. 23. C-signals of two different machines

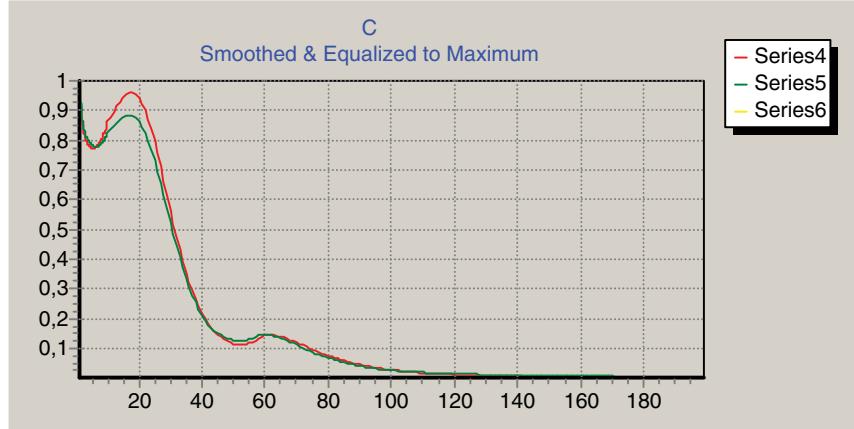


Fig. 24. Transformed C-signals of two different machines

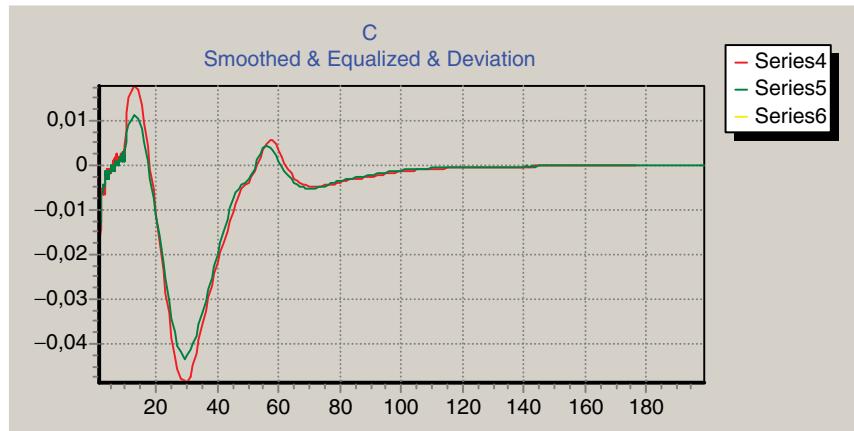


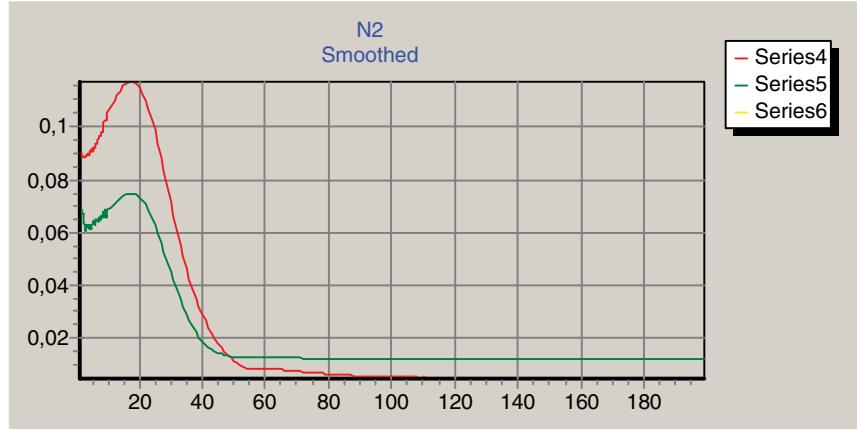
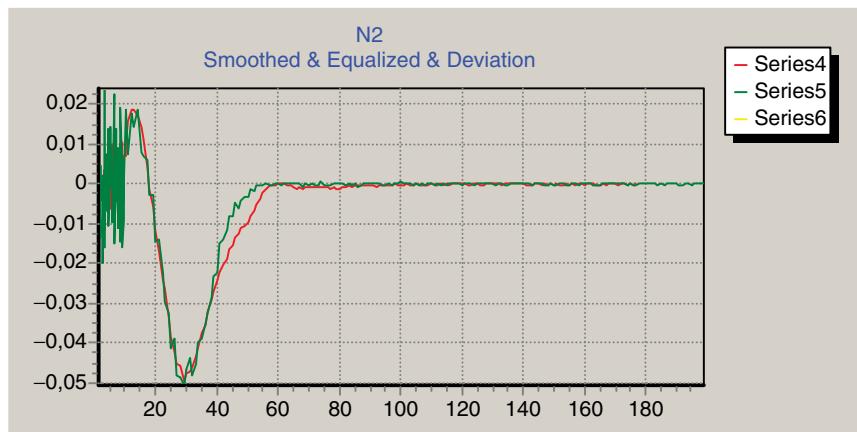
Fig. 25. Measure for the deviation of both C-signals

The signals seem to be significantly different but some transformations show that they are in fact nearly the same, see Fig. 24.

As measure for the degree of aberration the logarithm of the sum of squares of the first deviation can be used. The course of this measure is given in the following figure, see Fig. 25.

This method can not be influenced by different noise floors as shown in the next two figures where the N2-Signal is measured, see Fig. 26 and Fig. 27.

The proposed method is safe against different amplification of the used machines, methodical differences and different noise floors. All types of differences belong to the class of y -axis transformations. The method is not safe against transformations of the x -axis. Differences that belong to this class

**Fig. 26.** N2-signals of two different machines**Fig. 27.** Measure for the deviation of both N2-signals

are in most cases significant and need to be handled manually. The following figure shows an example for this, see Fig. 28.

The transformed data can be used to find trends in the bulk analysis. Trend deviation is due to warming of the sample and the machine or contamination of the optical system. If trends can be found, they are expressed in form of a trend line and the graphical representation of the single measurements against the mean of all measurements. The following figure shows an example for the H-signal, where a significant trend can be seen, see Fig. 29.

Figure 30 shows the Mo-signal, where no trend can be found.

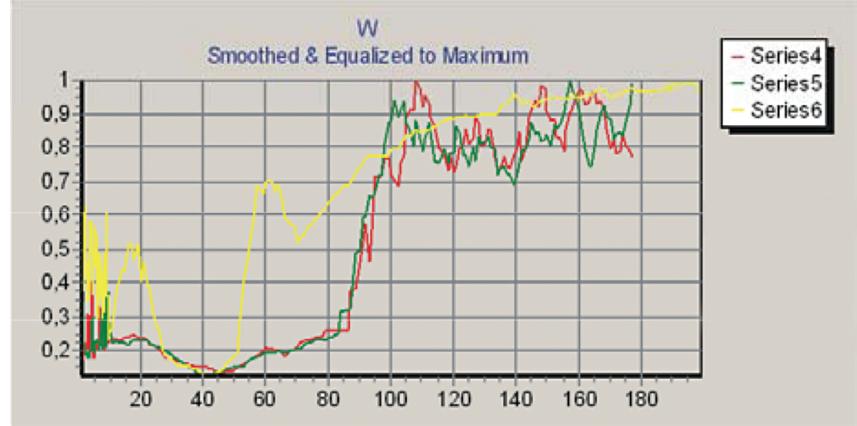


Fig. 28. Measure for the deviation of both N2-signals

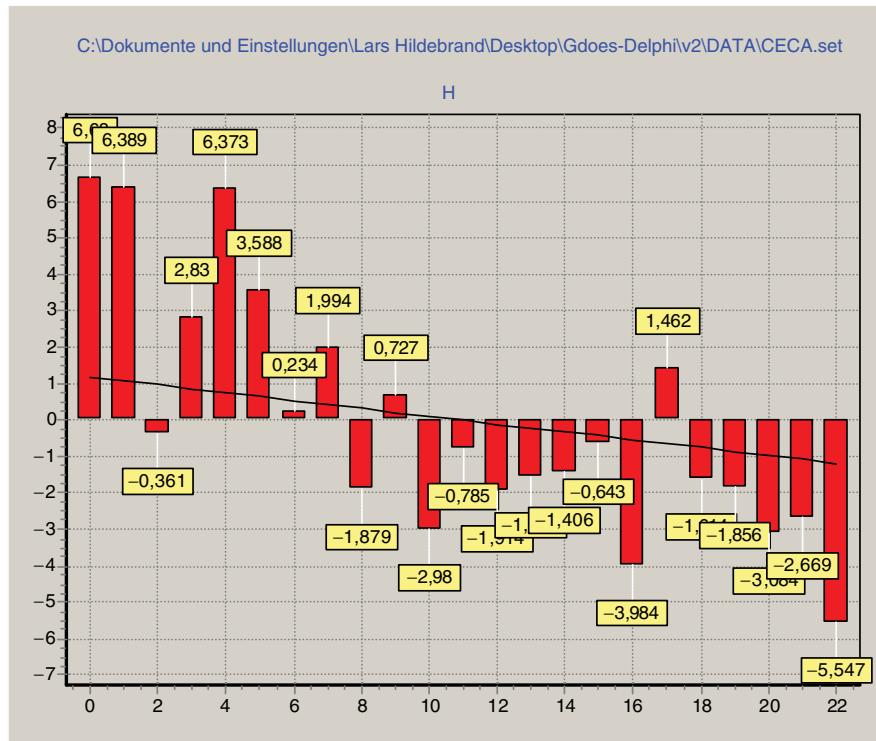


Fig. 29. Trend of the H-signal

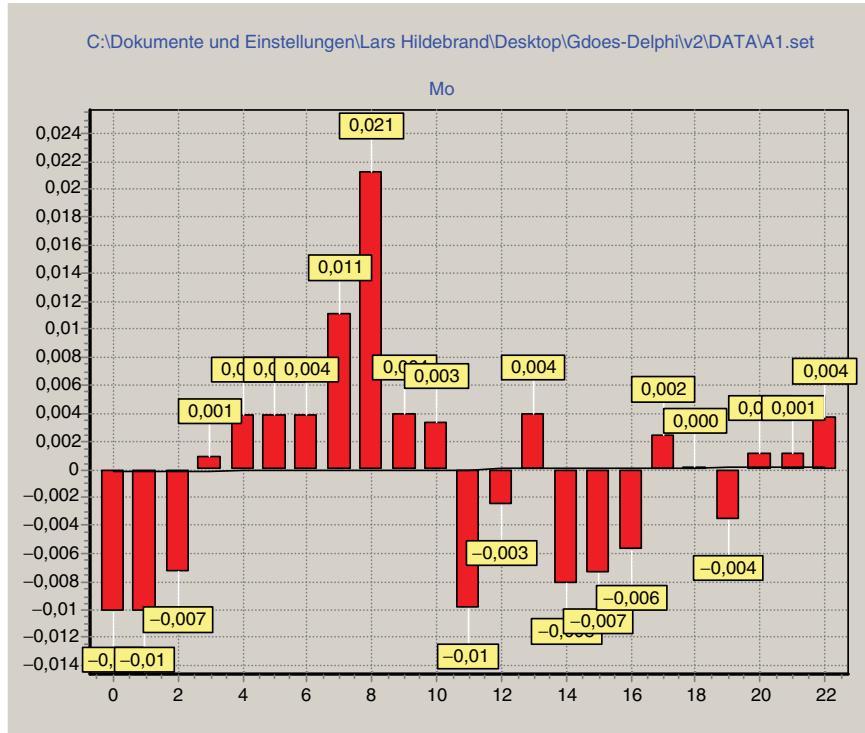


Fig. 30. No trend of the Mo-signal

Calculation of the Layer Thickness

The second main task of the last research period was the calculation of the layer thickness. Base of this calculation are key elements, tracer elements, and coating elements. All calculations are based on six different samples with varying coating thickness'. Figure 31 shows an example for the typical element signal, in this case for the Cr2-signal. The effect of the increasing coating layer can be clearly seen.

Figure 32 shows the position of the signal peak, based on the μm axis of the quantified data. All shown elements can be used as significant elements. The dependence of the peak and the coating thickness is nearly linear.

The position of the peak can be used to calculate the layer thickness. With the Cr2-signal as base an error of less than 6% can be achieved. The next table shows the result in detail, see Fig. 33.

Neural Network Based Method

To test if a self-organizing map can distinguish between good and poor samples we have created poor sample measurements artificially. Artificially samples

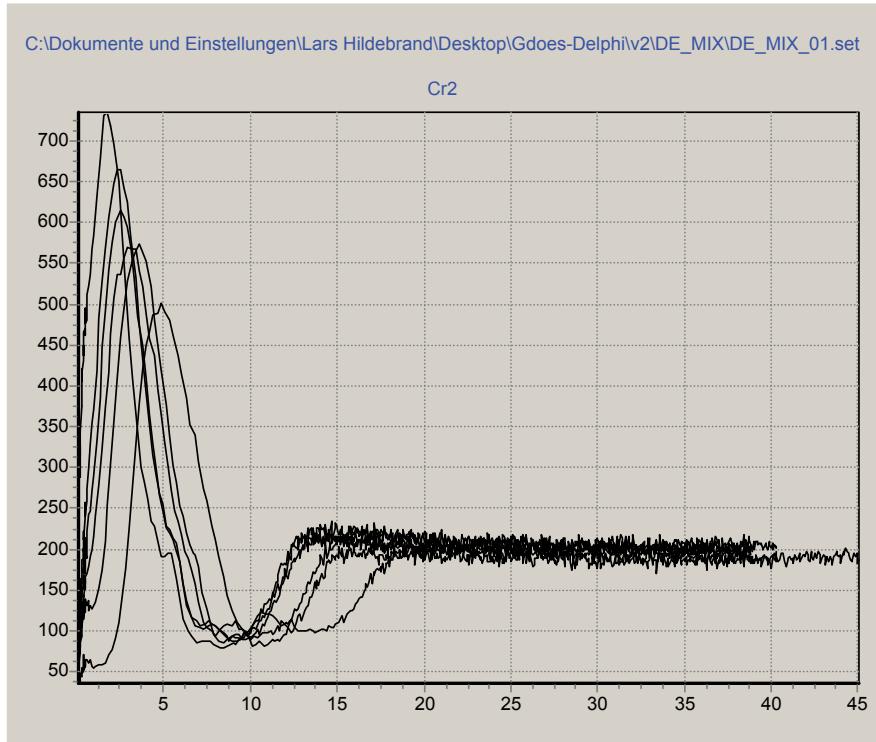
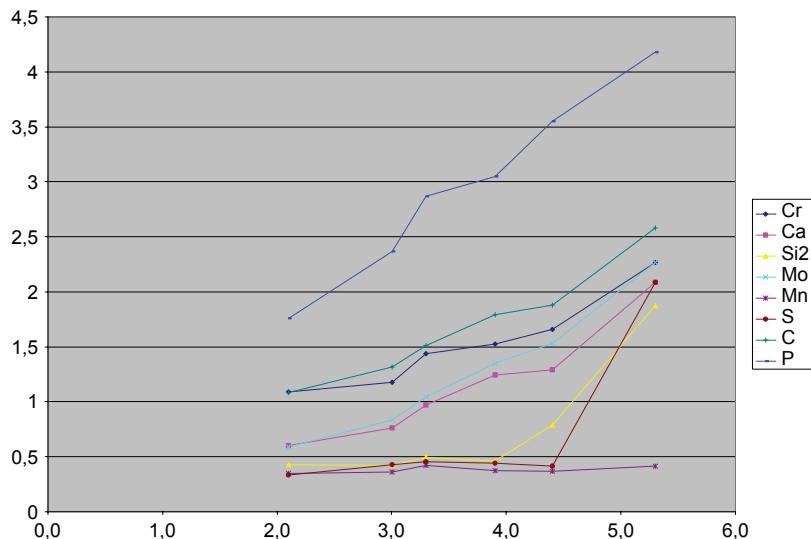


Fig. 31. Cr2-signal for different layer thickness

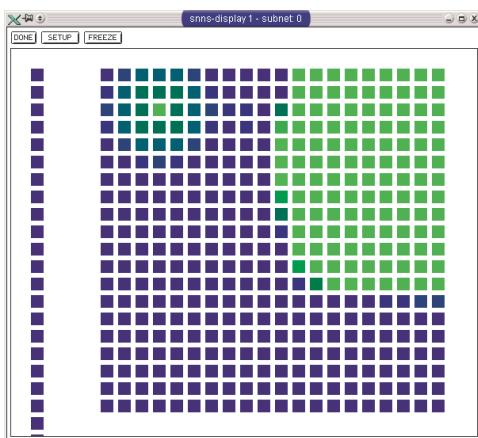
are needed, because the production process does not create poor samples currently. Figure 34 shows a typical example of the output of a self-organizing map. It can be clearly seen, that the poor example focuses at a different point compared with the good sample.

All element distributions, that are stable can be used as input for a self-organizing map. The placement of good and poor samples at different positions can be guaranteed for all stable element distributions.

The artificial neural networks that were used in the first period of the project have been programmed by the University of Stuttgart [24]. We used an existing system to minimize programming time and the danger of erroneous code and to maximize the force of expression and reliability of the self-organizing maps. To gain complete control over all features that exist in neural network systems we need a neural network system in source code. Such a system was programmed in the last period of the project. The system has the same functions as the one presented in the last report as well as the capability to extend it to more specific functions. This extension was not practical in the former system. The following figure shows the output of the trained elements as well as the centre of assessment.

**Fig. 32.** Position of the peaks

| real thickness [μm] | calculated thickness [μm] | error [%] |
|---------------------|---------------------------|-------------|
| 2.1 | 2.201987 | 4.85652381 |
| 3.0 | 2.8234894 | -5.88368667 |
| 3.3 | 3.317344 | 0.52557576 |
| 3.9 | 3.914366 | 0.36835897 |
| 4.4 | 4.1968898 | -4.61614091 |
| 5.3 | 5.6090092 | 5.83036226 |

Fig. 33. Detailed results of the thickness calculation**Fig. 34.** Output of a self-organizing map

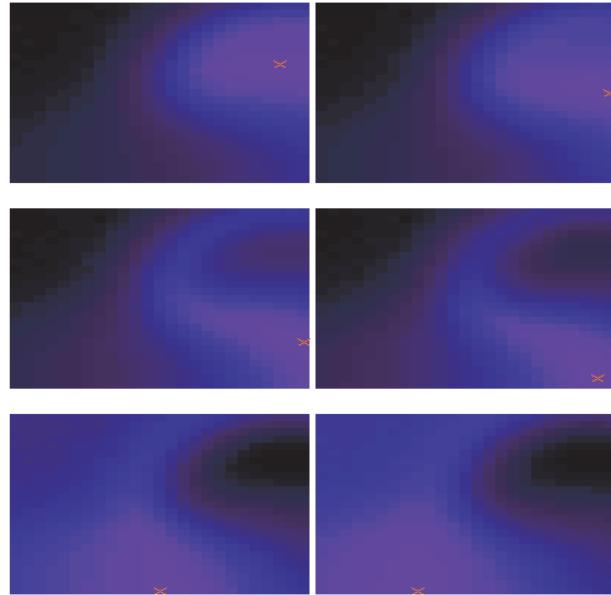


Fig. 35. Output of a self-organizing map

3 Conclusion

Computational intelligence offers a broad range of methods which can be used in real-world applications like the one described in this chapter. Each single method allows the solving of special problems. The combination of more than one method from computational intelligence in combination with methods from other disciplines like statistics offer the chance to solve even very difficult problems. Examples are the combination of fuzzy logic and evolutionary algorithms to allow a linguistic processing of color information in a rule-based way. The adaptation of the related fuzzy sets can be achieved by using frequency distributions and evolution strategies. Another example is the combination of digital image processing, digital filters and evolutionary algorithms. The last presented example show the benefits of using computational intelligence methods in combination with methods from statistics. The prediction of the layer thickness by using just the GDOES depth profiles is a challenging improvement of the cost and speed of industrial quality assessment.

References

1. Bäck, Th.; Hoffmeister F.; Extended Selection Mechanismus in Genetic Algorithms; in: Belew, R. K.; Booker, L. B. (edts.); 4th International Conference on Genetic Algorithms an their Applications, Morgan Kaufmann, San Mateo, 1991. (92–99) [168](#), [170](#), [171](#)

2. Bengtson, A., Payling, R., Jones, D. (eds.); *Glow Discharge Optical Emission Spectrometry*; John Wiley & Sons, 1997. **186**
3. Biewer, B.; *Fuzzy-Methoden: Praxisrelevante Rechenmodelle und Fuzzy-Programmiersprachen*; Springer, Berlin, 1997. **167**
4. Cox, E.; *The Fuzzy Systems Handbook*; Academic Press, San Diego, 1998. **167**
5. Dubois, D.; Prade, H.; Yager, R. R.; *Fuzzy Sets for Intelligent Systems*, Morgan Kaufmann, San Mateo, 1993. **167**
6. Fathi, M.; Hildebrand, L.; *The Application of Evolution Strategies to the Problem of Parameter Optimization in Fuzzy Rule Based Systems*; in: IEEE Computer Society (eds.); *IEEE International Conference on Evolutionary Computing – ICEC '95*, IEEE Press, Piscataway, 1995. (825–830) **166**
7. Fathi, M.; Hildebrand, L.; *Function Approximation using Fuzzy Logic and Evolutionstrategies*; in: Jamshidi, M.; Fathi, M.; Pierrot, F. (eds.); *Soft Computing with Industrial Applications*, World Automation Congress – WAC'96, TSI Press, Albuquerque, 1996. (197–202) **166**
8. Fathi, M.; Hildebrand, L.; *Modell Free Optimization of Fuzzy Rule Based Systems Using Evolution Strategies*; *IEEE International Journal on Transaction on Systems, Man and Cybernetics*, 1997. (270–277) **166, 180**
9. Fathi, M.; Hildebrand, L.; *Evolutionary Concepts for Image Processing Applications*; Zilouchian, A., Jamshidi, M. (eds.), *Intelligent Control Systems Using Soft Computing Methodologies*; CRC Press, Boca Raton, 2001. (381–408)
10. Fogel, L. J.; *Intelligence Through Simulated Evolution – Forty Years of Evolutionary Programming*; Series on Intelligent Systems, John Wiley & Sons, New York, 1999. **168**
11. Foley, J., VanDam, A., Feiner, S., *Computer Graphics: Principles and Practice*, 2nd Edition, Addison-Wesley, Reading, MA, 1990. **182**
12. Hildebrand, L.; Reusch, B.; Fathi, M.; *Directed Mutation – A New Selfadaptation for Evolution Strategies*; in: IEEE Computer Society (edt.); *Congress on Evolutionary Computation – CEC99*, IEEE Press, Piscataway, 1999. (1550–1557) **171**
13. Hildebrand, L.; Reusch, B.; *Fuzzy Color Processing*; in: Kerre, E.; Nachtegael, M. (eds.); *Fuzzy Techniques in Image Processing*, Studies in Fuziness and Soft Computing, Vol. 52, Physica-Verlag, Heidelberg, 2000. (267–286) **177, 178, 179**
14. Hildebrand, L.; Fathi, M.; *Linguistic Color Processing for Human-Like Vision Systems*; in: SPIE – The International Society for Optical Engineering (edts.); *Electronic Imaging 2000*, IS&T/SPIE 12th International Symposium, SPIE Press, Bellingham, 2000. (3959–32 S7) **177**
15. Hildebrand, L.; *Detection of 3-Dimensional Image Features Using a Single Camera*; in: Jamshidi, M.; Fathi, M.; Furuhashi, T. (edt.); *International Forum on Multimedia and Image Processing – IFMIP*, World Automation Congress – WAC2000, TSI Press, Albuquerque, 2000. (834–842) **177**
16. Hoffmeister, F.; Bäck, Th.; *Genetic Algorithms and Evolution Strategies: Similarities and Differences*; Technischer Bericht, Nr. SYS-1/92, Universität Dortmund, Fachbereich Informatik, LS11, Dortmund, 1992. **171**
17. Holland, J. H.; *Adaption in Natural and Artificial Systems*; University of Michigan Press, Ann Arbor, 1975. **168**
18. Rechenberg, I.; *Evolutionsstrategie '94*; Frommann-Holboog, Stuttgart, 1994. **168**
19. Rosenfeld, A., Kak, A. C., *Digital Picture Processing*, Vol. 1 and 2, 2nd Edition, Academic Press, San Diego, CA, 1982. **182**

20. Schwefel, H.-P.; Evolution and Optimum Seeking; John Wiley & Sons, New York, 1994. **168**
21. Silverstein, L. D.; Human factors for Color CRT Displays, Society for Information Displays, 1982. **178**
22. Teichner, W. H.; Color and Information Coding, Proceedings of the Society for Information Displays, Vol. 20, 1970. **178**
23. Waschkies, E.; Process-integrated resistance spot welding testing using ultrasonic techniques, Welding in the world 39, No. 6, 1997. **177**
24. Zell, A.; Simulation Neuronaler Netze; Addison Wesley, 1994. **191**
25. Zadeh, L. A.; Kacprzyk, J. (edts.); Fuzzy Logic for the Management of Uncertainty; John Wiley & Sons, New York, 1992. **166**

Autonomous Mobile Robots – From Science Fiction to Reality

N. Jesse

1 The Vision

For hundreds of years, people have dreamt of automatons that assist them in simple as well as difficult tasks. The term “automaton” dates back as far as the Iliad, where “automate” is used in connection with the self-opening doors of Mount Olympus. In the 18th century, mechanics and watchmakers began to build humanoid robots. The Swiss Jaquet Droz, for example, embarked on the difficult undertaking of developing a “writer” that was able to write without human assistance. The term “robot” was coined by the Czech novelist Capek and derives from the Czech robota (= labour) and robotnik (= workman). The idea of a highly intelligent automated being was picked up by other authors, the most prominent among them the Polish novelist Isaac Asimov, who predicted many contemporary concepts in his utopian writings. Asimov is also renowned for his three robot laws:

- A robot may not injure a human being, or, through inaction, allow a human being to come to harm. (= safety)
- A robot must obey the orders given it by human beings, except when such orders would conflict with the first law. (= shared control)
- A robot must protect its own existence as long as such protection does not conflict with the first or second laws. (= availability)

Science fiction movies like Star Wars, Terminator, I-Robot translate these visions into impressive pictures, influencing the public’s awareness of future possibilities.

The everyday usage of the term robot is very imprecise; however, a reasonable definition was provided by the Robotics Institute of America: A Robot is a reprogrammable, multi-functional manipulator (or device) designed to move material, parts, tools, or specialised devices through variable programmed motions for the performance of a variety of tasks. Devices that only perform predefined motion sequences, based on a rigid program, can be categorised

as robots no more than the remote control manipulators that are used in telemedicine, for example.

The expectations for future robots are great. Some anticipate that humanoid robots will be able to beat the ruling soccer world champions on the pitch. Three quotes by internationally acknowledged scientists highlight these expectations:

- “I believe that robots with an intelligence that is on a par with human intelligence will be a commonplace in 50 years!”¹
- “And once an intelligent robot exists, it is only a small step to a robot species – to an intelligent robot that can make evolved copies of itself.”²
- “Between humans and machines won’t exist any clear-cut distinctions.”³

Against the background of such expectations, Asimov’s three robot laws seem all the more relevant.

Irrespective of the different views held on these far-reaching prognoses, it can be noted that the technological advances of the last 50 years have been amazing, and the driving forces are unlikely to tire. This essay is based on a talk that has exemplified the state-of-the-art with a number of video clips. The aim of this essay is to underline with a few examples how far the development of autonomous, intelligent robots has progressed and what problems lie at the centre of research and development.

2 From Industry to Service Robots

2.1 Industry Robots: Fast and Precise

The first robot was installed by General Motors in 1961 to string and stack hot cast irons and many continued along this success. These early robot systems were inflexible numeric control machines, but since then, robots have developed into advanced sensor-control actuation-systems that can be programmed for more complex activities.

Today, conventional robots are a matter of course in many industries. Most of them are employed in a stationary capacity. The tasks they execute are clearly defined and performed in a static environment. These industrial robots can execute predefined movement sequences faster and with much more precision than humans, but they are not designed to work under varying conditions or in situations where flexible and co-operative actions are required (see Fig. 1). Some robots are able to move on predetermined routes, but their

¹Hans Moravec: Mind Children, Hamburg 1990, p. 16 (translated from German).

²Bill Joy: Why the future doesn’t need us, Wired, Online-Edition, p. 5.

³Ray Kurzweil: from an interview in: Frankfurter Allgemeine Zeitung, „Die Maschinen werden uns davon überzeugen, dass sie Menschen sind“ (The machines will convince us that they are human), 5.7.2000.



Fig. 1. Robots at work: (a) a car manufacturing line, (b) a palletizing robot with detention grip arm

Table 1. Robots at work for countries, estimated by the UNECE and IFR [UN02]

| Country | Operational Stock at Year-end | | | |
|---|-------------------------------|---------|----------|---------|
| | 2000 | 2001 | 2002 | 2005 |
| Japan (all types of Industrial robots) | 389,442 | 361,232 | 352,800 | 351,600 |
| United States | 89,880 | 97,268 | 1004,700 | 130,600 |
| European Union | 198,897 | 219,333 | 239,700 | 321,400 |
| ... | | | | |
| Other countries at | 8,900 | 10,840 | 13,000 | 21,000 |
| Subtotal, excl. Japan and Rep. Korea | 323,605 | 354,040 | 383,000 | 557,300 |
| Total, incl. all types of industrial robots. In Japan and Rep. of Korea | 751,034 | 756,539 | 780,600 | 964,500 |

movements are directed and by no means autonomous. A sensory perception of their environment or a reaction to external influences (with the exception of the classic safety circuits) is rarely necessary. Nevertheless, more and more robots are being equipped with sensors and used for more complex assembly and disassembly operations [9].

Currently, there are approximately 900,000 industrial robots in operation worldwide, (see Table 1) more than 240,000 of them within the European Community, but Japan has the highest density of robots, significantly exceeding the number within the European Community.⁴) Decreasing costs and the robots' constantly increasing performance are likely to lead to a further increase of these figures. As the computer technology and in particular embedded systems matured, the wide-spread use of robots expanded beyond the car manufacturing sector to areas such as assembly, die casting, plastic moulding, conveyor transfer, palletizing, inspection, etc.

⁴United Nations, Economic Commission for Europe, and International Federation of Robotics: World Robotics 2002, New York and Geneva 2002, p. 2. Interesting figures with respect to the German market are examined by [6].

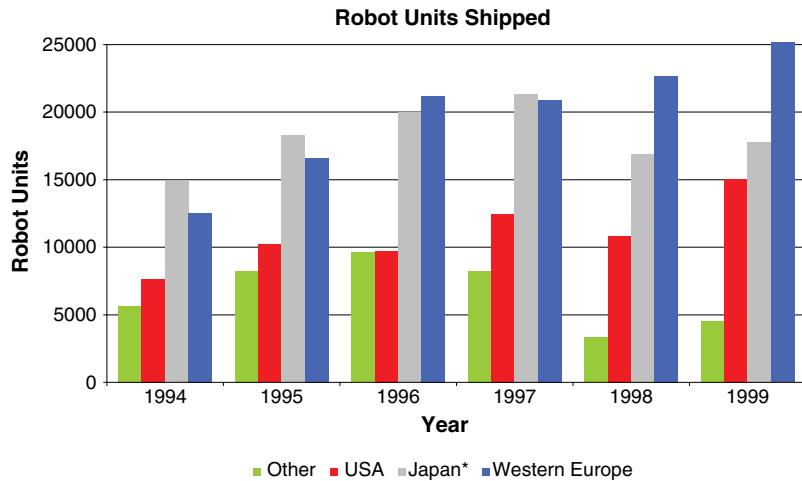


Fig. 2. Robot units shipped per year [IFR03]

Although the robot technology was invented in the US, the robotics industry is to a large extent dominated by Europe and Japan. Figure 2 illustrates that Western Europe is now the leading manufacturer, superseeding Japan.

2.2 From Industry to Service Robots

Robotics is one of the most innovative areas that have developed during the last decade, and one of the oldest dreams of scientific visionaries is starting to become a reality: the development of intelligent, mobile, and humanoid robots. Future robot generations are expected to operate independently and flexibly, i.e. to recognise and execute tasks autonomously, to perceive and assess objects and situations, and to react in a controlled way to unexpected or unfamiliar situations. Hardware technology has made considerable progress, for example in terms of chip performance and sensoric capabilities. Software has increasingly turned out to be the bottleneck: a robot's autonomy and competence to learn cannot be ensured by conventional software approaches, and significant progress is essential in this crucial field.

Table 2 outlines basic features that characterise industrial robots in contrast to completely new, challenging requirements that so-called *service robots* have to meet.

Experts predict that mobile, autonomous robots will influence our *lebenswelt* substantially in the near future. Their flexibility and their ability to react appropriately to different situations opens up a whole range of new applications. Current research and development focuses, among other sectors, on

Table 2. Industrial vs. service robots

| Industrial Robots | Service Robots |
|---|---|
| A high degree of precision | Adaptability to changes in environment (e.g. evasion of obstacles) |
| Repetitive tasks | Complex action sequences |
| Stationary or mobile along predefined coordinates | Free movement |
| “Closed world”, i.e. exact knowledge of the environment | “Open world”, i.e. unexpected objects and events can be handled |
| Monolithic, straightforward programs | “Variable” programs; computational intelligence (neuronal networks, fuzzy logic, evolutionary arithmetic) |
| No or very little co-operative ability | Co-operation with human and other robots |
| Separated from humans | Close proximity to humans |

- mobile manipulators in manufacturing environments;
- assembly and diagnosis robots for inaccessible or dangerous terrain;
- service robots for institutions like hospitals or for office environments;
- auxiliary robots for the disabled;
- transport robots for automatic distribution;
- supervision and surveillance robots;
- robots as part of home automation;
- robots for reconnaissance (e.g. on the Mars) and the extraction of raw materials (e.g. on the seabed).

The first generation of service robots on the market was designed for underwater, medical or demolition applications, to name but a few. Now service robots are about to enter our homes. According to an OECD survey, lawn-mowing robots have already made their commercial breakthrough, vacuum-cleaning robots were recently introduced to the market, and window-cleaning robots will be next. While 21,500 units were counted at the end of 2002, the forecasts for 2005 estimates 719,000 units, and the market for toy and entertainment robots is expected to exceed the one million mark (1,202,000: 2002–2005), most of them in the low-cost range.

The commercial success of service robots depends on their ability to offer a significant flexibility, quality and price benefit. Robots will only become a ubiquitous commodity if substantial progress is made.

2.3 Challenges in R&D: Autonomy and Co-operation

Robotics is a highly interdisciplinary sector, integrating contributions from the fields of mechanical and electrical engineering, computer science, neuroscience, and psychology, among others. The tendency of scientists to specialise



Fig. 3. The first envoys of service robots: lawnmover, vacuum cleaner and tennis ball collector

in their chosen field is reversed in robotics, where lateral thinking and the co-operation between many different disciplines is required.

Mobile robots combine physical (hardware) and computational (software) modules in a very special kind of way. From a hardware point of view, a robot is made up of movement, sensory and communication components. Due to the considerable technological development in recent years, we have efficient hardware featuring fast processors and multi-sensorial intelligence. In addition to the technological breakthrough, advanced intelligent software concepts are of vital importance if researchers are to tackle autonomy and co-operation concepts.

Service robots (see Fig. 3) in particular are designed for tasks that take place in an unstructured environment with the possibility of direct human-machine interaction. They are characterised by a certain degree of autonomy and, in many applications, the capability to co-operate with other robots or with humans. There are different degrees of autonomy. If a robot is equipped with an on-board controller and power supply (e.g. in case of automated guided vehicles), it is to some extent independent of outside control (weak autonomy). However, research focuses far beyond this narrow perspective, aiming at self-government capabilities (strong autonomy). In other words, research is working towards robots that

- are capable of making independent decisions, even if only vague or incomplete information is available (control autonomy);
- communicate with each other and with humans (communication competence);
- work within a team (ability to cooperate);
- plan behaviour over an extended period of time (planning competence);
- react quickly to unexpected changes in their environment (ability to adapt)
- learn from their experiences and adapt to long-term changes in their environment (ability to learn and to generalise).

To achieve these ambitious goals, the functionality of robots must be extended significantly and their performance improved in subsystems like environmental perception and modelling (3D vision, interpretation of blurred colours, analysis of pictures and scenes, or the classification of pictorial and

sensory data under real time conditions), intelligent navigation and collision prevention based on a multi-criteria decision-making process, task planning and interaction/communication.

2.4 Co-operative Robots: The Problem Space

State-of-the-art robotics technology concentrates on single-robot environments. In many ways, the technology used lacks the desired level of sophistication, but the academic world has a large body of theoretical and experimental knowledge at its disposal that can be applied to the sector of intelligently co-operating robots. Most of this knowledge, however, is “hidden away” in mono-disciplinary branches of science, conserved and expanded by specialists. The interdisciplinary nature of robot technology and the many problems as yet unsolved that materialise from the actual existence of artificial creatures provide the real circumstances to “match” theoretical abilities with practical implementation. A context like robot soccer competitions (see Chap. 3.3) confronts experts with complex dynamic requirements and demanding side-conditions.

Applications with multiple co-operating robots pose problems for which no clear solution has yet been defined or that have so far been solved only in theory. We face six different problem areas:

(a) *Control Theory*

The design of efficient control systems for single robots is a demanding task as motion is governed by highly non-linear equations. Moreover, physical constraints lead to limited controllability. The control of groups of robots creates an even bigger challenge, in particular when robots move in unstructured and sometimes unfriendly environments. The limitations of distributed onboard control will furthermore influence the type of solutions that are going to work, requiring research into new analytic solutions and behavioural approaches.

(b) *Perception*

Computer vision plays a decisive role in perceiving and understanding the world. The reconstruction of 3D scenes with moving objects or of scenes obtained through moving cameras from 2D images is a complex multi-levelled process. Low-level content-independent image processing techniques, such as filtering and feature extraction, rely on more or less well-established procedures. High-level content-dependent interpretation still poses the most difficult problems in the development of artificial human vision. The cognition process is generally linked to prior knowledge. Images are mapped into formalised world models, but these models must be constantly updated with new information. Therefore, computer vision has to integrate the results and methods of a range of disciplines, from mathematics to psychology.

(c) Planning & Navigation

Autonomous navigation is still a research area with numerous unsolved problems. Due to its complexity, multi-robot motion coordination is extremely computational-intensive, but the growing capacity of computers constantly extends the real-time “planning and learning capacity”. Whereas many contemporary multi-robot systems still employ simple reactive navigation methods, the “winning strategy” of the future will be to foresee coming events and predict other robots’ plans, making it possible to preplan or adjust motion behaviour more adequately. Look-ahead scheduling and real-time co-ordination create an enormous potential to improve the efficiency of computer-controlled robot collectives.

(d) Multi-agent Architecture

The concept of multiple interacting agents has received widespread attention in recent years. An agent – derived from the Latin “agere”, i.e. to do – perceives its environment through sensors and reacts to this environment through effectors. It acts on behalf of a person, but without human control. Attributes frequently regarded as constitutional are:

- Autonomy: acting independently, sensing changes and necessary actions,
- Goal-orientation: deciding how and where to fulfil user requests,
- Collaboration: interaction with user and agents,
- Flexibility: ability to choose between actions,
- Self-starting: sense changes, determining when to take which action,
- Temp. continuity: remaining active without user interaction,
- Character: a kind of identity,
- Communication: even across different platforms or systems,
- Adaptability: ability to learn,
- Mobility: move freely between different locations.

A large number of multi-agent architectures, approaches and interaction models has been proposed and is still being explored by researchers.

It is evident that centralised and hierarchical multi-agent systems (MAS) (see Fig. 4) are a difficult choice in obstructed or dynamic environments. Initiated by Rodney A. Brooks and Ronald C. Arkin in the 80th, there has been intensive discussion about the preference for deliberative or declarative MAS e.g. [2] (see Fig. 6). The basic characteristic for the deliberate approach is to decompose robot control systems into a series of functional units. These deliberative MAS depend on a model of the environment, and the control flow follows a hierarchical approach (see Fig. 5).

An appropriate world model and the ability to adapt this model, i.e. to learn to adapt to changing requirements, is the crucial part in this paradigm. Furthermore, if one of the modules fails, the whole system fails. The robot has no direct access to the sensor data but only refers to the world model.

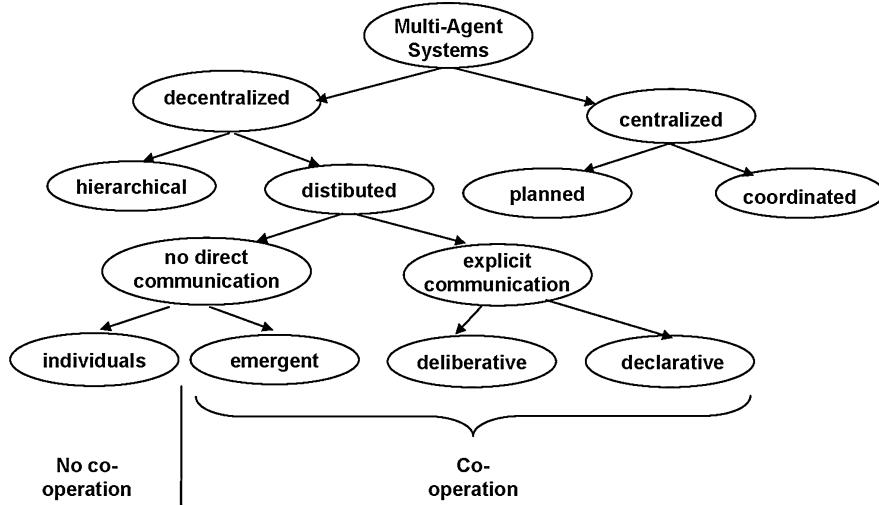


Fig. 4. Hierarchical decomposition of multi-agent architectures

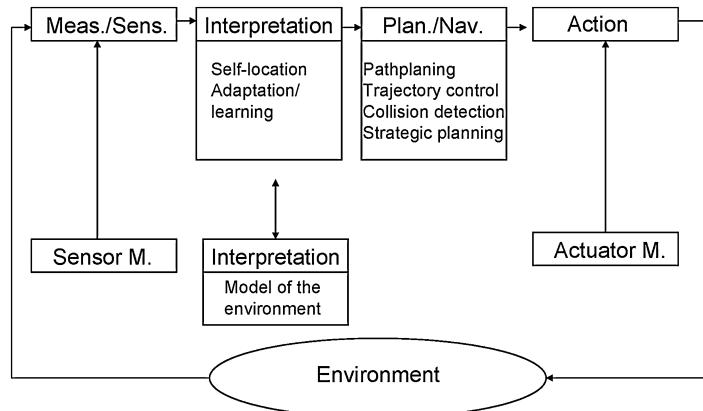


Fig. 5. The “measure-interpretate-plan-act”-cycle

On the other hand, declarative (reactive) MAS propose a radically different architecture for mobile robot control systems. This architecture operates without an explicit world model but implements different levels of competence. According to this paradigm, it is sufficient to sense the world appropriately and with sufficient frequency for the overall behaviour to emerge through concurrent operations. Coherent intelligence can emerge from modules interacting in the world. Architectures with purely reactive agents are simple and robust, but have the drawback of taking into consideration neither past experience nor future plans. The general tendency is to incorporate more cognitive

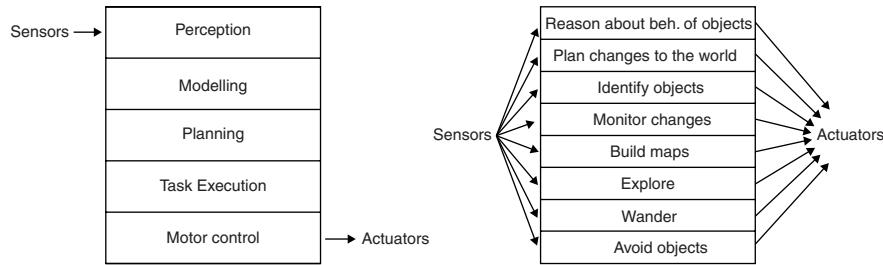


Fig. 6. Mobile robot control: (a) deliberate vs. (b) declarative (reactive) approach [11]

abilities to represent deliberative, long-term behaviour. A mix of both reactive and deliberative architecture would probably be the best combination. [1, 12]

(e) Artificial Intelligence

It is widely accepted that the traditional approaches to artificial intelligence, if based on symbolic reasoning, are inappropriate or not sufficient to produce intelligent behaviour in artificial creatures. One fascinating question remains unsolved: how do we create “intelligent” agents, e.g. robotic systems that react in an adequate and sound manner? The combination of theoretical and practical exploration is crucial if we want to determine along which lines we can best “generate intelligence”: by imitating biological entities, by designing models that can endow intelligence, or by creating systems that carefully perceive their surroundings and react intelligently?

Obviously, learning from nature is always a promising approach. During the last 10 years, a lot of research activity has been directed towards fuzzy logic, neural networks and evolutionary algorithms. These methods are expected to contribute significantly to robotics.

(f) Human-robot Interaction

Social skills and interaction modalities rely on physical presence, and embodiment is a precondition for interaction within a social context. The world of home and edutainment robotics in particular provides an appropriate environment to investigate different modes of verbal and non-verbal communication. In this context, research into non-verbal communication is very interesting: which signals are appropriate, what can we learn from human-human interaction, how should signals be synchronised or combined, how are they interpreted or evaluated by humans, and so on. It is hoped that current research into simple emotion, personality and dialogue models and other forms of social interaction will lead to useful and applicable results.

3 Humanoid Robots and Robot Soccer: Research Test-Beds and Education Tools

3.1 Humanoid Robots

Human beings prefer interaction with other human beings. The fascination for humanoid robots derives from this psychological fact. Obviously, there are further, simpler reasons that play a decisive role. Humanoid robots can make use of the same tools, objects and devices as humans (stairs, door knobs, screw-drivers etc.). Speech is the main carrier of information, but non-verbal signals (gestures, facial expressions, etc.) play an often underestimated role in human interaction. It is more authentic if humanoid robots express and recognise emotions, i.e. show some kind of emotional intelligence. Non-verbal signals are an integral part of the personality and are vital in passing on information and show emotional states effectively. The face is the most expressive part of the body and is able to communicate sadness, fear, anger or distress. Extending the robot's repertoire of non-verbal behaviour significantly improves the quality of the interaction. On the other hand, there are still a number of challenges in the verbal instruction of robots as there is no accepted general method for extracting the meaning of spoken communication [3].

Companies like Fujitsu, Honda and Sony have gained worldwide attention with their humanoid robots (see Fig. 7). Their research is motivated by two objectives:

- (a) to demonstrate human-like physiological abilities like stable, natural dynamic walking at reasonable speed⁵ and the ability to avoid obstacles,
- (b) to demonstrate human-like intelligent capabilities like sophisticated voice, face, and gesture recognition as well as a long-term memory for smart communication which also includes remembering the past.

Possibly the best-known humanoid robot is ASIMO. 120 cm tall and with a weight of 43 kg, ASIMO walks in rather fluid movements. It understands about 100 words, speaks in full sentences, has advanced face recognition capabilities and masters tasks like greeting and guiding people. QRIO is another powerful robot. 58 cm high and with a weight of 7 kg, he is able to execute 1,000 different movements (dance, balance, playing football etc.). QRIO has an extended speech recognition module (20,000 words, 200 pre-programmed conversational phrases) and an extended memory to make it capable of more in-depth communication with people.

Considerable resources were invested in the design and development of these robots. Now university researchers have developed low-cost robots to be used in education or as research test beds. For some years now, humanoid robots have taken part in robot soccer competitions, where they compete in

⁵Sony's QRIO for instance by 14 m/min (flat, smooth surface). For more examples see [14]

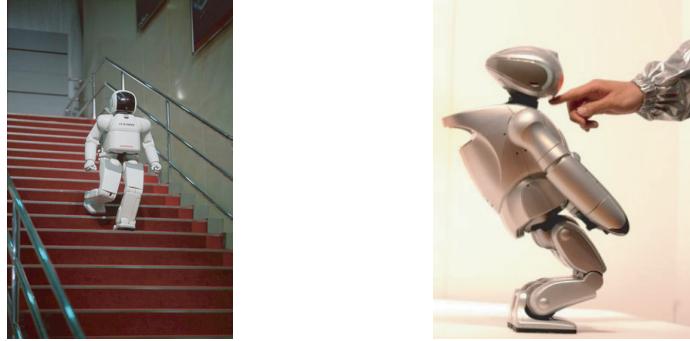


Fig. 7. Impressive humanoid robots: (a) Honda's ASIMO and (b) Sony's QIRO

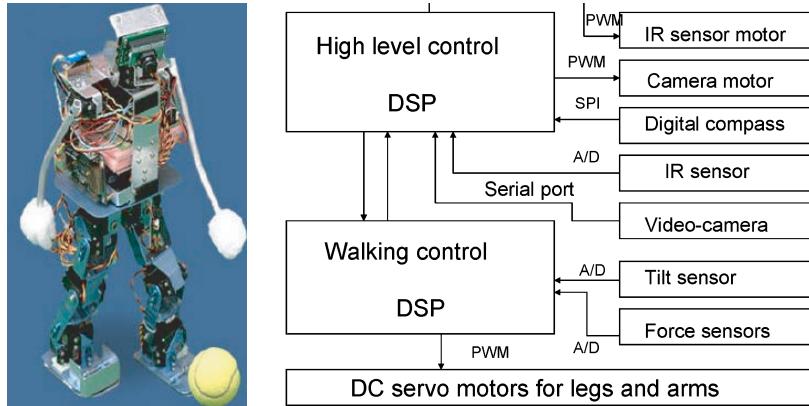


Fig. 8. Robosapiens-I from Singapore with control system

disciplines such as penalty kicks, backward runs, lift-and-carry competitions or balance keeping. These humanoid robots are still in their infancy, but the speed and stability of the walking gaits has significantly improved with every year.

One of these robots is the RoboSapiens-I, which complies with FIRA's humanoid competition rules. With 17 degrees of freedom, it has a vision system capable of tracking coloured targets and is able to stand up from a sitting position or to climb stairs. RoboSapiens is fully modularized and marketed as an open architecture (see Fig. 8).

One of the commercially most ambitious robots is NEC's PaPeRo (Partner-type Personal Robot). PaPeRo is an entertainment robot for the home, featuring a walking mode that makes it capable of wandering around rooms to look for a person to communicate with, and a talking mode for small talk, dancing or playing quizzes. PaPeRo follows and remembers faces, reacting with a happy expression and a preference for a particular person if stroked.

It is equipped with a sound direction detection function, speech recognition, cameras and LEDs for facial expressions.

3.2 Robotics for Entertainment and Education

Edutainment robotics is considered an emerging market. Many expect “that it is a growing new industry out of electronic dogs and cats, and toys that can serve both as a technology playground and, potentially, as a platform for consumer electronics”.⁶ Education and entertainment are being increasingly combined into a productive mix of playful studying and learning. Edutainment robotics addresses various target groups and goals, ranging from basic and higher education to leisure activities. It includes various age groups, from kindergarten and primary school to graduate university studies and life-long learning with re-creational systems like the Sony dog, Aibo, or robotic tour guides in science museums. It also covers various educational goals, ranging from highly specialized topics like control theory to the target of generating a general interest in science and technology.

One of the main reasons for the increasing interest in robotics edutainment is probably the wide availability of robot construction kits targeted at the mass market, like Lego Mindstorms or the earlier Fischertechnik Computing kits. Linked to the general availability of robotics construction kits, a growing popularity of large-scale robotics competitions has been observed in recent years. But the history of robotics in education started much earlier.

As early as 1980, Seymour Papert promoted in his book “Mindstorms” the idea that performance in learning activities may be significantly improved by using constructive methods, i.e. by involving the learner in an active design and building process. Papert’s ideas took concrete shape with the arrival of the programming language, LOGO, mainly targeted at the education of school kids in computer science. But LOGO and the concepts behind it were already used back in the mid-1980s for simulation-based robot labs. At the same time, the rise of new paradigms in artificial intelligence and in robotics, stressing bottom-up and simple biologically inspired designs, led to the development of low-cost control-boards and platforms, which were soon to be used not only in research but also in education. The MIT 6.270 Robot Design course is the most widely known example for this type of activity.

Successful examples of programmable mobile robots in primary education are PIP and PIXIE, two programmable robots for young children. These robots address the requirements of the UK National Curriculum for Schools and help children to learn numeracy and spatial awareness. From then on, robotics competitions evolved from small events as part of local hands-on courses over medium-sized sideline attractions like the AAAI robotics competitions into large-scale events in their own right, like the FIRA Championships and RoboCup. As these large-scale competitions promote scientific research as well

⁶Toshi Doi, Vice President Sony Corp.

as education, they further boost the popularity of robotics in education and even aim to play an active role in this field.

3.3 Fascination Robot-soccer: Trends and Challenges

Soccer is a game that stands for fitness, team spirit, individual flashes of genius, tactical discipline and creative flexibility, so it is no surprise that soccer-playing robots have become a fascinating education tool. However, their relevance extends far beyond this: robot soccer provides excellent research opportunities in a variety of fields, e.g. agent co-operation, robot control, sensor fusion, intelligent control, communication, image processing, mechatronics and artificial intelligence e.g. [4, 10]. Obviously, one of the basic ideas of robot soccer is to implement and test the co-operative group behaviour of mobile robots, but dynamic behaviour in a single robot remains a pre-condition. In the FIRA MiroSot league expectations for a single agent are for instance:

- the capability to stay on the given trajectory;
- an acceleration of more than 6 m/s^2 ;
- a speed of up to 3.5 m/s ;
- a motion tolerance of below $+/- 0.5 \text{ cm/m}$;
- a sound mechanical construction [7, 17].

Robot soccer tournaments have developed into highly dynamic and competitive games, and what may initially have appeared to be pure fun has established itself as an internationally accepted benchmark. The Federation of International Robot Soccer Association (FIRA) and RoboCup are two scientific communities that organise tournaments in many different leagues⁷, ranging from competitions for extremely small, centrally controlled robots in FIRA NaroSot ($4.5 \times 4.5 \times 5 \text{ cm}$) to humanoid, fully autonomous robots in humanoid leagues.

Introduction to the FIRA MiroSot-League

One of the robot soccer leagues most dedicated to entertainment, edutainment and research is the MiroSot FIRA league. It was designed for cube-shaped two-wheeled mobile robots with an edge length of up to 75 mm. It consists of a mechanical part with two wheels, two DC motors, a micro controller, a communication module and a power supply (see Fig. 10). These robots reach a maximum speed of 3.5 m/s . Each of the two wheels is connected with a gear to a DC motor. Each DC motor receives a command value – the desired speed – as input from the microcontroller-generated PWM (pulse-width modulation) signal. The robot’s behaviour depends on the accuracy and dynamics of the vision system as well as on the robot itself. Each robot receives reference velocity and reference angle velocity commands from the communication module. New values are transmitted to the robots at intervals of 33 ms.

⁷See www.fira.net and www.robocup.org.



Fig. 9. MiroSot game 5 vs 5 viewed from camera above the field

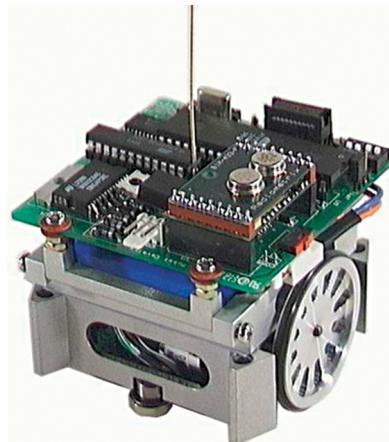


Fig. 10. The chassis of a robot with main board

MiroSot started in 1996 with 3 vs. 3 robots playing on a 150×130 cm-sized field. To imitate real soccer conditions, the number of players on the field slowly grew to 5 vs. 5, (see Fig. 9) later 7 vs. 7 (along with the field) and has now reached 11 vs. 11 robots on a 440×220 cm rectangular wooden field. During the early days of the league, the robots' mechanical designs, their control and the complex strategic demands were limiting factors. Since then, it has developed into a dynamic high-speed game with robots reaching speeds of more than 3 m/s. Based on refined strategies, the game proves a very entertaining experience for the spectators. Due to the very small size of the robots, they are supported by a host computer (usually an off-the-shelf PC), which receives images from the field via a camera that is mounted about 2.5 m above the field. The host processes the images, makes strategic decisions and transmits movement information via a radio link to the robots on the field, which they execute, thereby closing the control cycle.

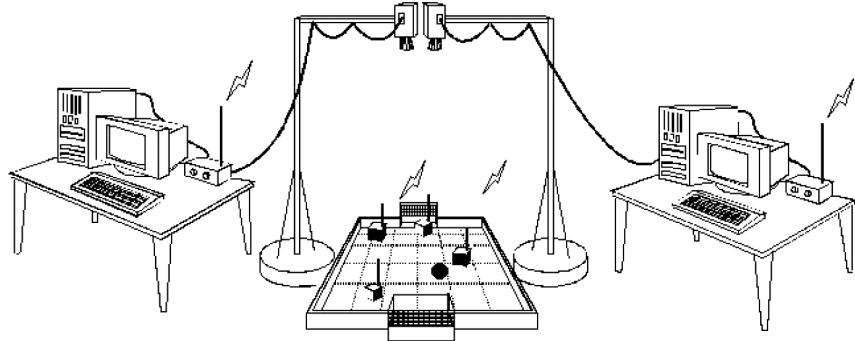


Fig. 11. The MiroSot-framework: robots, camera, host and communication

Only wireless communication is allowed. The radio module transmits in the 418, 433, 869 or 914 MHz ISM band range with specified gross data rates of up to 160 kbps. For various reasons, the net data rate is limited to around 35 kbps in this application; as the modules are half-duplex only with 22 robots on the field playing robot soccer, the limitations of the current modules have now been reached.

Originally, a FIRA MiroSot game consisted of only 6 robots (3 vs. 3). This design concentrated on control and vision problems, and the strategy was kept deliberately simple. One goalkeeper and two robots playing the game did not allow for any refined strategic moves. This aspect has changed during recent years, strategy design becoming much more complex in FIRA MiroSot due to the recent expansion to 11 vs. 11 robots on the field (see Fig. 11).

While some problems posed by robot soccer have been solved in the last few years, many still exist. Research focuses on software components capable of manifesting complex tactics, next to simple game technology, and the possibility of real team behaviour. The problems centre on the following areas:

- collision-free navigation;
- faster and more reliable mobile communication;
- adaptive pictorial assessment to compensate for light fluctuation;
- anticipation of the opponent's possible changes of direction;
- assessment of imprecisely described positions and situations;
- recognition of the opponent's tactics;
- intelligent robot navigation.

This paper emphasises two important aspects of robot soccer:

- (a) strategy design and
- (b) image processing.

We refer here to the experience and the implementation of the team of the University of Dortmund.⁸

A 3-Layer Strategy Module

The architecture of the Dortmund team's strategy module (see Fig. 12) consists of three layers: the tactics, role and action layer. The autonomy of the robots is confined by a coach responsible for the overall evaluation and the dynamic role assignment.

(a) The Tactics Layer

There are currently seven different types of tactics and nine standard situations (e.g. attack and kick-off). Based on the actual game situation, each tactics module calculates its own success probability. Relevant variables for this evaluation are, for example, ball position or the direction and velocity of the own team as well as of the opposing robots. The coach module (maximiser) (Fig. 12) decides the tactics with the highest success probability.

(b) The Role Module

The role module selects a specific role for every individual robot. These roles (e.g. active_defense or attack_with_ball) are chosen from a unique role set characteristic of the tactics. The role module re-analyses the actual situation focusing on important criteria like “maximum driving time to reach the ball” or “position of robots and ball”.

(c) The Action Modul

Each role has a choice of up to 14 different actions (e.g. block_the_opponent or goal_kick). Depending on the evaluation, the most promising course of action is selected. Selection criteria include “ball_at_the_border?” or “path_to_the_goal_unguarded?”.

Finally, these actions are translated into commands for the robot's control module, where the precise target coordinates for the robot and the wheel commands are calculated.

This process is illustrated by an example based on a team of three players:

- (a) the decision for the best tactics, i.e. “defence”, depends on the following criteria:
 - 1. ball position and distance between ball and own goal;
 - 2. opponent's activity, distance between opponent and ball, as well as angle and velocity of the opponent robots.

⁸The Dortmund Droids started with robot soccer in 1999. Most of the work has been done by students as part of one-year project groups or in diploma-theses. For more information with respect to the Droids visit www.robotsoccer.de [5].

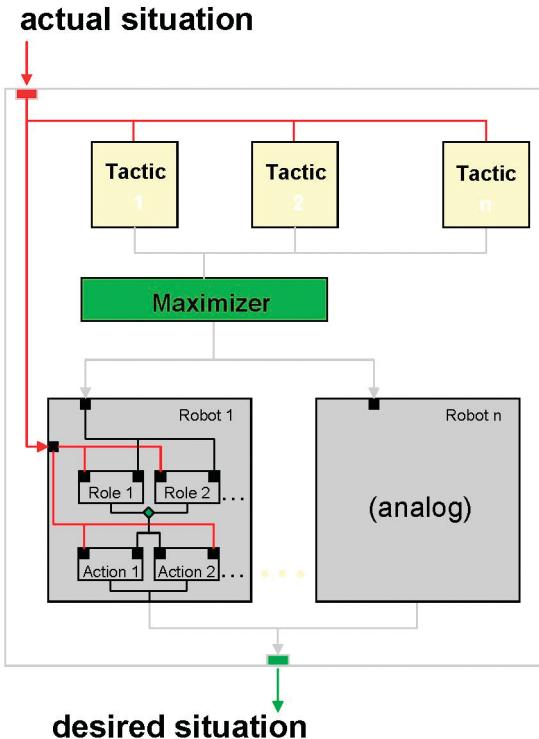


Fig. 12. The strategy module

3. ball “activity”, ball angle and velocity.

Three roles with a number of assigned sub-goals and actions are available within this defence scenario:

- | | |
|------------------|--|
| goalkeeper: | keep ball off the goal |
| active defence: | perform <i>Spinkick</i> if ball does not move and ball position is close to the border perform <i>BlockTheBall</i> if there is a chance to kick ball before opponent reaches it perform <i>BlockTheOpponent</i> in all other cases |
| Passive defence: | perform <i>DisturbThePassiveOpponent</i> if there is more than one opponent moving towards own goal perform <i>BlockTheGoalkick</i> in all other cases |

The final step is to choose the most promising action from the set of actions available. Different actions have been programmed for the goalkeeper:

1. if the ball moves slowly within the penalty area: movement towards the ball, unless the ball is between keeper and goal;

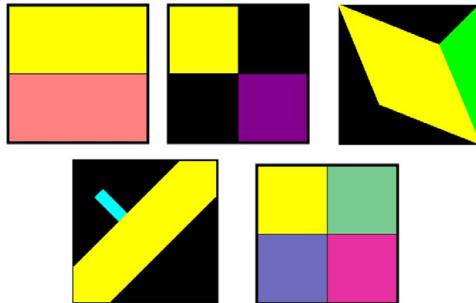


Fig. 13. Simple and complex designs of robot patches

2. if the ball is above/below the penalty area: movement along goal-line towards the upper/lower edge of penalty area;
3. if the ball moves towards the goal: take position on goal-line according to ball's calculated impact point;
4. if the ball is not moving towards the goal: take position on goal-line at ball level.

Obviously, robot soccer provides perfect examples for decisions that rely on vague and incomplete information. Fuzzy logic is predestined to cope with terms like "near", "high speed" and "defensive", and it seems promising to fuzzify rules like "if the opponent is near the penalty area and his speed is high, move towards the opponent to block as early as possible". Neural networks, especially self-organising maps, could also be effective e.g. in the analysis of the opponent's strategy and the evaluation of the overall situation on the field [13].

Image Processing for Robust Perception

Image processing is a crucial and sometimes underestimated part of robot soccer. The variety of colours, the speed of the game and the often unstable lighting conditions suggest that robot soccer is also an excellent platform to find attractive solutions for advanced industrial applications. Due to the high speed of the game, it is essential to work with at least 30 full frames/sec. The average camera resolution is currently 640* 480 pixels. Each team is identified by its team colour, a second colour is used to identify the alignment, and almost all teams use separate colours to identify individual team members. The image processing module has to handle 26 colours and the red ball in an 11 vs. 11 game.

The system of the Dortmund Droids consists of a highly modularised design that aims at flexibility, easy implementation and fast adaptation of the system. It is divided into a base system and a processing module. The base system is responsible for the control flow, triggering the recognition process and reporting its results to the overall robot soccer system. It also controls

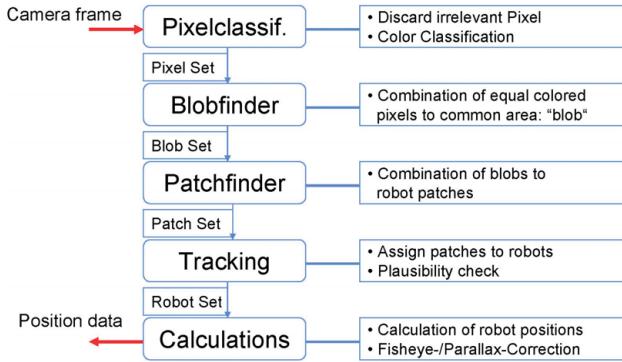


Fig. 14. Image processing: Control and data flow (simplified)

the camera hardware through an encapsulation module which makes it easy to switch to other camera types. The recognition process itself is modularised and consists of independent components which all contribute to the recognition process [15], see Fig. 14.

The first step of this recognition process is to find coherent areas of “similar” colours. This requires a method of pixel classification according to colour and relevance, which is frequently achieved by defining colour look-up tables (e.g. picking colours by hand in a freeze image). Regrettably, linear changes in human perception lead to non-linear changes in those colour spaces. As a consequence, vision systems are rather sensitive to changes in illumination density and colour temperature. Taking this aspect into account, the Dortmund Droids use the HLS colour space for recognition, see Fig. 15. The HLS colour space consists of the hue (pure colour hue), lightness (from black to white) and saturation (from fully unsaturated to fully saturated) dimensions. Under typical setting in robot soccer we need to recognize saturated light to dark colours only, i.e. all colour values that are close to white ($L = 1$) or black ($L = 0$), and all nearly unsaturated colours ($S = 0$) are classified as irrelevant for the recognition process. The field is dark and not saturated, so it is acceptable to take into consideration only pixels above a certain threshold. If a pixel is relevant, it is projected onto the hue circle of the HLS colour space (i.e.: the L value is set to 0.5, the S value to 1). Similar colours are then simply defined by their proximity on the hue circle [16].

If the relevant pixels have been detected, coherent areas of colors (called blobs) are identified by a standard region-growing algorithm. Finding objects, thereafter, means assigning blobs to disjoint groups. Each group of blobs forms a so-called patch. The design of a patch, see Fig. 13 is defined by the color of the team, the color for the identification of the alignment and in most cases a color for the object identification (the single team member). While considering single blobs to group into robot patches, the system calculates

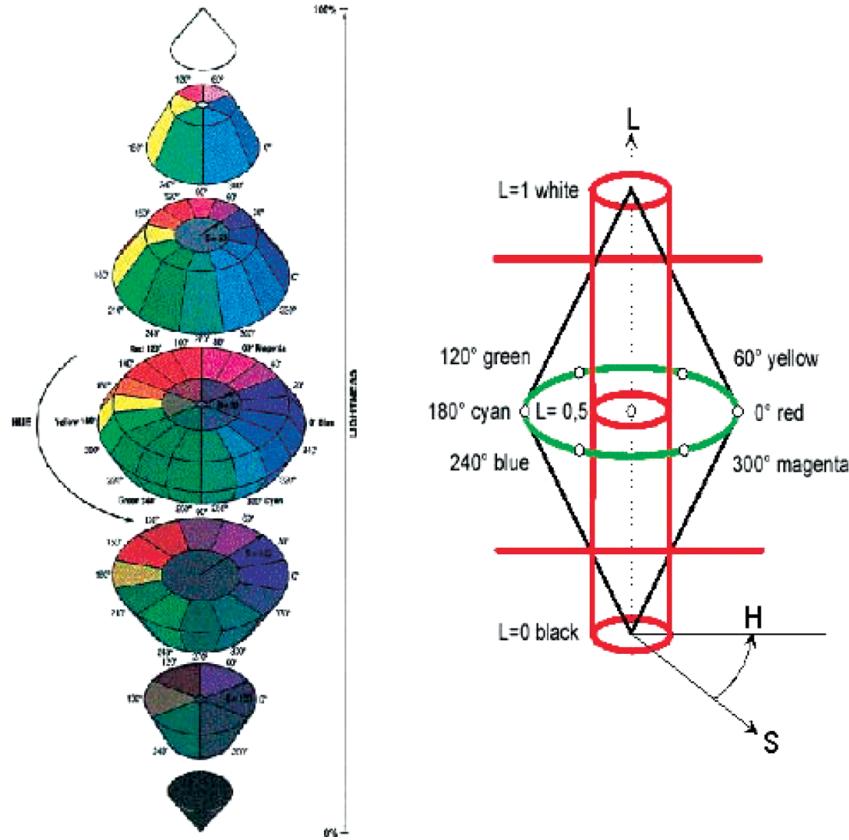


Fig. 15. The HSL colour model

quality measures for every blob found on the field. It considers area, exact color and geometry of the blobs.

Identifying a single player by a separate colour is an error-prone method. Therefore, the ability to track each object has significant advantages. But as a reliable identification of robots is absolutely essential to robot soccer, an object-tracking method must work extremely reliably. Simple greedy approaches that identify objects by matching a previous position to the closest current position are fast but unstable. Generalised optimisation methods that generate all possible assignments between previous and current object positions, choosing the assignment that minimises a least-squares criterion are not feasible because the complexity grows exponentially. The Dortmund Droids have implemented two other methods: greedy with local optimisation, and greedy with movement prediction. Both methods start with a greedy algorithm. All objects that cannot be identified greedily are grouped into clusters, and every cluster is then treated with a general optimisation method. The

greedy with movement prediction approach makes use of the fact that the system knows how a certain object moved in the last picture and can predict where that object has moved to.

Camera lenses inevitably produce fish-eye distortions into the picture. Therefore, as a final step, a distortion correction – that also compensate for the paralay effect – is essential while converting from a picture-based into a playing field-based coordinate system. A decision-making module finally decides on the identity of objects, i.e. it uniquely identifies objects and their positions.

The 440×220 cm-sized 11 vs. 11 field poses a new and demanding challenge. The cameras are currently mounted 2.5 m above the field, so strong fish-eye lenses are necessary to cover the whole field. This fish-eye effect impedes an exact calculation of the robots' positions. Raising the camera to a higher level is no option since this would rule out too many locations, so most teams playing in the 11 vs. 11 league have to revert to a multi-camera approach, where every camera records only a part of the field.

The next big step in image processing will be to include local vision of each robot and to merge these local images with the overall vision.

4 Summary

Robots are coming of age. From rigid automatons in the true sense of the word, they are slowly but surely developing into flexible protagonists capable of carrying out tasks autonomously. Even if the service robots currently on the market are only capable of simple tasks like vacuum- or window-cleaning: coming robot generations will be able to carry out more versatile actions, with characteristics that are in many respects more human. At the same time, and with all due respect to the visions of scientists in robot laboratories: there is still a long way to go before beings equal to humans will be created. The projects that focus on humanoid robots and robot soccer show the way, but only the future will tell where this will lead.

Acknowledgment

The author would like to thank Norman Weiss, Lars Hildebrand and Bernd Reusch for many nurturing discussions. Furthermore, I am indepted to many colleagues from the European FIRA-Chapter: Albert Schoute, Mannes Poel, Peter Kopacek, Man-Wook Han, Paul Robinson, Drago Matko and Olivier Simonin – to name just a few.

References

1. Arkin, Ronald C.: Just What is a Robot Architecture Anyway? Turing Equivalence versus Organizing Principles, Atlanta 1995 206
2. Brooks, Rodney A.: A Robust Layered Control System for Mobile Robot, M.I.T., A.I. Memo 864, September 1985 204
3. Bugmann, G.: Challenges in Verbal Instructions of Domestic Robots, Proceedings of the ASER'03, 1st International Workshop on Advances in Service Robotics, Bardolino, Italy, March 13–15, 2003, pp. 112–116 207
4. Cho, Hyungsuck; Kim, Jong-Hwan, Lee, Ju-Jang: Intelligent Robots: Vision, Learning and Interaction. Edited Volume of 2002 FIRA Robot World Congress Proceedings, KAIST Press 2003 210
5. Dortmund Droids team: <http://www.robosoccer.de> 213
6. Dr. Heuser AG: Market Study Automation Technologies. Invited Talk, European FIRA-Meeting, Dortmund, November 2002 199
7. Han, M.-W.; Kopacek, P.: Neural Networks for the Control of Soccer Robots, Internal Paper, TU Vienna, 2002 210
8. Hildebrand, L., Michalski, C., Valentin, H., Wickrath, M. Strategy Implementation For Mobile Robots Using The Pipes&Filters Architecture. Proc. FIRA World Congress 2003, Vienna, Austria
9. European Robotics Forum (IFR ERF) & European Robotics Research Network (EURON): European Robotics. A White Paper on the Status and Opportunities of the European Robotics Industry, Version 20 January 2003 (ed. By Henrik I. Christensen et al.) 199
10. Kim, J.-H., Kim, D.-H., Kim, Y.-J., Seow, K.-T. Springer Tracts in Advanced Robotics, Vol. 11: Soccer Robotics. Springer, Berlin etc. 2004 210
11. Nehmzow, U.: Mobile Robotics: A Practical Introduction. Springer etc. 2000 206
12. Ram, A., Santamaria, J. C.: Multistrategy Learning in Reactive Control Systems for Autonomous Robotic Navigation, *Informatica* 17 (4), pp. 347–369, 1993 206
13. Reuter, M.: Ruling Robots by the Activity Patterns of Hierachical SOMs, ISC 2003 (Eurosism03), Valencia, Spain, 2003 215
14. Schaft, R.D.; Hägele, M.; Wegener, K.: Service Roboter Visionen, München/Wien 2004 207
15. Weiss N., Konzeption und Implementierung einer flexiblen und robusten Positionserkennung für den Roboterfußball, University of Dortmund, Internal Reports, Dortmund, 2003 (in German) 216
16. Weiss, N., Jesse, N.. Towards Local Vision in Centralized Robot Soccer Leagues: A Robust And Flexible Vision System Also Allowing Varying Degrees of Robot Autonomy. Proc. FIRA World Congress 2004, Busan, Korea 216
17. Würzl, M.; Putz, B.: A New Generation of Soccer Robots, in: Proceedings of the 2nd FRA Robot Soccer World Congress, Vol. 2, Wien 2003, pp. 57–60 210
18. United Nations, International Federation of Robotics: World Robotics 2002, New York and Geneva 2002

Part II

Selected KES2004 Conference Tutorials

Rough Set Theory with Applications to Data Mining

J.W. Grzymala-Busse

Abstract. This paper is an introduction to rough set theory with an emphasis on applications to data mining. First, consistent data are discussed, including blocks of attribute-value pairs, reducts of information tables, indiscernibility relation, decision tables, and global and local coverings. Rule induction algorithms LEM1 and LEM2 are presented. Then the rough set approach to inconsistent data is introduced, with lower and upper approximations and certain and possible rule sets. The last topic is a rough set approach to incomplete data. How to define modified blocks of attribute-value pairs, characteristic sets, and characteristic relation are explained. Additionally, two definitions of definability and three definitions of approximations are presented. Finally, some remarks about applications of the LERS data mining system are included.

1 Introduction

Rough set theory was introduced in 1982 by Z. Pawlak, see [20]. As it is shown in [3], rough set theory represents an objective approach to imperfections in data, all computations are performed directly on data sets, i.e., no feedback from additional experts is necessary. Thus, there is no need for any additional information about data such as, for example, a probability distribution function in statistics, a grade of membership from fuzzy set theory, etc. [3].

This paper presents basic ideas of rough set theory and shows how these ideas may be utilized for data mining. In general, rough set theory may be applied to consistent data (without conflicting cases) to study relations between attributes. For example, this way we may eliminate redundant attributes. Another typical task is to find a minimal subset of the attribute set that may be used to identify all concepts. Yet another task is to compute a family of sets of attribute-value pairs for the same reason: to identify all concepts.

Inconsistent data sets are handled by rough set theory using lower and upper approximations for every concept. These approximations are definable using existing attributes. Furthermore, from concept lower and upper approximations certain and possible rule sets are induced.

Another important area of rough set theory applications are incomplete data, i.e., data with attribute missing values. Here rough set theory may be used, again, for computing generalized lower and upper approximations for all concepts.

In this paper we will restrict our attention to only one technique of data mining: rule induction.

2 Consistent Data

In this paper we will study data sets in two forms: as information tables and as decision tables. In both cases variables are presented in columns and cases in rows. In information tables all variables are called attributes while in decision tables one of the variables is called a decision, while the remaining are attributes.

2.1 Information Tables

An example of an information table is presented in Table 1. Four attributes: *Temperature*, *Headache*, *Nausea* and *Cough* characterize six cases.

Table 1. An information table

| Case | Attributes | | | |
|------|-------------|----------|--------|-------|
| | Temperature | Headache | Nausea | Cough |
| 1 | high | yes | no | yes |
| 2 | very_high | yes | yes | no |
| 3 | high | no | no | no |
| 4 | high | yes | yes | yes |
| 5 | normal | yes | no | no |
| 6 | normal | no | yes | yes |

Let U denote the set of all cases, A the set of all attributes, and V the set of all attribute values. Such a table defines an information function $\rho : U \times A \rightarrow V$. For example, $\rho(1, \text{Temperature}) = \text{high}$.

Let $a \in A, \nu \in V$, and $t = (a, \nu)$ be an attribute-value pair. A *block* of t , denoted by $[t]$, is a set of all cases from U for which attribute a has value ν . For the information table from Table 1,

$$\begin{aligned} [(\text{Temperature}, \text{high})] &= \{1, 3, 4\}, \\ [(\text{Temperature}, \text{very_high})] &= \{2\}, \\ [(\text{Temperature}, \text{normal})] &= \{5, 6\}, \\ [(\text{Headache}, \text{yes})] &= \{1, 2, 4, 5\}, \end{aligned}$$

$[(\text{Headache}, \text{no})] = \{3, 6\}$,
 $[(\text{Nausea}, \text{no})] = \{1, 3, 5\}$,
 $[(\text{Nausea}, \text{yes})] = \{2, 4, 6\}$,
 $[(\text{Cough}, \text{yes})] = \{1, 4, 6\}$, and
 $[(\text{Cough}, \text{no})] = \{2, 3, 5\}$.

Let $x \in U$ and $B \subseteq A$. An elementary set of B containing x , denoted by $[x]_B$, is the following set:

$$\bigcap \{[(a, \nu)] \mid a \in B, \rho(x, a) = \nu\}$$

Elementary sets are subset of U consisting all cases from U that are indistinguishable from x while using all attributes from B . In *soft computing* terminology elementary sets are called *information granules*. When subset B is restricted to a single attribute, elementary sets are blocks of attribute-value pairs defined by that specific attribute. Thus,

$[1]_{\{\text{Temperature}\}} = [3]_{\{\text{Temperature}\}} = [4]_{\{\text{Temperature}\}} = [(\text{Temperature, high})] = \{1, 3, 4\}$,
 $[2]_{\{\text{Temperature}\}} = [(\text{Temperature, very_high})] = \{2\}$,
 $[5]_{\{\text{Temperature}\}} = [6]_{\{\text{Temperature}\}} = [(\text{Temperature, normal})] = \{5, 6\}$.

Additionally, if $B = \{\text{Temperature, Headache}\}$,

$[1]_B = [4]_B = [(\text{Temperature, high})] \cap [(\text{Headache, yes})] = \{1, 4\}$,
 $[2]_B = [(\text{Temperature, very_high})] \cap [(\text{Headache, yes})] = \{2\}$,
 $[3]_B = [(\text{Temperature, high})] \cap [(\text{Headache, no})] = \{3\}$,
 $[5]_B = [(\text{Temperature, normal})] \cap [(\text{Headache, yes})] = \{5\}$, and
 $[6]_B = [(\text{Temperature, normal})] \cap [(\text{Headache, no})] = \{6\}$.

Elementary sets may be defined in another way, through the notion of an indiscernibility relation. Again, let B be a nonempty subset of the set A of all attributes. The *indiscernibility relation* $\text{IND}(B)$ is a binary relation on U defined for $x, y \in U$ as follows

$$(x, y) \in \text{IND}(B) \text{ if and only if } \rho(x, a) = \rho(y, a) \text{ for all } a \in B.$$

Obviously, $\text{IND}(B)$ is an equivalence relation. A convenient way to present equivalence relations is through partitions. A *partition* of U is a family of mutually disjoint nonempty subsets of U , called *blocks*, such that the union of all blocks is U . The partition induced by $\text{IND}(B)$ will be denoted by B^* . Blocks of B^* are also called *elementary sets* associated with B . For example,

$$\begin{aligned} \{\text{Temperature}\}^* &= \{\{1, 3, 4\}, \{2\}, \{5, 6\}\}, \\ \{\text{Temperature, Headache}\}^* &= \{\{1, 4\}, \{2\}, \{3\}, \{5\}, \{6\}\}. \end{aligned}$$

On the other hand,

$$\text{IND}(\{\text{Temperature}\}) = \{(1, 1), (1, 3), (1, 4), (2, 2), (3, 1), (3, 3), (3, 4), (4, 1), (4, 3), (4, 4), (5, 5), (5, 6), (6, 5), (6, 6)\}, \text{ and}$$

$$\text{IND}(\{\text{Temperature, Headache}\}) = \{(1, 1), (1, 4), (2, 2), (3, 3), (4, 1), (4, 4), (5, 5), (6, 6)\}.$$

There are important subsets of attributes called reducts. A subset B of the set A is called a *reduct* if and only if

1. $B^* = A^*$ and
2. B is minimal with this property, i.e., $(B - \{a\})^* \neq A^*$ for all $a \in B$.

For example, $\{\text{Temperature}\}$ is not a reduct since

$$\{\text{Temperature}\}^* = \{\{1, 3, 4\}, \{2\}, \{5, 6\}\} \neq A^* = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}\}.$$

Similarly, $\{\text{Temperature, Headache}\}$ is not a reduct since

$$\{\text{Temperature, Headache}\}^* = \{\{1, 4\}, \{2\}, \{3\}, \{5\}, \{6\}\} \neq A^* = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}\}.$$

On the other hand, $\{\text{Temperature, Headache, Nausea}\}$ is a reduct. The systematic way to compute all reducts is based on first checking all single attributes whether $\{a\}^* = A^*$, if so, the corresponding $\{a\}$ is a reduct. The next step is to check all subsets B of A with $|B| = 2$ such B is not a superset of any existing reduct, where $|X|$ is the cardinality of the set X . We check all of such subsets B whether $B^* = A^*$, if so, B is a reduct. The next step is to check all subsets B of set A of the cardinality equal to three such that B is not a superset of a reduct, etc. For the information table presented in Table 1,

$$\begin{aligned} &\{\text{Temperature}\}^* = A^*, \\ &\{\text{Headache}\}^* = A^*, \\ &\{\text{Nausea}\}^* = A^*, \\ &\{\text{Cough}\}^* = A^*, \end{aligned}$$

$$\begin{aligned} &\{\text{Temperature, Headache}\}^* = A^*, \\ &\{\text{Temperature, Nausea}\}^* = A^*, \\ &\{\text{Temperature, Cough}\}^* = A^*, \\ &\{\text{Headache, Nausea}\}^* = A^*, \\ &\{\text{Headache, Cough}\}^* = A^*, \\ &\{\text{Nausea, Cough}\}^* = A^*, \end{aligned}$$

$$\begin{aligned} &\{\text{Temperature, Headache, Nausea}\}^* = A^*, \\ &\{\text{Temperature, Headache, Cough}\}^* = A^*, \\ &\{\text{Temperature, Nausea, Cough}\}^* = A^*, \\ &\{\text{Headache, Nausea, Cough}\}^* = A^*. \end{aligned}$$

Therefore, reducts are:

$\{\text{Temperature}, \text{Headache}, \text{Nausea}\}$, $\{\text{Temperature}, \text{Nausea}, \text{Cough}\}$, and $\{\text{Headache}, \text{Nausea}, \text{Cough}\}$.

The above method of computing all reducts is of exponential worst time complexity with respect to the number of attributes. Therefore, in practice, we restrict our attention to compute a single reduct, using a heuristic algorithm. The first step of this algorithm is an attempt to eliminate the leftmost attribute a_1 of the set $\{a_1, a_2, \dots, a_n\} = A$ of all attributes. If

$$\{a_2, \dots, a_n\}^* = A^*$$

then a_1 can be eliminated for good, if not, we will put it back to the set. In the next step, we try to eliminate a_2 , and so on, in the last step, an attempt is to eliminate a_n . In our example, the first step is based on an attempt to eliminate *Temperature*, so we are checking whether

$$\{\text{Headache}, \text{Nausea}, \text{Cough}\} = A^*.$$

This attempt is successful. However,

$$\begin{aligned} &\{\text{Nausea}, \text{Cough}\} \quad A^*, \\ &\{\text{Headache}, \text{Cough}\} \quad A^*, \quad \text{and} \\ &\{\text{Headache}, \text{Nausea}\} \quad A^*, \end{aligned}$$

so $\{\text{Headache}, \text{Nausea}, \text{Cough}\}$ is a reduct.

2.2 Decision Tables

In a decision table variables, presented as columns, belong to either of two categories: attributes and decisions. Usually a decision table has only one decision. Rows, like in information tables, are labeled by case names. An example of a decision table is presented in Table 2. Attributes are: *Temperature*, *Headache*, *Nausea* and *Cough*, a decision is *Flu*.

In decision table from Table 2 there are two elementary sets of $\{\text{Flu}\}$: $\{1, 2, 4\}$ (for these cases *Flu* has value *yes*) and $\{3, 5, 6\}$ (for these cases *Flu* has value *no*). Elementary sets of *decision* are called *concepts*.

Table 2. A decision table

| Case | Attributes | | | | Decision |
|------|-------------|----------|--------|-------|----------|
| | Temperature | Headache | Nausea | Cough | |
| 1 | high | yes | no | yes | yes |
| 2 | very_high | yes | yes | no | yes |
| 3 | high | no | no | no | no |
| 4 | high | yes | yes | yes | yes |
| 5 | normal | yes | no | no | no |
| 6 | normal | no | yes | yes | no |

Decision tables present cases that are classified or diagnosed by experts. For example, attributes may be interpreted as medical tests, and decision may correspond to a disease. Decision tables are crucial for data mining (or knowledge acquisition, or machine learning).

There are two main approaches to data mining from complete data sets based on rough set theory. In both approaches decision tables are used. The first approach is *global*: the entire attributes are used for analysis. The second possibility is *local*, blocks of attribute-value pairs are used.

Global Coverings

Rule sets may be induced using global coverings [4], also called relative reducts [21]. We will start from the definition of a partition being *finer* than another partition. Let α and β be partitions of U . We say that α is *finer* than β , denoted $\alpha \leq \beta$, if and only if for each block X of α there exists a block Y of β such that $X \subseteq Y$.

Let d be a decision. A subset B of the attribute set A is a *global covering* if and only if

1. $B^* \leq \{d\}^*$ and
2. B is minimal with this property, i.e., $(B - \{a\})^* \leq \{d\}^*$ is false for any $a \in B$.

For example, $\{\text{Temperature}\}$ is not a global covering since $\{\text{Temperature}\}^* = \{\{1, 3, 4\}, \{2\}, \{5, 6\}\}$ is not finer than $\{\text{Flu}\}^* = \{\{1, 2, 4\}, \{3, 5, 6\}\}$. However, $\{\text{Temperature, Headache}\}$ is a global covering since

$$\{\text{Temperature, Headache}\}^* = \{\{1, 4\}, \{2\}, \{3\}, \{5\}, \{6\}\} \leq \{\{\text{Flu}\}^* = \{\{1, 2, 4\}, \{3, 5, 6\}\}\}$$

Algorithms for computing all global coverings and a single global covering, presented in [4], are similar to algorithms for computing all reducts and a single reduct. However, first we need to check whether

$$A^* \leq \{d\}^*.$$

If this condition is not satisfied, there is no one single global covering. In our example from Table 2,

$$A^* = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}\} \leq \{\text{Flu}\}^* = \{\{1, 2, 4\}, \{3, 5, 6\}\},$$

hence we may start the main algorithm to compute all global coverings. We start from checking all single attributes

$$\begin{aligned} \{\text{Temperature}\}^* &\leq \{\text{Flu}\}^* \text{ is false,} \\ \{\text{Headache}\}^* &\leq \{\text{Flu}\}^* \text{ is false,} \\ \{\text{Nausea}\}^* &\leq \{\text{Flu}\}^* \text{ is false,} \\ \{\text{Cough}\}^* &\leq \{\text{Flu}\}^* \text{ is false,} \end{aligned}$$

therefore, there is no one global covering of size one. Then we are checking all subsets of A with the cardinality equal to two

$$\begin{aligned}\{\text{Temperature, Headache}\}^* &\leq \{\text{Flu}\}^* \text{ is true,} \\ \{\text{Temperature, Nausea}\}^* &\leq \{\text{Flu}\}^* \text{ is false,} \\ \{\text{Temperature, Cough}\}^* &\leq \{\text{Flu}\}^* \text{ is true,} \\ \{\text{Headache, Nausea}\}^* &\leq \{\text{Flu}\}^* \text{ is false,} \\ \{\text{Headache, Cough}\}^* &\leq \{\text{Flu}\}^* \text{ is false,} \\ \{\text{Nausea, Cough}\}^* &\leq \{\text{Flu}\}^* \text{ is false,}\end{aligned}$$

so there are two global coverings of size two: $\{\text{Temperature, Headache}\}$ and $\{\text{Temperature, Cough}\}$. There is only one subset of A of the cardinality equal to three that is not a superset of existing global coverings, and that is $\{\text{Headache, Nausea, Cough}\}$. This set is tested in the next step of the algorithm

$$\{\text{Headache, Nausea, Cough}\}^* \leq \{\text{Flu}\}^* \text{ is true,}$$

so there is one global covering of size three. Obviously, the worst time complexity of the algorithm for computing all global coverings is the same as the algorithm for computing all reducts, i.e., exponential. Thus we should restrict our attention to computing a single global covering. The corresponding algorithm is an extension of the algorithm to compute a single reduct. First we need to test whether

$$A^* \leq \{d\}^*.$$

Then we will try to eliminate the first attribute, then the second one, etc. Since $\{\text{Headache, Nausea, Cough}\}^* \leq \{\text{Flu}\}^*$, we eliminate *Temperature*. However,

$$\begin{aligned}\{\text{Nausea, Cough}\}^* &\leq \{\text{Flu}\}^* \text{ is false,} \\ \{\text{Headache, Cough}\}^* &\leq \{\text{Flu}\}^* \text{ is false,} \\ \{\text{Headache, Nausea}\}^* &\leq \{\text{Flu}\}^* \text{ is false,}\end{aligned}$$

so this algorithm will return the following single global covering: $\{\text{Headache, Nausea, Cough}\}$. A single global covering may be used for rule induction. We restrict our attention to attributes from the global covering and then, for every case, we will check whether rule conditions can be dropped. Thus, for the first case our original rule, induced from Table 2, is

$$(\text{Headache, yes}) \ \& \ (\text{Nausea, no}) \ \& \ (\text{Cough, yes}) \rightarrow (\text{Flu, yes}).$$

We may drop the leftmost condition (*Headache, yes*) since

$$(\text{Nausea, no}) \ \& \ (\text{Cough, yes}) \rightarrow (\text{Flu, yes}).$$

is consistent with the Table 2. However, this rule cannot be further simplified since

$$(\text{Cough, yes}) \rightarrow (\text{Flu, yes}).$$

is not consistent with Table 2 (due to case 6 that will be misclassified by this rule), also

$$(\text{Nausea, no}) \rightarrow (\text{Flu, yes}).$$

is not consistent with Table 2 (due to cases 3 and 5). This rule covers only case 1. The next uncovered case is 2, the induced rule is

$$(\text{Headache, yes}) \& (\text{Nausea, yes}) \& (\text{Cough, no}) \rightarrow (\text{Flu, yes}).$$

The first condition can be dropped, the remaining two conditions cannot be dropped. The new rule, covering only case 2, is

$$(\text{Nausea, yes}) \& (\text{Cough, no}) \rightarrow (\text{Flu, yes}).$$

The case 3 implies the following rule

$$(\text{Headache, no}) \& (\text{Nausea, no}) \& (\text{Cough, no}) \rightarrow (\text{Flu, no}).$$

Again, the first condition may be dropped, the remaining two conditions cannot be dropped. The rule

$$(\text{Nausea, no}) \& (\text{Cough, no}) \rightarrow (\text{Flu, no}).$$

covers two cases, 3 and 5. Case 4 implies the rule

$$(\text{Headache, yes}) \& (\text{Nausea, yes}) \& (\text{Cough, yes}) \rightarrow (\text{Flu, yes}).$$

Here the first condition cannot be dropped, but we may drop the second condition

$$(\text{Headache, yes}) \& (\text{Cough, yes}) \rightarrow (\text{Flu, yes}).$$

Note that this rule covers two cases, 1 and 4. The last uncovered case is 6, the implied rule is

$$(\text{Headache, no}) \& (\text{Nausea, yes}) \& (\text{Cough, yes}) \rightarrow (\text{Flu, no}).$$

The first condition cannot be dropped, but we may drop the second condition to obtain the following rule

$$(\text{Headache, no}) \& (\text{Cough, yes}) \rightarrow (\text{Flu, no}).$$

The rightmost condition may be dropped as well, our final rule is

$$(\text{Headache, no}) \rightarrow (\text{Flu, no}).$$

This rule cannot be further simplified. It covers two cases, 3 and 6. At the end we should eliminate the redundant rule

$$(\text{Nausea, no}) \& (\text{Cough, yes}) \rightarrow (\text{Flu, yes}).$$

since it covers only one case (1), while the rule

$(\text{Headache, yes}) \ \& \ (\text{Cough, yes}) \rightarrow (\text{Flu, yes}).$

covers two cases: 1 and 4. Thus, the final rule set is

2, 1, 1
 $(\text{Nausea, yes}) \ \& \ (\text{Cough, no}) \rightarrow (\text{Flu, yes}).$

2, 2, 2
 $(\text{Headache, yes}) \ \& \ (\text{Cough, yes}) \rightarrow (\text{Flu, yes}),$

2, 2, 2
 $(\text{Nausea, no}) \ \& \ (\text{Cough, no}) \rightarrow (\text{Flu, no}),$

1, 2, 2
 $(\text{Headache, no}) \rightarrow (\text{Flu, no}).$

Rules are presented in the LERS format (every rule is equipped with three numbers, the total number of attribute-value pairs on the left-hand side of the rule, the total number of cases correctly classified by the rule during training, and the total number of training cases matching the left-hand side of the rule).

The above algorithm is a basis for the algorithm LEM1 (Learning from Examples, Module 1) of the data mining system LERS (Learning from Examples based on Rough Sets), see [6].

Local Coverings

First we will introduce the idea of a minimal complex, which corresponds to a single rule. Let X be a concept. Let t be an attribute-value pair (a, v) , and let T be a set of attribute-value pairs. Then the *block* of t , denoted $[t]$, is the set of examples for which attribute a has value v . Set X depends on a set T of attribute-value pairs, if and only if

$$\emptyset \cap \{[t] \mid t \in T\} \subseteq X$$

A set T is a *minimal complex* of X if and only if X depends on T and no proper subset T' of T exists such that X depends on T' .

A minimal complex for $\{1, 2, 4\}$ (i.e., for $[(\text{Flu, yes})]$) is $\{(\text{Temperature, high}), (\text{Headache, yes})\}$.

Now we may introduce a local covering, which corresponds to a rule set describing a concept. Similarly to global coverings, local coverings are also useful for rule induction. Let \mathbb{T} be a non-empty collection of non-empty sets of attribute value pairs. Then \mathbb{T} is a *local covering* of X if and only if the following conditions are satisfied:

1. Each member T of \mathbb{T} is a minimal complex of X ,
2. $\cup\{T \mid T \in \mathbb{T}\} = X$, and
3. \mathbb{T} is minimal, i.e., \mathbb{T} has the smallest possible number of members.

An example of a local covering for $\{1, 2, 4\}$ (i.e., for $[(\text{Flu}, \text{yes})]$) is $\{\{(\text{Temperature}, \text{high}), (\text{Headache}, \text{yes})\}, \{(\text{Temperature}, \text{very_high})\}\}$.

The algorithm LEM2 (Learning from Examples Module, version 2) for rule induction is based on computing a single local covering for each concept from a decision table [1, 6]. The user may select an option of LEM2 with or without taking into account attribute priorities. When LEM2 does not take attribute priorities into account, the first criterion is ignored of the following procedure. The procedure LEM2 with attribute priority is presented below.

```
Procedure LEM2
  (input: a set  $X$ ;
   output: a single local covering  $\mathbb{T}$  of set  $X$ );
  begin
     $G := X$ ;
     $\mathbb{T} := \emptyset$ ;
    while  $G \neq \emptyset$  do
      begin
         $T := \emptyset$ 
         $T(G) := \{t \mid [t] \cap G \neq \emptyset\}$ ;
        while  $T = \emptyset$  or not  $([T] \subseteq X)$ 
          begin
            select a pair  $t \in T(G)$  with the highest attribute
            priority, if a tie occurs, select a pair  $t \in T(G)$ 
            such that  $|[t] \cap G|$  is maximum; if
            another tie occurs, select a pair  $t \in T(G)$  with the
            smallest cardinality of  $[t]$ ; if a further tie occurs,
            select first pair;
             $T := T \cup \{t\}$ ;
             $G := [t] \cap G$ ;
             $T(G) := \{t \mid [t] \cap G \neq \emptyset\}$ ;
             $T(G) := T(G) - T$ ;
          end; {while}
        for each  $t$  in  $T$  do
          if  $[T - \{t\}] \subseteq X$  then  $T := T - \{t\}$ ;
         $\mathbb{T} := \mathbb{T} \cup \{T\}$ ;
         $G := X - \cup_{T \in \mathbb{T}} [T]$ ;
      end {while};
      for each  $T \in \mathbb{T}$  do
        if  $\cup_{S \in \mathbb{T} - \{T\}} [S] = X$  then  $\mathbb{T} := \mathbb{T} - \{T\}$ ;
    end {procedure}.
```

The algorithm LEM2 induced the following rule set, presented in the LERS format, from Table 2

2, 2, 2
 $(\text{Headache, yes}) \ \& \ (\text{Temperature, high}) \rightarrow (\text{Flu, yes})$

1, 1, 1
 $(\text{Temperature, very_high}) \rightarrow (\text{Flu, yes})$

1, 2, 2
 $(\text{Headache, no}) \rightarrow (\text{Flu, no})$

1, 2, 2
 $(\text{Temperature, normal}) \rightarrow (\text{Flu, no})$

3 Inconsistent Data

In many cases data mining is performed on inconsistent data, i.e., on data in which there exist cases that are characterized by the same attribute values yet they were classified as members of different concepts. Such cases are called conflicting. Usually it means that in the data set some attributes are missing. Rough set theory is a perfect tool to handle inconsistent data [21, 22]. Using rough set theory, conflicting cases are not removed from the data set. Instead, concepts are approximated by new sets called lower and upper approximations. An example of the inconsistent data set, taken from [12], is presented in Table 3.

Table 3. An inconsistent decision table

| Case | Attributes | | | Decision |
|------|-------------|----------|--------|----------|
| | Temperature | Headache | Nausea | |
| 1 | high | yes | no | yes |
| 2 | very_high | yes | yes | yes |
| 3 | high | no | no | no |
| 4 | high | yes | yes | yes |
| 5 | high | yes | yes | no |
| 6 | normal | yes | no | no |
| 7 | normal | no | yes | no |
| 8 | normal | yes | no | yes |

A decision table is inconsistent if and only if

$$A^* \leq \{d\}^* \text{ is false .}$$

In the example from Table 3,

$$\{\text{Temperature, Headache, Nausea}\}^* = \{\{1\}, \{2\}, \{3\}, \{4, 5\}, \{6, 8\}, \{7\}\},$$

$$\{\text{Flu}\}^* = \{(1, 2, 4, 8), \{3, 5, 6, 7\}\},$$

so

$$\{\text{Temperature, Headache, Nausea}\}^* \leq \{\text{Flu}\}^* \text{ is false.}$$

The decision table from Table 3 is inconsistent since there exist conflicting cases 4 and 5 (and, independently, conflicting cases 6 and 8).

An obvious question is what subsets of the set U of all cases can be uniquely identified using attributes. Let B be a subset of the set A of attributes. Any union of elementary sets of B is called a *B-definable* set. An example of an A -definable set is $\{2, 3, 4, 5\}$. However, neither of the two concepts: $\{1, 2, 4, 8\}$ and $\{3, 5, 6, 7\}$ is A -definable. In rough set theory, for every concept X , two subsets of the set U of all cases are defined: lower and upper approximations. There exist two definitions of these approximations. For data sets in which all values are specified (i.e., no attribute value is missing), as in Table 3, these two definitions result in the same sets [21]. According to the first definition, a *B-lower approximation* of X , denoted by $\underline{B}X$, is equal to the following set

$$\{x \in U | [x]_B \subseteq X\},$$

and a *B-upper approximation* of X , denoted by $\overline{B}X$, is equal to the following set

$$\{x \in U | [x]_B \cap X \neq \emptyset\}.$$

The second definition of lower and upper approximations provides the following formulas

$$\underline{B}X = \cup\{[x]_B | x \in U, [x]_B \subseteq X\},$$

and

$$\overline{B}X = \cup\{[x]_B | x \in U, [x]_B \cap X \neq \emptyset\}.$$

Thus, for both concepts from Table 3, $\{1, 2, 4, 8\}$ and $\{3, 5, 6, 7\}$, lower and upper approximations are

$$\begin{aligned}\underline{A}\{1, 2, 4, 8\} &= \{1, 2\}, \\ \underline{A}\{3, 5, 6, 7\} &= \{3, 7\}, \\ \overline{A}\{1, 2, 4, 8\} &= \{1, 2, 4, 5, 6, 8\}, \quad \text{and} \\ \overline{A}\{3, 5, 6, 7\} &= \{3, 4, 5, 6, 7, 8\}.\end{aligned}$$

Let Y be any subset of the set U of all cases and let B be a subset of the set A of all attributes. The lower approximation $\underline{B}Y$ is the largest definable set contained in Y . On the other hand, the upper approximation $\overline{B}Y$ is the smallest definable set containing Y . Moreover,

$$\underline{B}Y \subseteq Y \subseteq \overline{B}Y$$

hence any case from $\underline{B}Y$ is *certainly* a member of Y , while any member of $\overline{B}Y$ is *possibly* (or *plausibly*) a member of Y .

The above idea is utilized for data mining. For example, the LERS data mining system handles computing lower and upper approximation for every concept, and then inducing certain rule sets from lower approximations and possible rule sets from upper approximations. For details of LERS see, e.g., [6, 7]. The performance of LERS is fully comparable with performance of AQ15 [19] and C4.5 [23], see Table 4.

Table 4. Performance of AQ15, C4.5 and LERS

| Data Set | Error Rate | | |
|---------------|------------|------|------|
| | AQ15 | C4.5 | LERS |
| Lymphography | 18–20% | 23% | 19% |
| Breast cancer | 32–34% | 33% | 30% |
| Primary tumor | 59–71% | 60% | 67% |

Recently [8] LEM2 algorithm was enhanced by a new version of LEM2, called MLEM2 (Modified Learning from Examples Module, version 2). MLEM2 induces rule sets directly from data with symbolic and numerical attributes, while LEM2 needs pre-discretized data. Additionally, MLEM2 induces rule sets from data with missing attribute values, see the next section.

The algorithm LEM2 induced the following rule set from the decision table presented in Table 3

Certain rule set:

1, 1, 1
 (Temperature, very_high) → (Flu, yes),

3, 1, 1
 (Temperature, high) & (Nausea, no) & (Headache, yes) → (Flu, yes),

1, 2, 2
 (Headache, no) → (Flu, no),

and possible rule set:

1, 4, 6
 (Headache, yes) → (Flu, yes),

1, 2, 3
 (Temperature, normal) → (Flu, no),

2, 1, 2
 (Temperature, high) & (Nausea, yes) → (Flu, no),

1, 2, 2
 (Headache, no) → (Flu, no).

4 Incomplete Data

Input data sets are frequently incomplete, i.e., some attribute values are missing. In other words, corresponding decision tables are incomplete. In general, in data mining two approaches are used to handle incomplete data:

pre-processing of input data and then main processing of data mining such as rule induction. Typically, preprocessing means replacing missing attribute values by the most common value, ignoring cases with missing attribute values, etc [13]. Knowledge is acquired directly from incomplete data sets taking into account that some attribute values are missing. Typical systems using this approach are C4.5 and MLEM2.

In this paper we will apply the latter approach, i.e., rule induction is performed directly from incomplete data. Furthermore, we will assume that there are two reasons for data to be incomplete. The first reason is that an attribute value, for a specific case, is lost. This may happen when the original value was erased or mistakenly not included into the data set. The second reason for incompleteness is based on the lack of relevance, e.g., given case was diagnosed on the basis of some attribute values while other attribute values were irrelevant. For example, it was possible to diagnose a patient using only selected results of tests (attributes), while other test results were redundant. Such missing attribute values will be called “do not care” conditions. We will assume that in the same decision table some attribute values are lost and some are “do not care” conditions. Such decision tables were studied in [9].

Incomplete data with lost attribute values, from the viewpoint of rough set theory, were studied for the first time in [15]. In this paper two algorithms for rule induction, modified to handle lost attribute values, were presented. This approach was studied later in [25, 26, 27], where the indiscernibility relation was generalized to describe such incomplete data.

On the other hand, incomplete data in which all missing attribute values are “do not care” conditions, again from the view point of rough set theory, were studied for the first time in [5], where a method for rule induction was introduced in which each missing attribute value was replaced by all values from the domain of the attribute. Originally such values were replaced by all values from the entire domain of the attribute, later by attribute values restricted to the same concept to which a case with a missing attribute value belongs. Such incomplete decision tables, with all missing attribute values being “do not care conditions”, were extensively studied in [16, 17], including extending the idea of the indiscernibility relation to describe such incomplete decision tables.

In general, incomplete decision tables are described by characteristic relations, in a similar way as complete decision tables are described by indiscernibility relations [9, 10, 11, 14]. Also, elementary sets are replaced by characteristic sets. For complete decision tables, once the indiscernibility relation is fixed and the concept (a set of cases) is given, the lower and upper approximations are unique. For incomplete decision tables, for a given characteristic relation and the concept, there are three different possible ways to define lower and upper approximations, called singleton, subset, and concept approximations [9]. The singleton lower and upper approximations were studied in [16, 17, 25, 26, 27]. Similar ideas were studied in [2, 24, 28, 29, 30]. As it was observed in [9], singleton lower and upper approximations are not applicable in data mining.

4.1 Blocks of Attribute-Value Pairs, Characteristic Sets, and Characteristic Relation

In the sequel we will assume that all decision values are specified, i.e., they are not missing. Also, we will assume that all missing attribute values are denoted by “?” and by “*”, lost values will be denoted by “?” and “do not care” conditions will be denoted by “*”. Additionally, we will assume that for each case at least one attribute value is specified. An example of an incomplete table, taken from [12], is presented in Table 5.

Table 5. An incomplete decision table

| Case | Attributes | | | Decision |
|------|-------------|----------|--------|----------|
| | Temperature | Headache | Nausea | |
| 1 | high | ? | no | yes |
| 2 | very_high | yes | yes | yes |
| 3 | ? | no | no | no |
| 4 | high | yes | yes | yes |
| 5 | high | ? | yes | no |
| 6 | normal | yes | no | no |
| 7 | normal | no | yes | no |
| 8 | * | yes | * | yes |

For incomplete decision tables the definition of a block of an attribute-value pair must be modified.

If an attribute a there exists a case x such that $\rho(x, a) = ?$, i.e., the corresponding value is lost, then the case x should not be included in any block $[(a, v)]$ for all values v of attribute a .

If for an attribute a there exists a case x such that the corresponding value is a “do not care” condition, i.e., $\rho(x, a) = *$, then the

corresponding case x should be included in blocks $[(a, v)]$ for all specified values v of attribute a .

For the example of an incomplete data set from Table 5,

$$\begin{aligned} [(\text{Temperature}, \text{high})] &= \{1, 4, 5, 8\}, \\ [(\text{Temperature}, \text{very_high})] &= \{2, 8\}, \\ [(\text{Temperature}, \text{normal})] &= \{6, 7, 8\}, \\ [(\text{Headache}, \text{yes})] &= \{2, 4, 6, 8\}, \\ [(\text{Headache}, \text{no})] &= \{3, 7\}, \\ [(\text{Nausea}, \text{no})] &= \{1, 3, 6, 8\}, \\ [(\text{Nausea}, \text{yes})] &= \{2, 4, 5, 7, 8\}. \end{aligned}$$

The *characteristic set* $K_B(x)$ is the intersection of blocks of attribute-value pairs (a, v) for all attributes a from B for which $\rho(x, a)$ is specified and $\rho(x, a) = v$. For Table 2 and $B = A$,

$$\begin{aligned} K_A(1) &= \{1, 4, 5, 8\} \cap \{1, 3, 6, 8\} = \{1, 8\}, \\ K_A(2) &= \{2, 8\} \cap \{2, 4, 6, 8\} \cap \{2, 4, 5, 7, 8\} = \{2, 8\}, \\ K_A(3) &= \{3, 7\} \cap \{1, 3, 6, 8\} = \{3\}, \\ K_A(4) &= \{1, 4, 5, 8\} \cap \{2, 4, 6, 8\} \cap \{2, 4, 5, 7, 8\} = \{4, 8\}, \\ K_A(5) &= \{1, 4, 5, 8\} \cap \{2, 4, 5, 7, 8\} = \{4, 5, 8\}, \\ K_A(6) &= \{6, 7, 8\} \cap \{2, 4, 6, 8\} \cap \{1, 3, 6, 8\} = \{6, 8\}, \\ K_A(7) &= \{6, 7, 8\} \cap \{3, 7\} \cap \{2, 4, 5, 7, 8\} = \{7\}, \text{ and} \\ K_A(8) &= \{2, 4, 6, 8\}. \end{aligned}$$

Characteristic set $K_B(x)$ may be interpreted as the smallest set of cases that are indistinguishable from x using all attributes from B and using a given interpretation of missing attribute values. Thus, $K_A(x)$ is the set of all cases that cannot be distinguished from x using all attributes.

The *characteristic relation* $R(B)$ is a relation on U defined for $x, y \in U$ as follows

$$(x, y) \in R(B) \text{ if and only if } y \in K_B(x).$$

The characteristic relation $R(B)$ is reflexive but—in general—does not need to be symmetric or transitive. Also, the characteristic relation $R(B)$ is known if we know characteristic sets $K_B(x)$ for all $x \in U$. In our example,

$$R(A) = \{(1, 1), (1, 8), (2, 2), (2, 8), (3, 3), (4, 4), (4, 8), (5, 4), (5, 5), (5, 8), (6, 6), (6, 8), (7, 7), (8, 2), (8, 4), (8, 6), (8, 8)\}.$$

4.2 Approximations

The definition of definability of completely specified data should be modified to cover incomplete data. There exist two different ways to define definable sets. According to the first definition, a union of characteristic sets, associated

with B , will be called a *B-globally-definable* set. The second definition of definability is different: a union of intersections of blocks of attribute-value pairs will be called a *B-locally-definable* set. In the example of Table 5, the set $\{7, 8\}$ is *A-locally-definable* since it is equal to the intersection of $[(\text{Temperature}, \text{normal})]$ and $[(\text{Nausea}, \text{yes})]$. Nevertheless, $\{6, 7\}$ is not *A-globally-definable*. Obviously, any *B-globally definable* set is a *B-locally definable* set, but the converse is false. *A-locally definable* sets have the following important property: they can be described using rules.

Let X be a concept. In general, X is not *B-globally-definable*. As for completely specified decision tables, set X may be approximated by two *B-globally-definable* sets, a *B-lower approximation of X* , denoted by $\underline{B}X$ and a *B-upper approximation of X* , denoted by $\overline{B}X$.

However, for incompletely specified decision tables lower and upper approximations may be defined in a few different ways. Following [9], we are going to define three different approximations. Our first definition uses a similar idea as in the previous articles on incompletely specified decision tables [16, 17, 25, 26, 27], i.e., lower and upper approximations are sets of singletons from the universe U satisfying some properties. Thus we are defining lower and upper approximations by analogy with the first definition, by constructing both sets from singletons. We will call these definitions *singleton*. A singleton *B-lower approximation of X* is defined as follows:

$$\underline{B}X = \{x \in U | K_B(x) \subseteq X\}.$$

A singleton *B-upper approximation of X* is

$$\overline{B}X = \{x \in U | K_B(x) \cap X \neq \emptyset\}.$$

In our example presented in Table 2 let us say that $B = A$. Then the singleton *A-lower* and *A-upper* approximations of the two concepts: $\{1, 2, 4, 8\}$ and $\{3, 5, 6, 7\}$ are:

$$\begin{aligned}\underline{A}\{1, 2, 4, 8\} &= \{1, 2, 4\}, \\ \underline{A}\{3, 5, 6, 7\} &= \{3, 7\}, \\ \overline{A}\{1, 2, 4, 8\} &= \{1, 2, 4, 5, 6, 8\}, \\ \overline{A}\{3, 5, 6, 7\} &= \{3, 5, 6, 7, 8\}.\end{aligned}$$

Note that $\underline{A}\{1, 2, 4, 8\} = \{1, 2, 4\}$. But the set $\{1, 2, 4\}$ is not *A-globally-definable*. Furthermore, the set $\{1, 2, 4\}$ is not even *A-locally-definable*, so no set of rules can cover precisely this set. Similarly, $\overline{A}\{3, 5, 6, 7\} = \{3, 5, 6, 7, 8\}$, and the set $\{3, 5, 6, 7, 8\}$ is also not *A-locally-definable*. Therefore, in general, singleton approximations should not be used for data mining.

The second method of defining lower and upper approximations for complete decision tables uses another idea: lower and upper approximations are unions of elementary sets, subsets of U [9]. Therefore we may define lower and upper approximations for incomplete decision tables by analogy with the

second definition of approximations for completely specified data, using characteristic sets instead of elementary sets. There are two ways to do this. Using the first way, a *subset B*-lower approximation of X is defined as follows:

$$\underline{B}X = \cup\{K_B(x) | x \in U, K_B(x) \subseteq X\}.$$

A *subset B*-upper approximation of X is

$$\overline{B}X = \cup\{K_B(x) | x \in U, K_B(x) \cap X \neq \emptyset\}.$$

Since any characteristic relation $R(B)$ is reflexive, for any concept X , singleton B -lower and B -upper approximations of X are subsets of subset B -lower and B -upper approximations of X , respectively. For the same the decision presented in Table 2, the subset A -lower and A -upper approximations are:

$$\begin{aligned}\underline{A}\{1, 2, 4, 8\} &= \{1, 2, 4, 8\}, \\ \underline{\underline{A}}\{3, 5, 6, 7\} &= \{3, 7\}, \\ \overline{A}\{1, 2, 4, 8\} &= \{1, 2, 4, 5, 6, 8\}, \\ \overline{\overline{A}}\{3, 5, 6, 7\} &= \{2, 3, 4, 5, 6, 7, 8\}.\end{aligned}$$

The second possibility is to modify the subset definition of lower and upper approximation by replacing the universe U from the subset definition by a concept X . A *concept B*-lower approximation of the concept X is defined as follows:

$$\underline{BX} = \cup\{K_B(x) | x \in X, K_B(x) \subseteq X\}.$$

Obviously, the subset B -lower approximation of X is the same set as the concept B -lower approximation of X . A *concept B*-upper approximation of the concept X is defined as follows:

$$\overline{BX} = \cup\{K_B(x) | x \in X, K_B(x) \cap X \neq \emptyset\} = \cup\{K_B(x) | x \in X\}.$$

The concept B -upper approximation of X are subsets of the subset B -upper approximations of X . For the decision presented in Table 2, the concept A -lower and A -upper approximations are:

$$\begin{aligned}\underline{A}\{1, 2, 4, 8\} &= \{1, 2, 4, 8\}, \\ \underline{\underline{A}}\{3, 5, 6, 7\} &= \{3, 7\}, \\ \overline{A}\{1, 2, 4, 8\} &= \{1, 2, 4, 6, 8\}, \\ \overline{\overline{A}}\{3, 5, 6, 7\} &= \{3, 4, 5, 6, 7, 8\}.\end{aligned}$$

For complete decision tables, all three definitions of lower approximations, singleton, subset and concept, coalesce to the same definition. Also, for complete decision tables, all three definitions of upper approximations coalesce to the same definition. This is not true for incomplete decision tables, as our example shows.

Singleton B -lower and B -upper approximations of the set X are subsets of the subset B -lower and B -upper approximations of X , respectively. The

subset B -lower approximation of X is the same set as the concept B -lower approximation of X . The concept B -upper approximation of X is a subset of the subset B -upper approximation of X . The concept B -upper approximations are the smallest B -definable sets containing X .

Rules in LERS format induced from Table 5 using concept approximations are:

the certain rule set:

2, 2, 2
 $(\text{Temperature, high}) \ \& \ (\text{Nausea, no}) \rightarrow (\text{Flu, yes})$

2, 3, 3
 $(\text{Headache, yes}) \ \& \ (\text{Nausea, yes}) \rightarrow (\text{Flu, yes})$

1, 2, 2
 $(\text{Headache, no}) \rightarrow (\text{Flu, no})$

and the possible rule set:

2, 2, 2
 $(\text{Temperature, high}) \ \& \ (\text{Nausea, no}) \rightarrow (\text{Flu, yes})$

1, 3, 4
 $(\text{Headache, yes}) \rightarrow (\text{Flu, yes})$

2, 1, 3
 $(\text{Temperature, high}) \ \& \ (\text{Nausea, yes}) \rightarrow (\text{Flu, no})$

1, 2, 3
 $(\text{Temperature, normal}) \rightarrow (\text{Flu, no})$

1, 2, 2
 $(\text{Headache, no}) \rightarrow (\text{Flu, no})$

5 Final Remarks

A successful example of rough set theory application to data mining is the LERS data mining system. The machine learning/ data mining system LERS has proven its applicability having been used for years by NASA Johnson Space Center (Automation and Robotics Division), as a tool to develop expert systems of the type most likely to be used in medical decision - making on board the International Space Station. LERS was also used to enhance facility compliance under Sections 311, 312, and 313 of Title III, the Emergency Planning and Community Right to Know. The project was funded by the U. S. Environmental Protection Agency. LERS was used in other areas as

well, e.g., in the medical field to assess preterm labor risk for pregnant women and to compare the effects of warming devices for postoperative patients. Currently used traditional methods to assess preterm labor risk have positive predictive value (the ratio of all true positives to the sum of all true positives and false positives) between 17 and 38%, while the expert systems with the rule sets induced by LERS have positive predictive value between 59 and 93%. Moreover, LERS was successfully applied to diagnosis of melanoma, to prediction of behavior under mental retardation, analysis of animal models for prediction of self-injurious behavior, global warming, natural language and data transmission.

References

1. Chan C-C, Grzymala-Busse JW (1994) On the two local inductive algorithms: PRISM and LEM2. *Foundations of Computing and Decision Sciences* 19: 185–203. [232](#)
2. Greco S, Matarazzo B, Slowinski R (2000) Dealing with missing data in rough set analysis of multi-attribute and multi-criteria decision problems. In Zanakis SH, Doukidis G, Zopounidis Z (eds.) *Decision Making: Recent developments and Worldwide Applications*, Kluwer Academic Publishers, Boston Dordrecht, London, 295–316. [237](#)
3. Grzymala-Busse JW (1988). Knowledge acquisition under uncertainty—A rough set approach. *Journal of Intelligent & Robotic Systems* 1: 3–16. [223](#)
4. Grzymala-Busse JW (1991a) Managing Uncertainty in Expert Systems, Kluwer Acad. Publ., Boston/ Dordrecht London vol. 143 of the Kluwer International Series in Engineering and Computer Science. [228](#)
5. Grzymala-Busse JW (1991b). On the unknown attribute values in learning from examples. Proc. of the ISMIS-91, 6th International Symposium on Methodologies for Intelligent Systems, Charlotte, North Carolina, October 16–19, 1991, 368–377, *Lecture Notes in Artificial Intelligence*, vol. 542, Springer-Verlag, Berlin, Heidelberg, New York. [236](#)
6. Grzymala-Busse JW (1992) LERS—A system for learning from examples based on rough sets. In Slowinski R (ed.) *Intelligent Decision Support. Handbook of Applications and Advances of the Rough Sets Theory*. Kluwer, 3–18. [231](#), [232](#), [235](#)
7. Grzymala-Busse JW (1997) A new version of the rule induction system LERS. *Fundamenta Informaticae* 31: 27–39. [235](#)
8. Grzymala-Busse JW (2002) MLEM2: A new algorithm for rule induction from imperfect data. Proceedings of the 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU 2002, Annecy, France, 243–250. [235](#)
9. Grzymala-Busse JW (2003). Rough set strategies to data with missing attribute values. Proceedings of the Workshop on Foundations and New Directions in Data Mining, associated with the third IEEE International Conference on Data Mining, Melbourne, FL, 56–63. [236](#), [237](#), [239](#)
10. Grzymala-Busse JW (2004a) Characteristic relations for incomplete data: A generalization of the indiscernibility relation. Proceedings of the RSCTC'2004, the

- Fourth International Conference on Rough Sets and Current Trends in Computing, Uppsala, Sweden., Lecture Notes in Artificial Intelligence 3066, Springer-Verlag 2004, 244–253. [237](#)
11. Grzymala-Busse JW (2004b). Rough set approach to incomplete data. Proceedings of the ICAISC'2004, the Seventh International Conference on Artificial Intelligence and Soft Computing, Zakopane, Poland. Lecture Notes in Artificial Intelligence 3070, Springer-Verlag 2004, 50–55. [237](#)
 12. Grzymala-Busse JW (2004c). Data with missing attribute values: Generalization of indiscernibility relation and rule induction. *Transactions on Rough Sets*, Lecture Notes in Computer Science Journal Subline, Springer-Verlag, 1: 78–95. [233](#), [237](#)
 13. Grzymala-Busse JW, Hu M (2001). A comparison of several approaches to missing attribute values in data mining. Proceedings of the Second International Conference, RSCTC'2000, Banff, Canada, Revised Papers. Lecture Notes in Artificial Intelligence, 2005, Subseries of Lecture Notes in Computer Science, Springer Verlag, 2001, 378–385. [236](#)
 14. Grzymala-Busse JW, Siddhaye S (2004). Rough set approaches to rule induction from incomplete data. Proceedings of the IPMU'2004, the 10th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Perugia, Italy, vol. 2, 923–930. [237](#)
 15. Grzymala-Busse JW, AY Wang AY (1997) Modified algorithms LEM1 and LEM2 for rule induction from data with missing attribute values. Proc. of the Fifth International Workshop on Rough Sets and Soft Computing (RSSC'97) at the Third Joint Conference on Information Sciences (JCIS'97), Research Triangle Park, NC, 69–72. [236](#)
 16. Kryszkiewicz M (1995) Rough set approach to incomplete information systems. Proceedings of the Second Annual Joint Conference on Information Sciences, Wrightsville Beach, NC, 194–197. [236](#), [237](#), [239](#)
 17. Kryszkiewicz M (1999) Rules in incomplete information systems. *Information Sciences* 113: 271–292. [236](#), [237](#), [239](#)
 18. Lin TY (1989) Neighborhood systems and approximation in database and knowledge base systems. Fourth International Symposium on Methodologies of Intelligent Systems (Poster Sessions), Charlotte, North Carolina, 75–86.
 19. Michalski RS, Mozetic I, Hong J, Lavrac N (1986) The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. Proc. of the Nat. Conf. on AI, 1041–1045. [235](#)
 20. Pawlak Z (1982) Rough Sets. *Int. J. of Computer and Information Sciences* 11: 341–356. [223](#)
 21. Pawlak Z (1991) *Rough Sets. Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, Boston Dordrecht, London. [228](#), [233](#), [234](#)
 22. Pawlak Z, Grzymala-Busse JW, Slowinski R, Ziarko W (1995) Rough Sets. *Communications of the ACM* 38: 89–95. [233](#)
 23. Quinlan JR (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, CA. [235](#)
 24. Slowinski R, Vanderpooten D (2000). A generalized definition of rough approximations based on similarity. *IEEE Transactions on Knowledge and Data Engineering* 12: 331–336. [237](#)
 25. Stefanowski J (2001) *Algorithms of Decision Rule Induction in Data Mining*. Poznan University of Technology Press, Poznan, Poland. [236](#), [237](#), [239](#)

26. Stefanowski J Tsoukias A (1999) On the extension of rough sets under incomplete information. Proceedings of the 7th International Workshop on New Directions in Rough Sets, Data Mining, and Granular-Soft Computing, RSFDGrC'1999, Yamaguchi, Japan, 73–81. [236](#), [237](#), [239](#)
27. Stefanowski J Tsoukias A (2001) Incomplete information tables and rough classification. *Computational Intelligence* 17: 545–566. [236](#), [237](#), [239](#)
28. Yao YY (1996) Two views of the theory of rough sets in finite universes. *International J. of Approximate Reasoning* 15: 291–317. [237](#)
29. Yao YY (1998) Relational interpretations of neighborhood operators and rough set approximation operators. *Information Sciences* 111: 239–259. [237](#)
30. Yao YY (2003) On the generalizing rough set theory. Proc. of the 9th Int. Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing (RSFDGrC'2003), Chongqing, China, 44–51. [237](#)

Bioinformatics with Evolutionary Computation

D. Howard

1 Introduction

This chapter makes the presumption that it is more important to understand the domain of the problem of interest and to pursue the best achievable definition of the problem, than it is to apply our powerful algorithms in an attempt to solve it. Even with evolutionary computation, a serious attempt at specifying a definition of the problem must come first (the homework), before any attempt to apply the computational intelligence algorithm. This is a more rewarding and easier approach.

Computational Intelligence is at the point where different techniques are being combined to take advantage of their collective benefits [1]. However, a computational intelligence algorithm is only as useful as its appropriateness to the problem that it is trying to solve. One must, therefore, avoid applying such powerful computational methods to Bioinformatics problems without prior knowledge of the needs of Biology. Algorithms must address the needs of the biologists that present these problems.

The purpose of this chapter is to introduce the subject of Bioinformatics to the Computational Intelligence community. It may be used as a source of reference, or discussion, for the role of Computational Intelligence (specifically evolutionary computation) in Bioinformatics. Information concerning Biology that is not specifically cited in this chapter will have been gathered from a number of different sources of general knowledge (and on-line dictionaries) too large to cite specifically. However, three sources have contributed significantly to this review: [2, 3] and [4].

2 Essential Cell Biology

2.1 History of Life

Analysis of the most highly conserved genes and of differences between cells has revealed a classification of all present day life forms into three taxa:

Archaea, Bacteria, and Eukarya. The eukaryotic cell contains a nucleus and several organelles. A set of hypotheses for the origin of the nucleus are centred on the belief that the eukaryotic cell is a fusion of an archaeon with a bacterium, but these were refuted by more recent studies [5] in favour of the hypothesis that it came about from the endosymbiosis of three cells: an archaeon, a bacterium, and a third cell. By studying the proteins of extremely early diverging eukaryotes, Eukaryotic Signature Proteins (ESP), eukaryotic origins have been traced to test hypotheses of eukaryotic origins. They conclude that the third cell (called a Chronocyte) was an RNA-based cell that branched away from a pre-prokaryotic stage in cellular evolution before the Archaea and Bacteria. Chronocyte had a cytoskeleton that enabled it to engulf prokaryotic cells. It had a complex internal membrane system where lipids and proteins were synthesized. It also had a complex internal signalling system involving calcium ions, calmodulin, inositol phosphates, ubiquitin, cyclin and GTP binding proteins. The chronocytic cell contributed its cytoskeleton, ER, Golgi apparatus and major intracellular control systems to the resulting eukaryotic cell, and the nucleus was formed when it engulfed archaeon and bacterium [5].

2.2 Technology of Life

The living cell is a compartment that maintains a certain order within it. Although the net effect of the cell is to increase entropy, it must obtain energy and expend it methodically to attain its goal of maintaining low entropy within the cell wall as compared to its surroundings. Although a handful of eukaryotes are unicellular, they usually congregate together as organisms to assist with this requirement. Such a community of cells adopts various organizations with associated emergent functions (motion, communication, vision, digestion, etc.). Yet underpinning all of Life are two cellular technologies:

- Machinery exists within each cell to build proteins and RNA (the three dimensional molecular material that makes up the organism and carries out its functions).
- DNA implements a linear code that determines how, where, and when to build these three dimensional molecules.

Both organism and the cell need to acquire additional energy to reproduce, and since it occurs it must have justification. Damage will take place during the cell's life and so it might conceivably be argued that being over defensive with molecular repair (as a result of damage caused by chemicals within the cell or by radiation over long periods of time) would not allow a cell to change, and that change is necessary to adapt to a changing world. Therefore, a "solution" would be to create copies of the organism that maintain a separate existence. This increased redundancy in numbers and the variety (in the progeny) would ensure the survival of the life-form. Moreover, the cellular compartment is a complex system, and the cell or organism would find it advantageous to create

the child as a copy compartment partly because organized compartments are sources of protection – to protect complex molecules.

The organism's blue print is stored in nuclear DNA (as chromosomes) and the same DNA message is present in every cell of the organism (though it is absent in some cells such as our body's blood cells). RNA molecules repeatedly copy those parts of the blue print known as the genes and process that information as input to an organelle known as the ribosome that translates the RNA message to assemble a corresponding protein molecule. The translation code has an in-built redundancy because it uses the codon, a 3-letter message that admits 64 possibilities and maps these to the 20 different amino acid types that constitute such manufactured proteins. It is interesting to note that surveying all life forms reveals a “standard” translation code that is predominant but not universal. This indicates that the translation code itself has evolved to be more robust and impervious to mistakes (see [6, 7] and [8]).

DNA occurs in cells as a double stranded helix, but both proteins and RNA can fold and rearrange into a variety of shapes. Different cellular functions are derived from three-dimensional shape. The manufactured proteins are responsible for the structures and functions in the cell and organism. In addition, certain RNA molecules also have important structural and catalytic functions, for example in the ribosome, where the polypeptide formation from the aminoacylated tRNA (transfer RNA) to the peptide chain or transesterification is mediated by its rRNA (ribosomal RNA) [9].

What justifies the presence of both RNA and DNA in these technologies when RNA and DNA are so similar? What characterizes our cells is that DNA carries the blue print and is arranged in a double helical structure while RNA strands are involved in repetitive intron splicing or in translation regulation. A double helix is the formation of base pairing interactions with a second strand (or same-strand in the case of hairpin formation), which involves hydrogen-bond formation between opposing strands, stacking of base pairs on top of one another, and reducing conformational freedom of the phosphodiester backbone. However, RNA has the same capacity to form a double helix. The precise shape of this double helix, though, will be different because while both RNA and DNA have a sugar backbone and four types of bases, the sugar backbone of RNA contains an additional OH group that confers it with different folding properties, and the detailed shape of the double-helix is A-form for RNA and predominantly B-form for DNA. Also, DNA can form a double helix with RNA. However, RNA and DNA only share three of their four bases, with the difference that DNA uses thymine (T) when RNA uses uracil (U). The latter, U, is simpler and less costly to assemble compared to the former, T, and RNA is used a lot, so one could justify the use of RNA. Moreover, using T rather than U in the alphabet should lead to a more efficient or quicker repair (see Fig. 1) justifying DNA as the long-term carrier of the information. Therefore, it might conceivably be argued that today's cell involves DNA and RNA for different purposes in some optimal sense.

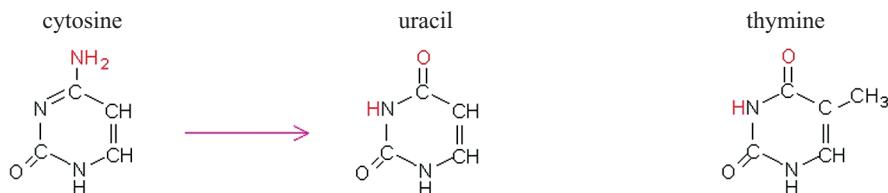


Fig. 1. *Left:* Nitrous acid performs an oxidative deamination that converts cytosine into uracil (DNA can detect it with ease because uracil is not part of its alphabet). *Right:* chemical formula for thymine

2.3 Complexity of Life

A human being is a cell community with enormous numbers of cells that come in approximately 200 different cell types. How are so many different cell types possible, each with an identical copy of the unique linear message (a unique DNA message characterizes this cell community)? Genes are blue prints for proteins and there exist about 32,000 in the human message and other organisms can have tens of thousands more. Repressor and enhancer proteins act to block and to promote expression of particular genes at different times and in different locations of the body resulting in the different cell types.

The information stored in DNA results in largely predictable outcomes but under highly non-linear and protracted developments, and the organism takes certain precautions. For instance, blood cells are only created in the bone marrow from stem cells. Blood cells are not allowed to divide, and this remains true even when they contain a nuclear DNA, as is the case with the blood cells of birds. Arguably, the organism has in-built controls to ensure its largely predictable outcomes.

Millions of molecular events that transpire such as cell division and intron splicing may introduce small errors (mutations) but these are few. There is a lot of DNA to absorb such changes and repair mechanisms in place, such that the great majority of us will survive these rare errors. Cancer (an accumulation of several errors at one or more cells) does eventually afflict one third of us.

A protein must assume a stable and precisely ordered conformation to perform its biological function properly. Within the cytoplasm, special proteins called chaperones assist protein folding. However, all of the information that is needed to fold the protein is with the amino acid chain. This can be confirmed in the laboratory by means of a test tube experiment that uses certain solvents that break the weak non-covalent interactions that hold the protein's shape. By this means, a very pure protein can be unfolded to become a flexible polypeptide chain. However, it will refold spontaneously when the solvent is removed.

Proteins have different lengths and some get to be several thousand amino acids long. The protein building block is a very powerful functional unit both because of the uniqueness, reliability and speed of the protein fold, and

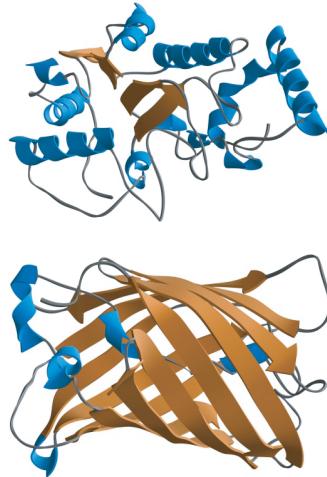


Fig. 2. *Top:* This is the human uracil-DNA glycosylase protein, which has 223 amino acids and 1993 atoms, has a 3-layer sandwich architecture with both α helices and β sheets. *Bottom:* This is the green fluorescent protein found in jellyfish (*Aequorea victoria*), which has 237 amino acids and 1863 atoms, is barrel-like, and mainly contains β sheets with some alpha helices. These images were produced using IBM's Prototype Protein Viewer (PPV) for PCs [12]

because of the potential availability of a formidable variety of 3D shapes in terms of proteins and ensemble of proteins. A lexicon of tens of thousands of different proteins is available to life, and this can be produced repeatedly. The cell can also manipulate these proteins to change their activity by various means such as altering their 3D shape.

Figure 2 illustrates the 3D conformations of two simple proteins (they can be found in the protein data bank [98] by searching for “1AKZ” and “1eme” respectively). Note that these proteins contain a prevalence of two structures: α helices and β sheets. These structures were deduced in the 1950s (e.g. [10]) at a time when scientists became excited about the role of proteins in life. Helices, sheets and turns are the classical secondary structures. Turns were discovered in 1968 [11], are found primarily on the protein surface, and serve to reverse the direction of the polypeptide chain. Turns include many polar and charged amino acids.

It is postulated that the biological system behaves by analogy to the continuous execution of a distributed, decentralized, and parallel computer program (others have made this connection, e.g. [13]). Simpler analogies have their appeal, e.g. the regime of decision and control under both centralized and devolved management and involving both sequential and parallel steps [14]. Yet, as discussed later in this chapter, it is apparent that the degree of metabolic and regulatory complexity is high and intricate. There even exist delay mechanisms such as DNA polymerase pausing, whereby gene

expression is interrupted and then reactivated (polymerase stalling is involved in the process of replication slippage, which underlies many instances of genome rearrangements [15]).

3 Relevance of Evolutionary Computation

3.1 Questions with Many Answers

As we start to understand the workings of the cells and the cell community that is the organism, techniques of Artificial Intelligence that can discover representative and non-trivial computer programs will gain in popularity for simulation and understanding of these highly non-linear systems. The family of Genetic Programming methods are in this category [16, 17, 18, 19, 20]. Genetic Programming has proved time and again that it can scale up to high dimensional large data sets – it can handle many variables and discover their interrelation and information value.

There are also new and potentially useful models. For example: [21] offers a simplified biological model of evolutionary computation that considers the importance of regulatory networks and incorporates developmental approaches more explicitly. In [22] the analogy between chemical pathways and electrical circuits is exploited. The development or evolution of simple computer programs to obtain complex behaviour [23] may call for the wider practice of separating the genotype (the blue print) from the phenotype in evolutionary computation. For example, a simple genotype is evolved and its fitness is the result of a possibly quite complex pattern that ensues from its expression.

Fuzzy Sets and Fuzzy Logic must find roles in this modelling because of the sheer number of chemical pathways and the ensuing system complexity. Although Biologists study quantitative measures such as concentrations of metabolites, meaning accrues from the context and the timing in which events occur. For this reason, linguistic labels defined more or less precisely by the relevance of surrounding facts [24], surely must have a role to play to efficiently describe (and simulate) the process of decision and control of cell and organism. The literature is already concerned with “masks” to decide upon information significance [25] (what must not be reported). Moreover, Fuzzy Cognitive Maps capture the intuitions of biologists, providing a modelling framework for assessing large amounts of information on pathways [26].

Although statistics is the favorite tool of the Biologist, it will not suffice to gain an incremental understanding of the system. The non-linearity of the system calls for discovery of clues. Informative features can be discovered by “searching large computational spaces in an efficient manner without any prior assumption or simplification” (page 18 of [27]).

Moreover, in the process of scientific discovery findings need not be statistically significant to become immediately useful. What the Biologist requires are theories that can be examined further by experiment. Experimentation

that is heuristically driven has an important role to play because genes are coupled, and knocking out single genes cannot always provide sufficient knowledge about the nature of the system (about its behaviour).

This last observation should motivate us to develop new methods of Computational Intelligence. For example, any technique that can produce a “solution fan” would be useful to Bioinformatics. What does “solution fan” mean? When faced with under-determination presented in the data the question that is asked will admit many equally valid and perhaps also equally plausible explanations, and their number could be extremely large. It then befits to discover a handful of equally valid explanations that are quite different from one another in terms of their component “tactics” (the emergent features).

Finding the “solution fan” must not be confused with finding solutions in a Pareto front (as in multi-objective optimization). In the scenario just described, the Biologist hands us a problem to be reversed engineered. He or she does not give us a utility value to be maximized along different dimensions that would drive a Pareto front. Instead, the Biologist may tell us what we might expect to find and things to watch out for. Our problem is to find several equivalent explanations to the Biologist’s question, several results that answer the question by different means (different explanations). The main difficulty being that while the Biologist would welcome surprise in the explanations that we provide (lateral thinking) we need foreknowledge about what is important to the Biologist so as to define the distance measure by which to return a few representative solutions from the possibly hundreds of working explanations that our intelligent model might generate.

The appendix at the end of this chapter explains all of this more clearly by analogy to an illustrative toy problem (see Appendix A) purposely designed and solved by this author to illustrate this idea.

3.2 Population Based Methods as Candidate Technology

In Fig. 3, a Biologist needs to get to the white circle from the grey circle. We carry out an exhaustive search to find the “solution fan” and this gives us 100 solutions to this problem – a large but a finite and countable number. Let us say that 99 of these solutions are variations on the path of the solid arrows in Fig. 3, where the path may take one small detour (an excursion away from the straight path). Each of the 99 solutions is a variation on this theme. The hundredth solution looks different, in the distance sense, and is denoted by the other non-solid arrows that go to the white circle up and sideways.

What should we report back to the Biologist? First, note that discovery of 99 answers that can be distinguished from 1 answer by using a linguistic label in no way guarantees that the real world follows the popular answer. We cannot conclude that it is more likely. The real world may just as well correspond to the hundredth solution that looks rather different to us. All of these tours have passed the test of being solutions, and we are required to present a few plausible models to guide further experimentation by our

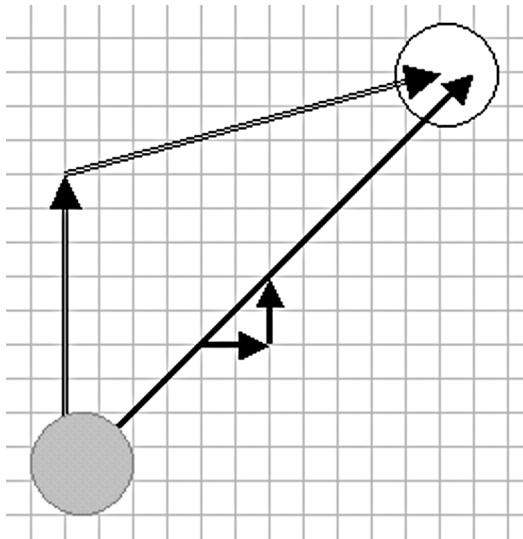


Fig. 3. Solution Fan: getting from *grey* to *white* (see text)

Biologist. However, we do not want to give him 100 models. How to choose the subset? The choice of measurement and distance that gave us both clusterings (99 solutions that are different from 1 solution) and the linguistic label chosen to describe these clusterings are rather subjective and arbitrary.

Therefore, we need some guidance from the Biologist about what is important to his or her work. What is the current view of reality?; what things may be implausible? and so on. However, we must not allow these ideas to pollute our method of finding the solution. We do not want to tell the machine how to solve the problem because we will prevent emergence of an innovative solution. The idea is to straddle a middle ground. This is not a constrained multi-objective optimization scenario because what the Biologist provides is not crisp. The Biologist offers things to watch out for, linguistic descriptions of what we should or should not expect.

At any rate one thing is for certain. The application of Computational Intelligence to Bioinformatics generally requires that we make a concerted effort to produce a “solution fan” and to decide on some way of reporting back information that may be interesting. Finally, it is possible that Occam’s Razor may assist with our choice among solutions. It is usually the case that the simpler solutions are the ones that are more likely to exist in the real world. This, however, is dangerous territory because is predicated by the definition of simplicity, by a choice of distance measure (what are the important differences), and how to look at the solutions.

What is a feature? how to select and how to present the handful of most informative and representative solutions? These are worthwhile questions. Concern with the validity and plausibility of models in Bioinformatics

research [28] is equally shared by weather and climate modellers and by other scientists.

Evolutionary computation is an ideal technology for discovering the “solution fan” since it maintains and works with a population of candidate explanations. And although research for maintaining diversity of search is maturing [29, 30, 31], to the knowledge of this author, it has not yet been posited for the explicit objective of discovering the “solution fan”. For instance, it would need to compare phenotypes in many ways to measure distance between solutions in feature space. It could in this way distinguish various approaches in the generated solutions to select what solutions to report back to the Biologist.

4 Cell Biology and Bioinformatics Research Fields

4.1 New Fields of Research

In recent years, successful large scale DNA sequencing, the invention of the polymerase chain reaction (PCR) and improvements in molecular identification techniques (microarrays, NMR, etc.) gave birth to a number of research fields. The number of organisms with completely sequenced genomes has grown considerably in the last few years. For example it exceeded 100 in 2003, plus an additional 600 since then.

The new research fields both generate and work on this data and concern themselves with DNA analysis, protein structure, protein function, gene expression, the chemical reactions responsible for metabolism and regulation in cells. Moreover, numerous research laboratories throughout the world are active in research fields that study cell division and cell communication in model organisms (organisms that are a simplification of ourselves with a simpler anatomy to our own, or that have similar genomes to our own and fast reproduction).

Names of these new fields of study use the suffix -omics (e.g. gen-omics, prote-omics). Today there exist over one hundred words in use with this suffix, but the etymology of this suffix is complicated. Starting from sarcoma, the medical profession has co-opted the -oma suffix (from New Latin and Greek) to signify an overgrowth or neoplasm of any specified tissue. Words such as biome, rhizome, phyllome, thallome, tracheome share in common the suffix *ome* signifying the collectivity of the units in the stem of the word. The word genome was invented by Hans Winkler, apparently as an irregular formation from *gen* and *some* (from chromo-some). The term “genomics” arose during a meeting at a bar in Bethesda, Maryland, and was later introduced by McKusick and Ruddle to name a new genome-oriented scientific journal [32].

A recent field of study goes by the name: Metabolomics. This uses *meta* the Greek for change (adjacent or after) and *ballein* from German meaning to throw a ball (or by abstraction to change the chemistry of a molecule by moving it through a pathway) and ends with the suffix -omics. Yet another

field of study, Metabonomics, again is derived from the Greek word *meta* but uses *nomos* which is Greek for rules or rule set (as in the word Economics).

The origin of the names of these fields is important because scientists who coined these names were trying to describe the essence of their research activity. What, for example, is Metabonomics hoping to describe from metabolic measurements? and what limitations are to be placed on the measurements and the ensuing interpretation? The word Metabonomics would seem to indicate a view of organism as complex system and the aim to understand it by discovering the rules governing changes to the organism.

It is useful to provide a description of some of these fields of research activity because such terms appear with frequency in the Bioinformatics literature:

- Genomics
- Transcriptomics
- Proteomics
- Metabolomics
- Metabonomics

Genomics is the study of the entire genome (rather than one particular gene) of the organism, and Proteomics is the study of the entire protein content of the organism (study of the proteome). Transcriptomics describes regulation of gene expression, the key process for adaptation to changes in environmental conditions. The transcriptome is the subset of genes that are transcribed. It is the dynamic link between the genome, the proteome and the cellular phenotype. Transcriptomics studies the regulation of the transcriptome.

The fields of Metabonomics and of Metabolomics were born at about the same time in London, England and in Manchester, England respectively. Definition of the former has not changed whereas definition of the latter has changed somewhat over the years. These very similar terms have arisen at about the same time in different areas of bioscience research, mainly animal biochemistry and microbial/plant biochemistry respectively. Although both involve the multi-parametric measurement of metabolites they are not philosophically identical. Metabonomics deals with integrated, multicellular, biological systems including communicating extracellular environments and Metabolomics deals with simple cell systems and, at least in terms of published data, mainly intracellular metabolite concentrations.

The term Metabonomics [33] was conceived in 1996, and first appeared in the research literature in 1999 [34]. It considers mammalian system function and dysfunction and how biofluid metabolic profiles may be used to measure system failure [35]. It measures multiparametric metabolic responses of complex systems to pathophysiological intervention and or genetic modification. Important scientific and philosophical differences give good reasons not to confuse these two fields. While Metabolomics has a very analytical definition, Metabonomics is a reactive science that looks at the way a total complex system responds to a deliberate intervention (systems biology). Metabonomics

accommodates the idea that many different cell types are doing different things at the same time and yet coordinate together. Moreover, Metabonomics accommodates the important notion that metabolic contributions to a complex system can come from diverse chemical and extragenomic sources – especially gut bugs or even parasites [36]. This notion is important because most of the low molecular weight species in human biofluids (probably 95% of them) are not under direct human genome regulation.

Metabolomics originates in microbial and plant biochemistry and evolved independently from Metabonomics as a field of study. The term Metabolomics first appeared in the literature in 1998 where it was described as the measurement of the metabolome – the total content of low molecular weight metabolites in a cell. The meaning of Metabolomics has evolved. The most popular definition today [37] is the measurement of the totality of low molecular weight metabolites in a biological system. Metabolomics researchers are particularly interested in the measurements leading to better understanding of cellular metabolic and regulatory pathways.

4.2 Evolutionary Computation for Metabolomics

The Metabolomics field has used both Genetic Programming and Genetic Algorithms. Both techniques are used for classification and regression. The focus of this activity is the calibration of spectral, NMR and other equipment for experimental measurement of proteins and metabolites. Supervised and unsupervised methods aim to recognize the levels and identity of proteins, separation of molecules in mixtures, and the interpretation of spectra. Early work was carried out by Lucasius [38]. Rowland offers an excellent review [39].

4.3 Evolutionary Computation for Metabolic Pathways

A metabolic pathway is a series of chemical reactions occurring within a cell, catalyzed by enzymes, and resulting in either the formation of a metabolic product to be used or stored by the cell (metabolic sink), or the initiation of another metabolic pathway (then called a flux generating step).

Genetic Programming offers powerful technology to infer pathways from data (reverse engineer chemical reactions) and two methods have been demonstrated [22, 40]. The first maps the biologist's notation to the Genetic Programming representation using sophisticated function and terminal sets, and the second method converts the pathway to an analog electrical circuit and makes use of those powerful representations developed with Genetic Programming for discovering novel electrical circuits [41]. Other researchers have tackled this problem with Petri Nets and Genetic Programming [42].

However much hierarchical and auto-recursive power such techniques have for reverse engineering the pathways, they must grapple with the realities of the complex regulatory system. It would be naive to think that observation of metabolite levels on its own can obtain the structure of the pathways

directly [46]. Many years ago research pioneers Kaczer and Burns [45] put forward the view that there exists a web of regulation rather than straight forward regulation. Control is shared by all enzymes of a pathway. They advanced the notion that too much of a substrate in one place can have an effect in another completely different part of the system.

Therefore, even should these metabolite levels be quantitative absolute concentrations (which they are not) it would be impossible to obtain the structure of the pathways with this data alone. Recently, Fiehn's group [47, 48], devised a novel extraction protocol to sequentially extract metabolites, proteins and RNA from the same sample, providing a convenient procedure for the analysis of replicates as well as exploiting the inherent biological variation of independent samples for multivariate data analysis. They merged a subset of the most abundant proteins and metabolites from replicate analysis of different *Arabidopsis* accessions to form an integrative dataset allowing both classification of different genotypes and the unbiased analysis of the hierarchical organization of proteins and metabolites within a real biochemical network. This represents a superposition of the stoichiometric network and global network regulation. A top-down approach is being followed by Mendes [46] at the Virginia Bioinformatics Institute. This starts with global data (metabolomics and microarrays data) and ends up with a "mechanistic" model. The observational data are a convoluted reflection of the network structure and of the network dynamics. Each can mask the other and they need to be disentangled.

Why this complexity? The cell can do a similar job to the situation in the chemistry laboratory but only at much smaller temperature and pressure. The cell also tries to operate at minimum energy expenditure and to avoid bi-products that might kill it. It is the instant termination of cell pathways, which die, what stops a reaction going in this direction. Therefore, the interpretation that must be superimposed for modelling metabolic pathways is that of a complex regulatory system, which requires sophisticated modelling of constraints. Things are not simple (for example see [43]). Other factors such as metabolic stress (poor supply) permit some enzymes to have more control than they otherwise would. A useful illustration is the interaction between the glyoxylate cycle (see p. 668 of [44]) and the TCA or Krebs cycle. A pathway acts as a control point to skip metabolic steps in the Krebs cycle, e.g. to produce heat in warm blooded animals to keep them warm under harsh temperature conditions. Hans Krebs never spotted the glyoxylate cycle because it is not dominant.

At UC San Diego, Palsson has tried to cope with this problem by flux balance analysis and the technique that he developed called *extreme pathway analysis* [40, 50]. The idea is to look at what are the limits of what is possible so as to bound the pathways solution space. It aims to identify those regions of the search space that are not available to the organism. The problem of under determined data (already encountered in Sect. 3) is also present in the problem of reverse engineering of pathways. However, what is needed are

techniques that map out the structure of the solution space (as Palsson is doing) because the number of solutions could be very large.

Finally, two related fields: *Epistasis* and *Quantitative Genetic Theory* are of interest when considering the significance of modelling metabolic pathways, regulatory networks and gene expression:

- Epistasis is the masking of the phenotypic effect of alleles at one gene by alleles of another gene. An *allele* is an alternative form of a gene, one of the different forms of a gene that can exist at a single locus. An epistatic gene suppresses the effect of a gene at another locus, i.e. the influence of one locus on the expression of genetic variation at another locus [51, 52]. Epistasis modulates the effects of future mutations and may thus have a long-term effect on the patterns of natural variation [53].
- Quantitative Genetic Theory concerns itself with choosing how to measure the effect of the genotype on the phenotype or how to assign phenotypes to haplotypes. The term *haplotype* stands for a set of closely linked genetic markers present on one chromosome which tend to be inherited together (not easily separable by recombination). Two measures exist for this: *average excess* and *average effect*. The former is the easiest to calculate and has more direct biological meaning. Devised by R.A. Fisher in 1918 [54], average excess of a haplotype is the average phenotypic measurements made of bearers of that haplotype minus the overall population mean [55]. The average effect is the least squares regression coefficient that defines the linear relationship between the phenotype and the number of copies of each haplotype (zero, one or two) borne by an individual. Average effects can be calculated from average excesses and genotype frequencies.

These notions are relevant because the cell has many mechanisms available to slow down or to stop a pathway. For example, a molecule can dock with a protein, or it can act to change the protein's shape. However, we are not all the same. Some of us have versions of the gene (or one malfunctioning gene and one good gene in the pair) able to produce lower or higher quantities of an enzyme. More or less amounts of enzymes will affect the way in which our organisms manage regulation.

4.4 Evolutionary Computation and Gene Expression

The molecular mechanisms for expressing a gene are quite different in prokaryotes even at the level of how polymerase (pol-II) works. Pol-II is the molecule that plays a very active part in separating the strands of DNA to start transcription. In prokaryotes, pol-II is assisted by the sigma factors, but in eukaryotes pol-II is recruited to the scene by a number of transcription factors. These molecules and all of their interactions have been illustrated beautifully by Tsonis [56].

A handful of these eukaryotic transcription factors were identified *in vitro* but hundreds or thousands of these factors may participate *in vivo* to enhance

or to inhibit gene expression. They may act from thousands of bases upstream of the gene, close and immediately upstream of the gene, inside of the gene, and even somewhat downstream of the gene. Nuclear chromosomes are surrounded by chromatin but wrapped up inside of the nucleus (see illustration in [57]) and this may be why these factors can act from such base pair distances and from different directions.

The *promoter* region is a region upstream of the start of transcription site of the gene. This is where pol-II and some of the transcription factors attach to the strand of DNA to commence transcription of the gene. There exists a *eukaryotic promoter database* [58] with information about promoter regions in DNA. This can be used to learn to detect promoters [59, 60], or perhaps consider learning to detect the transcription start site (see discussion in [61] in prokaryotes).

The objective of promoter prediction is not to be definitive as to what is a promoter region but to down select to candidate regions to offer the Biologist putative promoter regions. Experimentation must of necessity be combined with promoter prediction tools to obtain any benefit [62].

Recent attempts at predicting eukaryotic promoters in *Drosophila* assisted with Genetic Algorithms [63], and in human DNA directly with evolutionary computation [64], both recognize the importance of “context” in pattern matching. The method in [64] uses a modified version of the GP-Automata [65] to enable the scheme to take large jumps about the genome. The idea of this facility is to imbue the scheme with the potential for discovery of the *cis*-acting sites which cause genes to become co-expressed by analysis of genetic microarray results. Several genes seem to be transcribed coordinately. For example, the different members of a storage protein or photosynthetic protein family are expressed at the same time in development. These genes have sequence modules in common that control the coordinate regulation. These modules are called response elements. These elements are a class of *cis*-acting elements.

Microarrays (gene chips) facilitate the “whole picture” view of gene expression because they can measure the expression of hundreds of genes concurrently at a moment in time. Microarrays are a source of the “data deluge” in Bioinformatics. An array is an orderly arrangement for matching known and unknown DNA samples by hybridization (A-T and G-C for DNA; A-U and G-C for RNA), and this is their underlining principle. The sample spot sizes in microarray are typically less than 200 microns in diameter and a microarray usually contains thousands of spots. Microarrays require specialized robotics and imaging equipment.

Microarrays obtain expression as snapshots in time and can be used to monitor the reaction of an organism to a drug, or a plant to environmental changes, and so on. Also, they can provide data for characterizing different types of cancer (classification). Microarrays use colours that enable studies in differential expression, or the gene expression comparison between two experiments.

Microarray experimentation can suffer two drawbacks (at the time of writing). If laboratory practice and experimental technique are not highly skilled (not following the advice of the manufacturer or specialist group) then results may not repeat. Thus, it would be important to confirm that experiments are reproducible. Sometimes, the under-determination presented in the data is of concern because there could exist far fewer experimental training samples for classification than the number of genes expressed in the microarray. Genetic Programming and evolutionary computation have produced a number of results for microarray problems: [66, 67, 68, 69, 70, 71].

4.5 Evolutionary Computation and Multiple Sequence Alignment

Comparison of sequences is the bread and butter activity in Bioinformatics. However, evolutionary computation does not have a commanding lead in this important topic. Dynamic Programming and Hidden Markov Models have dominated this topic. However, a number of recent papers apply evolutionary computation to multiple sequence alignment (e.g. [72]).

4.6 Evolutionary Computation and Protein Folding

Protein folding is an area of much activity and this section aims to point the reader to some of its developments. It is one of the fundamental unsolved problems in molecular biology. It is also amongst the most important of its challenges. Solving the protein-folding problem would have some impact on understanding life and on curing disease. However, the *ab initio* protein-folding problem, or the prediction of the 3D structure based on the sequence of amino acids and using only a universal energy function and minimization strategy that can be applied to any protein, remains unsolved.

This seems odd because as was already discussed in Sect. 2.3, all of the information that is needed to fold the protein is with the amino acid chain¹. A protein assumes a stable and precisely ordered conformation every time and this enables it to perform its biological function properly. Large proteins can fold in seconds, while the *ab initio* protein folding takes the age of the universe to compute, but why is this problem so hard to solve?

Arguably, it is not a paradox as maintained in [73]. The observation has a very clear explanation. Folding follows a process with intermediate stages but more importantly it is a physical phenomenon. However, the objective in current protein folding research is either to predict the most likely resulting fold among all possible folds, or to simulate the process of folding the protein to arrive at the correct fold. Regardless of this choice, however, we are required to make another modelling choice. As in other fields (e.g. Computational Fluid Dynamics) two choices are available: modelling as a discrete particle

¹This is a simplification of *in vivo* folding for the purpose of this discussion.

simulation, or modelling as a continuum by means of specification of and numerical (or analytical) solution of a set of partial differential equations.

Considering the first modelling choice, the classical mechanics problem alternative, is also problematic. Unlike Computational Fluid Dynamics particle simulations (examples: [74, 75, 76]) where the particles simply collide with one another (as in the kinetic theory of gases), the amino acid “ball and stick” approach can be a victim of an exponential number of conformations to choose the lowest energy alternative. Each residue has a number of conformations and all conformation combinations must be explored for all residues. The number of possible folds grows exponentially with the number of residues and the problem quickly turns into a needle in a haystack problem or NP. Thus, a simulation of a protein with thousands of residues that folds in seconds becomes a search that no existing computer can cope with in eons of CPU time.

The problem with the second modelling choice is that the system of differential equations that would reflect the nature of protein-folding reality is exceedingly difficult (perhaps currently impossible) to write down. It is possible to imagine that a theoretical construction of a quantum wave-like model would be possible, and also that it might be possible to discretize it and to numerically solve it using local (compact) operators. With some luck the differential equations are well posed and no more difficult to numerically solve than solving complex problems in electrodynamics. Under those conditions an alternative method of solving this problem would become available. Application of the quantum mechanics approach to an ensemble of atoms is a problem in the research frontier. Thus, is important to appreciate that the excessive compute time required by the particle way (the first choice) has to do with the approach that is taken to model the problem because real world protein folding is a physical resolution.

A useful forum to meet the scientists involved in protein folding is CASP [77]. Some have used spare worldwide computer resources to fold proteins in a distributed computational fashion [78]. The NP nature of the search space makes Genetic Algorithms useful here and calculations depend on the energy measures that are derived from experimental data [79] and that account for hydrophobic and hydrophilic interactions (hydrophobic packing), amino acid size, hydrogen bonding, the electrostatic and Van der Waals interactions of surrounding atoms, solvent or environmental effects. Parallel computing is essential and solutions with Fast Parallel Messy Genetic Algorithms [80] have been attempted. Design of the fitness measure is not trivial [81].

By incorporating knowledge of other known protein structures, homology, and much information, certain proteins have been approximated to a good degree without prior knowledge of their structure. There is a lot of uncertainty and complexity in *ab initio* protein folding. Simulations that take advantage of knowledge gained from thousands of proteins of known 3D structure have more chance of success. Genetic Algorithms are important to these efforts (e.g. [82]).

Inverse folding also known as “threading” is an alternative to *ab initio* prediction. Both require exhaustive search or heuristics, but the options are far more constrained in the former as compared to the latter. Threading computer simulations require a scoring function to evaluate the inverse model. They input the sequence together with some model of the position of the core residues and allowable lengths of loops. The method considers: the similarity of the bases to the original at each position; the length and similarities of the loops and the pairwise interactions between bases at core positions. A pairwise interacting optimization function requires tabulating the possible substructures for every base assignment, not just the best matching prefix structures. The model of pairwise interactions turns the problem NP-complete and candidate for evolutionary computation.

Small manageable protein segments have been successfully folded and unfolded numerically by those simulating the folding pathway mechanisms at microsecond intervals in atomistic detail [83]. Their objective is to study the dynamics of proteins (a useful overview is given in [84]). Thus, they cope with small molecular ensembles rather than large proteins but have taken these small ensembles to a complete fold arriving very close to the measured 3D structure. They have made observations also about the unfolded state of proteins [85]. Their technique can investigate mis-folding of proteins, which is known to lead to important diseases, and other simulations which cannot be easily carried out experimentally. The *beta* “hairpin” (a substructure of the *beta* sheet secondary structure) is popular for these experiments. A hairpin is composed of two strands of antiparallel β -secondary structure connected by a tight turn as illustrated in Fig. 4. The approach is classical but because they follow the trajectories of protein folding the problem is not NP-complete. However, computational requirements are daunting for today’s computational power and have succeeded only by involving the computers of the world in a massive distributed computing exercise [86, 87]. Recently, attempts have been made to accelerate these remarkable distributed computations as they aim to tackle larger protein folding problems [88].

In 1961, at the university of Madras and without the benefit of present day computers, an Indian scientist called Ramachandran worked out the limiting distances of closest approach between various pairs on non-bonded atoms in a protein structure. These set limits on dihedral angles (also known as torsion angles or twist angles) and are used in protein folding simulations. His analysis excludes certain possibilities and the space-filling nature of side chains makes some conformations impossible. There is similarity between Ramachandran’s work and that of modern day Palsson (see Sect. 4.3) because both discovered forbidden parts of a search space. Recently, a group of researchers at Berkeley [91] is pursuing this type of approach. They are cataloguing the secondary structures of the proteins with known structure and hope to organize the building blocks of 3D protein structure from an evolutionary standpoint. This approach may help to predict protein structures. Ideas from “chemical linguistics” (or “cell language”, see [92]) might relate to this technology. The future

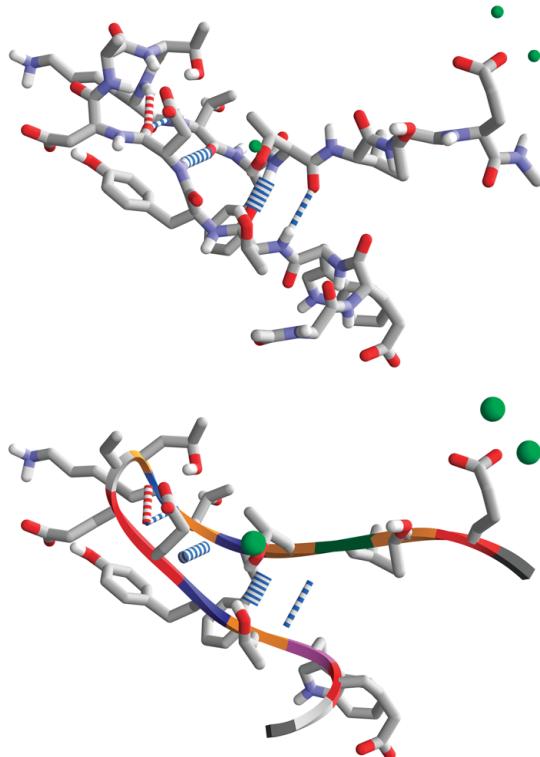


Fig. 4. Depiction of the β -hairpin from the c-terminus of protein G from Streptococcus, showing the actual hydrogen bonds that are responsible for the secondary structure. The green spheres in these views are counter-ions that are part of the simulation by the IBM team [89, 90] used to neutralize the net charge. The blue stripes are hydrogen bonds between the H and O on either side of the hairpin. Blue are “native” bonds and red are “non-native”. These images were produced using IBM’s Prototype Protein Viewer (PPV) for PCs [12]

of protein folding is probably modelling new structures based on relatives of known structure (homology modelling).

4.7 Evolutionary Computation and Molecular Docking

The way in which a protein folds is closely related to how drugs may bind to proteins and evolutionary computation has been successfully applied to this problem. In [93], Thomsen uses a simple evolutionary algorithm to solve the *docking problem*, which tries to identify the bound conformation of a small molecule to a protein of known structure. This is highly relevant to drug discovery research. The docking problem aims to find the most energetically favorable conformation of the molecule (the ligand) when bound to the protein, i.e. the smallest binding energy. It has similarity with the “protein folding

problem” because it is an energy minimization problem, and because there exists high complexity in having to search the space of all possible 3D conformations to determine the most energetically favorable solution.

In [93], the protein is assumed to be rigid during docking and only the ligand is represented as an individual to be evolved. It is an array of real valued numbers that encode: the ligand’s position, orientation and conformation as ligand coordinates for the ligand translation, four variables defining a quaternion specifying the ligand orientation (specifies an axis of rotation along with an angle of rotation about this axis), and one angle for each flexible torsion angle in the ligand. Evolutionary computation has addressed these types of molecular problems for some time [94, 95, 96].

4.8 Evolutionary Computation and Sequence Information

Sequence and function information about many thousands of proteins is publicly available on the Internet [97]. Although, information about the 3D conformation of proteins is also universally available [98] there are far fewer proteins listed in the latter (just over 20,000 from a broad variety of organisms). This is because the sequence is relatively easy to obtain by DNA sequencing but the protein structure is difficult and expensive to work out. Protein structures can be experimentally determined by crystallizing the protein and then using x-ray crystallography or NMR to find the position of the atoms, but this is a very difficult procedure. Widely available data promotes all sorts of research. For example, analysis of the frequency of a particular residue in different proteins has encouraged speculation about past evolutionary choices [99].

Genetic Programming search with astute use of function and architecture to process variable length sequences of amino acids and their hydrophobicity values has revealed how to detect transmembrane domain proteins. It has also predicted the likely location of proteins in the cell and the prediction of protein secondary structures [17, 18]. Genetic Programming devised solutions that could be examined to understand the principles of detection and of classification. Moreover, Genetic Programming exploited re-use through explicit modularization (see section entitled “Automatic Discovery of Protein Motifs” in [18]) producing an extremely simple example of a grammar that involved *Automatic Defined Functions (ADFs)*. The problem involved evolving a motif-detecting program. Motifs (such as those found in the PROSITE database [97]) are very simple grammars used by molecular biologists to describe protein sequences. This particular problem was solved using ADFs to organize the amino acid residues into various subsets of interchangeable residues (corresponding to the square brackets of PROSITE expressions). It used the *result-producing branch* to decide whether or not a given protein subsequence is an instance of the motif by conjunctively joining the outcomes of the matching at each position of the protein sequence. The salient advantage of Genetic Programming in this particular domain being that it can not only explicitly

exploit combinations of variables (in this case amino acids in particular relative positions that happen to be contiguous and constituting the so-called *dead box*), but also it can as easily deal with non-contiguous variables. Practically no other method of Computational Intelligence can do this with any facility, and it is vital in this particular domain.

5 Genetic Variability and Genetic Markers

In Genetics as in Computer Science the term *polymorphism* means “having many forms”. It is the regular and simultaneous occurrence in a single interbreeding population of two or more discontinuous genotypes and occurs when two or more alleles are maintained at frequencies greater than 1 percent. Differences in genotypes can range in size from a single nucleotide site to large nucleotide sequence visible at a chromosomal level. Readers may also wish to become familiar with the following notions before tackling this section as they affect the distribution of differences between genomes: natural selection; gene flow; genetic drift; founder effect; population bottleneck; molecular clock; and the neutral theory of molecular evolution [100].

Certain rare events change the constitution of the genome and over many generations (vast time) their accumulated effect is significant and noticeable in the genome. The human genome is vast (more than three billion nucleotide pairs). This and the redundancy in the translation offer ample opportunity for tolerance to mutations. Random interactions with the environment or cellular operations can cause what are called *spontaneous* mutations and the rate at which they occur is called the *background level*. The single point mutation and the tandem repeat are the two types of mutation of interest. The tandem repeat mutations occur when the machinery for copying DNA makes a mistake and inserts a different number of sequence copies. Both single point location differences and differences in tandem repeats between genomes are genetic markers. Markers are essential for genetic fingerprinting, finding the causes of diseases, and measuring ecological diversity. Strictly speaking, a marker is any segment of DNA with an identifiable physical location on a chromosome whose inheritance can be followed. It can be a gene, or some section of DNA with no known function. Because DNA segments that lie near each other on a chromosome tend to be inherited together, markers are often used as indirect ways of tracking the inheritance patterns of genes that have not yet been identified, but whose approximate locations may be known. There exist many genetic markers (particularly for Ecology [101]).

A lot of effort has been expended to produce maps of genetic markers that are based on tandem repeats and single point changes. The chemical method used to identify the first tandem repeats displayed them as peaks of very high value above all other peaks and resulted in use of the name “satellite” to denote them:

- Single Nucleotide Polymorphism (SNP) are differences in a single nucleotide at different parts of the genome. They represent about 1 percent of the genome. In humans they occur every 1400 bp in inter genetic regions and every 1430 bp in the coding regions of the gene. Because only about 3 to 5 percent of a person's DNA sequence codes for the production of proteins, most SNPs are found outside of "coding sequences". Finding single nucleotide changes in the human genome seems like a daunting prospect, but over the last 20 years, biomedical researchers have developed a number of techniques that make it possible to do just that. Each technique uses a different method to compare selected regions of a DNA sequence obtained from multiple individuals who share a common trait. In each test, the result shows a physical difference in the DNA samples only when a SNP is detected in one individual and not in the other.
- Microsatellite are runs of mono-, di- and trinucleotide repeats (usually 3 to 4 base pairs are repeated from 10 to 100 times) that can represent a sizeable proportion of the genetic material and are associated with the instability of the genome. They are also known as *dynamic* mutations [102] because they exhibit a strong level of instability, undergoing additions or deletions of repeated units, which lead to variations in the number of copies of the repeated stretches (highly polymorphic number of repeats between individuals of the same species). Their high polymorphism allows individuals to be reliably distinguished and it arises because the repeated simple sequences cause a high frequency of loss or insertion of additional repeats by confusing the DNA replication machinery. Interestingly, cancer cells possess damaged replication control causing microsatellites to be gained and lost at high frequency during each round of mitosis. A cancerous cell line, therefore, can show a dissimilar genetic fingerprint from that of the normal host tissue. Microsatellites are popular for population studies. They are present in all genomes and are particularly abundant in eukaryotes and find use in genetic fingerprinting and paternity testing. They are present in coding and non-coding parts of the genome [103].
- Minisatellites are DNA sequences of a tandem repeated sequence 10 to 100 nucleotides long. The overall size of the minisatellite is between 100 and 2000 base pairs.

Other than size and number of repeats what further justifies the distinction between minisatellites and microsatellites is their mutational mechanism. The latter mutate primarily by strand slippage during replication and the former by recombinational processes [104]. The discovery that the tandem repeats are connected with evolutionary pressure, and that the alteration in the number of repeats is associated with disease (particularly triple expansions of repeated sequences [104]) was unexpected at the time of its discovery.

Most SNPs are not responsible for a disease state but instead serve as biological markers for pinpointing a disease on the human genome map because they are usually located near a gene found to be associated with a certain

disease. Occasionally, a SNP may actually cause a disease and, therefore, can be used to search for and isolate the disease-causing gene.

5.1 Search for Genetic Factors Involved in Diseases

There are broadly speaking two types of disease: monogenetic diseases and common complex disorders:

1. Though rare, there exist thousands of monogenetic diseases that are incurable and lack effective treatment. An abnormal and poorly functioning gene at a locus causes disease as in Huntington's disease. If some form of genetic repair were successful then DNA screening the population for the gene could result in early intervention because most have delayed manifestations until midlife and then are debilitating progressive.
2. Common complex disorders are more prevalent in the population at large. These are the big killers such high blood pressure and diabetes and can involve a number of genes interacting with environmental factors.

Each person's genetic material contains a unique SNP pattern that is made up of many different genetic variations. To create a genetic test that will screen for a disease in which the disease-causing gene has already been identified, scientists collect blood samples from a group of individuals affected by the disease and analyze their DNA for SNP patterns. Next, researchers compare these patterns to patterns obtained by analyzing the DNA from a group of individuals unaffected by the disease. This type of comparison, called an "association study", can detect differences between the SNP patterns of the two groups, thereby indicating which pattern is most likely associated with the disease-causing gene.

Familial aggregation studies designed to identify the causes of the common complex disorders need to use founder populations, as in Iceland, because of the need to identify only those genomic differences that matter. SNPs and microsatellites play a big part in such studies. A recent study [105] shows that even in the small and relatively homogeneous population of Iceland, there exist sub-populations with discernible genetic differences. It concludes that this population structure could be taken into account particularly in the design of large-scale association studies to correlate genetic variation with susceptibility to disease. The objective is to find such associations when they exist while minimizing the risk of false positive results. If SNP profiles for diseases could be established individuals could be screened for susceptibility to the disease.

A useful concept is *linkage disequilibrium* or the deviation from probabilistic independence between alleles at two different loci (for a mathematical definition for multiple loci see [106]). This deviation from independence can have different causes such as any number of evolutionary forces. Its presence is an indication that either stochastic (drift) or deterministic (selection, gene flow) evolutionary forces have been acting on a population.

Genetic Algorithms have been applied to discover haplotypes of SNPs (see Sect. 4.3 for a definition of haplotype) in a linkage disequilibrium study to identify factors in diabetes and obesity (these are common complex disorders), i.e. the Genetic Algorithm discovered candidate haplotypes that could explain the disease [107]. Other investigators have applied data mining techniques based on genetic algorithms to discovery of SNP patterns [108].

5.2 Evolutionary Computation and Phylogeny

The research field of Phylogeny asks the question about the origin of the molecule of interest. Where did it come from? Cladistics is a branch of biology that determines the evolutionary relationships of living things based on derived similarities. A start has been made to apply evolutionary computation to the difficult combinatorial problem of Phylogenetic Inferencing [109, 110], which requires heuristic approaches. These papers explore a number of evolutionary computation variations to determine their suitability for addressing this problem.

The appearance of new species frequently occurs in rapid bursts and this has caused a great deal of debate amongst scientists. For more than 60 years, evolutionary biologists have debated the issue of whether the processes of genetic change observable within populations (micro-evolution) can provide an adequate explanation for the large-scale patterns in the history of life (macro-evolution [101, 111]). In general, population geneticists have argued in favor of micro-evolutionary extrapolation, while paleontologists have sought to establish an autonomous and hierarchical macro-evolutionary theory based on the operation of selection at several levels of biological organization (especially species) [111]. The late John Maynard Smith considered the interaction between both disciplines of paramount importance [112].

Biology tends to be a descriptive undertaking, and therefore it is important for evolutionary computation to help Biology answer the big questions such as the ones considered by Maynard Smith [113] and others. For example, it could attempt to test theories of Natural evolution and of alternative life forms by numerical simulation to answer fundamental questions about Life. Evolutionary computation has already made serious attempts at recreating aspects of natural evolution on Earth in terms of computer simulation [114, 115, 116], and these will continue.

Another big question concerns the variable size of the genome for different species (and the C-value enigma [111]). Genetic Programming researchers and researchers using a variable length representation Genetic Algorithm (e.g. Grammatical Evolution [19]) are familiar with the inter-related observations of bloat (sudden growth in the length of the individual) and introns (ineffective code) [117, 118, 119, 120]. Rosca gives a mathematical proof [121] of these phenomena for the variable length representation in evolutionary computation. His result might be relevant to the evolution of genomes in Nature and to the C-value enigma.

6 Conclusions

The purpose of this chapter is to bring the Computational Intelligence and Biology communities closer together in a shared understanding of domain issues in Bioinformatics. Knowledge of cell biochemistry is advancing at a fast pace and the most pressing objectives are for rapid and cost-effective progress in Medicine, Public Health, and Environmental Science. Opportunities for cost-effective advances are possible in the interdisciplinary direction but this requires educating computer scientists about Biology and vice-versa as pre-requisite for this cost-effective progress to take place. This interaction is important because the costs of research in drug discovery can be very high.

Acknowledgments

Many have provided advice and useful comment and I am grateful to them. In particular I would like to thank Joseph Kolibal, Stephen Firth, Richard Brankin, and Philip Varney for assisting me in this endeavour.

References

1. Negoita, M. Gh., Neagu, C.-D., Palade, V., *Computational Intelligence-Engineering Intelligent Hybrid Systems*, Springer Verlag, (2005). [245](#)
2. Lewin, B., *Genes VII*, Oxford University Press (2000). [245](#)
3. Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., Walter, P., *Molecular Biology of The Cell*, Garland Science, 4th Edition (2002). [245](#)
4. Altman, R. B., *Molecular Biology for Computer Scientists Tutorial*, in John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba and Rick L. Riolo (editors), Genetic Programming 1997: Proceedings of the Second Annual Conference (GP97), Stanford University, CA, USA, 18–26, Morgan Kaufmann (1997). [245](#)
5. Hartman, H., Fedorov, A., *The origin of the eukaryotic cell: A genomic investigation*, Proceedings of the National Academy of Science of the United States of America (PNAS), January (2002). [246](#)
6. Freeland, S.J. and Hurst, L.D., *Evolution encoded*. Scientific American 290:84–91 (2004). [247](#)
7. Freeland, S.J., *The genetic code: an adaptation for adapting?*, Journal of Genetic Programming and Evolvable Machines 3:113–127 (2002). [247](#)
8. Stella, G., Ardell, D. H., *The Impact of Message Mutation on the Fitness of a Genetic Code*, J Mol Evol 54:638–651 (2002). [247](#)
9. Nissen, P., Hansen, J., Ban, N., Moore, P.B., Steitz, T.A. *The structural basis of ribosome activity in peptide bond synthesis*, Science 289(5481): 920–930 (2000). [247](#)
10. Pauling, L., Corey, R.B., *Two Hydrogen Bonded Spiral Configurations of the Polypeptide Chain*, Journal of the American Chemical Society 72, 5349 (1950), or Pauling, L., Corey, R.B., Proc. Natl. Acad. Sci. USA 37, 251256 (1951). [249](#)

11. Venkatachalam, C.M., *Stereochemical Criteria for Polypeptides and Proteins. V Conformation of a System of Three Linked Peptide Units*, Biopolymers 6, 1425–1436 (1968). [249](#)
12. Gresh, D., Suits, F., Sham, Y., *Case Study: An Environment for Understanding Protein Simulations Using Game Graphics*, Proceedings of the IEEE Conference on Visualization, 445–8 (2001). See also <http://alphaworks.ibm.com/tech/ppv> [249](#), [262](#)
13. Kell, D.B., *Genotype-phenotype mapping: genes as computer programs*, Trends in Genetics, 18(11):555:559, (2002). [249](#)
14. Beer, S., *Decision and Control: The Meaning of Operational Research and Management Cybernetics*, Wiley Press (1994). ISBN: 0-471-94838-1. [249](#)
15. Viguera, E., Cancéll, D., Ehrlich, D.S., *Replication slippage involves DNA polymerase pausing and dissociation*, The EMBO Journal, Vol. 20, No. 10 pp. 2587–2595, (2001). [250](#)
16. Koza, J.R., *Genetic Programming*, MIT Press (1992). [250](#)
17. Koza, J.R., *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press (1994). [250](#), [263](#)
18. Koza, J.R., Bennett III, F.H., Stiffelman, O., *Genetic Programming as a Darwinian Invention Machine*. Lecture Notes in Computer Science 1598, 93–108 (1999). [250](#), [263](#)
19. O'Neill, M., Ryan, C., *Grammatical Evolution*, IEEE Trans. Evolutionary Computation 5, No. 4, 349358 (2001). [250](#), [267](#)
20. Kell, D.B., *Metabolomics and Machine Learning: Explanatory Analysis of Complex Metabolome Data Using Genetic Programming to Produce Simple, Robust Rules*, Molecular Biology Reports, 29(1–2):237–241, (2002). [250](#)
21. Banzhaf, W., *Artificial Regulatory Networks and Genetic Programming*, Chap. 4 in Genetic Programming Theory and Practice, Kluwer Academic Publishers (2003). [250](#)
22. Koza, J.R., Mydlowec, W., Lanza, G., Yu, J., Keane, M.A., *Reverse Engineering of Metabolic Pathways from Observed Data Using Genetic Programming*, Pacific Symposium on Biocomputing 6:434–445 (2001) or <http://helix-web.stanford.edu/psb01/koza.pdf> [250](#), [255](#)
23. Wolfram S., *A New Kind of Science*, Wolfram Media Inc., (2002). [250](#)
24. Zadeh, L., *From Computing with Numbers to Computing with Words – From Manipulation of Measurements to Manipulation of Perceptions*, IEEE Transactions on Circuits and Systems, 45, 105–119, (1999). [250](#), [276](#)
25. Tsien, C.L., Libermann, T.A., Gu, X., Kohane, I.S., *On Reporting Fold Differences*, Pacific Symposium on Biocomputing 6:496–507 (2001). [250](#)
26. Dickerson, J.A., Berleant, D., Cox, Z., Qi, W., Wurtele, E., *Creating Metabolic Network Models using Text Mining and Expert Knowledge*, Atlantic Symposium on Molecular Biology and Genome Information Systems and Technology (CBGIST 2001), 26–30, Durham, North Carolina, March, (2001) or <http://orion.math.iastate.edu/danwell/GET/Dickersonfinalv1.pdf> [250](#)
27. Fogel, G.B., Corne, D.W., *An Introduction to Bioinformatics for Computer Scientists*, Chap. 1 of: *Evolutionary Computation in Bioinformatics*, Morgan Kauffman Publishers (2003). [250](#)
28. Wimpenny J.W., *The validity of models*, Adv Dent Res., 11(1):150–9 (1997). [253](#)
29. Hu, J., Goodman, E.D., Seo, K., *Continuous Hierarchical Fair Competition Model for Sustainable Innovation in Genetic Programming*, Chap. 6 in Genetic Programming Theory and Practice, Kluwer Academic Publishers (2003). [253](#)

30. Howard, D., *Modularization by Multi-Run Frequency Driven Subtree Encapsulation*, Chap. 10 in Genetic Programming Theory and Practice, 155–172, Kluwer Academic Publishers (2003). [253](#)
31. Ryan, C., Keijzer, M., Cattolico, M., *Favorable Biasing of Function Sets Using Run Transferable Libraries* Chap. 7 in Genetic Programming Theory and Practice II, Kluwer Academic Publishers (2004). [253](#)
32. Kuska, B., Beer, Bethesda, and biology: how “genomics” came into being, *J Natl Cancer Inst.* 90(2):93 (1998). [253](#)
33. Nicholson, J.K., Connelly, J., Lindon, J.C., Holmes, E., *Metabonomics: A platform for studying drug toxicity and gene function*, *Nature Reviews Drug Discovery*. 1, 153–161 (2002). [254](#)
34. Nicholson, J.K., Lindon, J.C., Holmes, E., “Metabonomics”: understanding the metabolic responses of living systems to pathophysiological stimuli via multivariate statistical analysis of biological NMR spectroscopic data, *Xenobiotica* 29, 1181–1189 (1999). [254](#)
35. Lindon, J.C., Nicholson, J.K., Holmes, E., Everett, J.R., *Metabonomics: metabolic processes studied by NMR spectroscopy of biofluids*, *Concepts in Magnetic Resonance*. 12, 289–320 (2000). [254](#)
36. Nicholson, J.K., Holmes, E., Lindon, J.C., Wilson, D., *The Challenges of Modelling Mammalian Biocomplexity*, *Nature Biotechnology* 22 (10) 1268–1274 (2004). [255](#)
37. Raamsdonk, L.M., Teusink, B., Broadhurst, D., Zhang, N., Hayes, A., Walsh, M., Berden, J.A., Brindle, K.M., Kell, D.B., Rowland, J.J., Westerhoff, H.V., van Dam, K., Oliver, S.G. *A functional genomics strategy that uses metabolome data to reveal the phenotype of silent mutations*. *Nature Biotechnology* 19, 45–50 (2001). [255](#)
38. Lucasius C.B., Kateman, G. *Application of Genetic Algorithms in Chemometrics*, International Conference on Genetic Algorithms (ICGA-89): 170–176 (1989). [255](#)
39. Rowland, J.J., *Interpreting Analytical Spectra with Evolutionary Computation*, Chap. 16 in G.B. Fogel and D.W. Corne (editors) *Evolutionary Computation in Bioinformatics*, Morgan Kauffman Publishers (2003). [255](#)
40. Scientific American, *Cybernetic Cells*, Special Issue, August 2001. [255, 256](#)
41. Koza, J.R., Keane, M.A., Streeter, M.J., Mydlowec, W., Yu, J., Lanza, G., *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*, Kluwer Academic Publishers (2003). [255](#)
42. Kitagawa, J., Iba, H., *Identifying Metabolic Pathways and Gene Regulation Networks with Evolutionary Computation*, Chap. 12 in G. B. Fogel and D. W. Corne (editors) *Evolutionary Computation in Bioinformatics*, Morgan Kauffman Publishers (2003). [255](#)
43. Fox, T.C., Green, B.J., Kennedy, R.A., Rumpho, M.E., *Changes in Hexokinase Activity in Echinochloa phyllopogon and Echinochloa crus-pavonis in Response to Abiotic Stress*, *Plant Physiol.* 118: 1403–1409 (1998). [256](#)
44. Garrett, R.H., Grisham, C.M., *Biochemistry*, Harcourt College Publishers, 2nd Edition (1999). [256](#)
45. Kaczer H., Burns J.A., *The control of flux*, *Symp Soc Exp Bot* 28: 65–104 (1973). [256](#)
46. Private communication with Pedro Mendes, Virginia Bioinformatics Institute, Virginia, USA (October 2004). [256](#)

47. Weckwerth, W., Wenzel, K., Fiehn, O., *Process for the integrated extraction, identification and quantification of metabolites, proteins and RNA to reveal their co-regulation in biochemical networks*, Proteomics Volume 4, Issue 1, 78–83 (2004). [256](#)
48. Weckwerth, W., Fiehn, O., *Can we discover novel pathways using metabolomic analysis?*, Curr. Opin. Biotechnol. 13, 156–160 (2002). [256](#)
49. Martins, A.M., Camacho, D., Shuman, J., Sha, W., Mendes, P., Shulaev, V., *A Systems Biology Study of Two Distinct Growth Phases of *Saccharomyces cerevisiae* Cultures*, Current Genomics, December 2004, 5(8):649–663(15) (2004).
50. Bell, S.L., Palsson, B.O., *ExPA: A Program for Calculating Extreme Pathways in Biochemical Reaction Networks*, Bioinformatics, in press (2005). [256](#)
51. Cheverud, J., Routman, E., *Epistasis and its contribution to genetic variance components*, Genetics 130: 1455–1461 (1995). [257](#)
52. Wagner, G.P., Laubichler, M.D., Bagheri-Chaichian, H., *Genetic Measurement Theory of Epistatic Effects*, Genetica 102/103, 569–580 (1998). [257](#)
53. Wagner, G.P., Altenberg, L., *Complex Adaptations and the Evolution of Evolvability*, Evolution 50(3): 967–976 (1996). [257](#)
54. Fisher, R.A., *The correlation between relatives on the supposition of Mendelian inheritance*, Trans. Roy. Soc. Edinburgh 52, 399–433 (1918). [257](#)
55. Templeton, A.R., *The general relationship between average effect and average excess*. Genet Res. 49(1):69–70 (Feb. 1987). [257](#)
56. Tsonis, P.A., *Anatomy of Gene Regulation*, Cambridge University Press, (2003). ISBN 0-521-80030-7. [257](#)
57. Holmes, B., *Location location location: who would have thought that a chromosome's position in the nucleus would be so crucial to the functioning of its genes?*, New Scientist Magazine, issue 2386, 15 March (2003). [258](#)
58. <http://www.epd.isb-sib.ch/> [258](#)
59. Petridge, D.S., *Predicting Pol-II Promoter Sequences Using Transcription Factor Binding Sites*, Journal of Molecular Biology, 249: 923–932 (1995). [258](#)
60. Pedersen, A.G., Baldi, P., Chauvin, Y., Brunak, S., *The Biology of Eukaryotic Promoter Prediction – a Review*, Computers and Chemistry, 23: 191–207 (1999). [258](#)
61. Pedersen, A.G., Engelbrecht, J., *Investigations of *Escherichia coli* Promoter Sequences with ANN: New Signals Discovered Upstream of the Transcription Start Point*, Proceedings of Third International Conference on Intelligent Systems for Molecular Biology, 292–299 (1995). [258](#)
62. Private communication: Thomas Werner, Genomatix, Munich, Germany (2003). [258](#)
63. Levitsky, V.G., Katokhin, A.V., *Recognition of eukaryotic promoters using a genetic algorithm based on iterative discriminant analysis*, In Silico Biology 3, 0008 (2003). [258](#)
64. Howard, D., Benson, K.A., *Evolutionary computation method for pattern recognition of cis-acting sites*, Biosystems 72(1–2):19–27, (2003). [258](#)
65. Ashlock, D., *GP-Automata for Dividing the Dollar*, in John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba and Rick L. Riolo (editors), Genetic Programming 1997: Proceedings of the Second Annual Conference, Stanford University, CA, USA, 18–26, Morgan Kaufmann (1997). [258](#)
66. Driscoll, J.A., Worzel, B., MacLean, D. *Classification of Gene Expression Data with Genetic Programming*, Chap. 3 in Genetic Programming Theory and Practice, 25–42, Kluwer Academic Publishers (2003). [259](#)

67. Moore, J.H., Parker, J.S., *Evolutionary computation in microarray data analysis* in: Methods of Microarray Data Analysis, Kluwer Academic Publishers, (2005) in press. [259](#)
68. Ando, S., Iba, H., *Artificial Immune System for Classification of Cancer*, Applications of Evolutionary Computing, Springer Lecture Notes in Computer Science 2611, 1–10 (2003). [259](#)
69. Falkenauer, E., Marchand, A., *Clustering Microarray Data with Evolutionary Algorithms*, Chap. 10 in G. B. Fogel and D. W. Corne (editors) Evolutionary Computation in Bioinformatics, Morgan Kauffman Publishers (2003). [259](#)
70. Reif, D.M., et al., *Complex Function Sets Improve Symbolic Discriminant Analysis of Microarray Data*, Genetic and Evolutionary Computation Conference (GECCO 2003), 2277–2287 (2003). [259](#)
71. Lee, S.-K., et al., *Finding the Optimal Gene Order in Displaying Microarray Data*, Genetic and Evolutionary Computation Conference (GECCO 2003), 2215–2226 (2003). [259](#)
72. Shyu, C., Foster, J.A., *Evolving Consensus Sequence for Multiple Sequence Alignment with a Genetic Algorithm*, Genetic and Evolutionary Computation Conference (GECCO 2003), 2313–2324 (2003). [259](#)
73. Shortle, D., Wang, Y.I., Gillespie, K.R., Wrabl, J.O., *Proteins folding for realists: A timeless phenomenon*, Protein Science 5: 991–1000 (1996). [259](#)
74. Bird, G.A., *Molecular Gas Dynamics and the Direct Simulation of Gas Flows*, Clarendon Press, Oxford (1994). [260](#)
75. Alexander, F. J., Garcia, A. L., *The Direct Simulation Monte Carlo*, Computers in Physics, 11:588–593 (1997). [260](#)
76. Libersky, L.D., Petschek, A.G., Carney, C.C., Hipp, J.R., Allabadi, F.A., *High strain Lagrangian hydrodynamics: a three-dimensional SPH code for dynamic material response*, J. Comput. Phys., 109:67–75 (1993). [260](#)
77. <http://predictioncenter.llnl.gov/casp6/submit/casp6-ver.html> and Proteins: 23(5), 1995; Suppl 1, 1997; Suppl 3, 1999; Suppl 5, 2001; Suppl 6, 2003. [260](#)
78. <http://anteater.blueprint.org/science.html> [260](#)
79. Falkenauer, E., Marchand, A., *On the Evolutionary Search for Solutions of the Protein Folding Problem*, Chap. 6 in G. B. Fogel and D.W. Corne (editors) Evolutionary Computation in Bioinformatics, Morgan Kauffman Publishers (2003). [260](#)
80. Lamont, G.B., Merkle, L.D., *Toward Effective Polypeptide Structure Prediction with Parallel Fast Messy Genetic Algorithms*, Chap. 7 in G.B. Fogel and D.W. Corne (editors) Evolutionary Computation in Bioinformatics, Morgan Kauffman Publishers (2003). [260](#)
81. Lamont, G.B., Merkle, L.D., *Application of Evolutionary Computation to Protein Folding with Specialized Operators*, Chap. 8 in G.B. Fogel and D.W. Corne (editors) Evolutionary Computation in Bioinformatics, Morgan Kauffman Publishers (2003). [260](#)
82. Gunn, J.R., *Hierarchical Minimization with Distance and Angle Constraints*, Proc. Sixth International Conference on Intelligent Systems for Molecular Biology (ISMB-98), 78–84, The AAAI Press (1998). [260](#)
83. Chong, L.T., Snow, C.D., Rhee, Y.M., Pande, V.S., *Dimerization of the p53 Oligomerization Domain: Identification of a Folding Nucleus by Molecular Dynamics Simulations*, Journal of Molecular Biology (2005). [261](#)
84. Vendruscolo, M., *Simultaneous determination of protein structure and dynamics*, Nature 433, 128 (2005). [261](#)

85. Lenz, P., Zagrovic, B., Shapiro, J., Pande, V.S. *Folding Probabilities: A Novel Approach to Folding Transitions and the Two-dimensional Ising Model* J. Chem. Phys., 120 (14), 6769–6778 (2004). [261](#)
86. <http://folding.stanford.edu/> [261](#)
87. Paci, E., Cavalli, A., Vendruscolo, M., Caflisch, A., *Analysis of the distributed computing approach applied to the folding of a small β peptide*, Proceedings of the National Academy of Sciences of the United States of America (PNAS), 100: 8217–8222 (2003). [261](#)
88. Singhal, N., Snow, C.D., Pande, V. S., *Using path sampling to build better Markovian state models: Predicting the folding rate and mechanism of a tryptophan zipper beta hairpin*, Journal of Chemical Physics (2004). [261](#)
89. Swope, W.C., Pitera, J.W., Suits, F., *Describing Protein Folding Kinetics by Molecular Dynamics Simulations 1. Theory*. J. Phys. Chem. B 108: 6571–6581 (2004). [262](#)
90. Pitman, M., Eleftheriou, M., Fitch, B.G., Germain, R.S., Rayshubski, A., Ward, T.J.C. Zhestkov, Y., Zhou, R. *Describing Protein Folding Kinetics by Molecular Dynamics Simulations. 2. Example Applications to Alanine Dipeptide and a Beta-Hairpin Peptide*. J. Phys. Chem. B 108: 6582–6594 (2004). [262](#)
91. Choi, I.-G., Kwon, J., Kim, S.-H., *Local feature frequency profile: A method to measure structural similarity in proteins*, Proc. of the Nat. Acad. Sciences. 101: (2004). [261](#)
92. Ji, S., *Isomorphism between cell and human languages: molecular biological, bioinformatic and linguistic implications*, BioSystems 44(1):17–39 (1997). [261](#)
93. Thomsen, R., *Flexible Ligand Docking Using Evolutionary Algorithms: Investigating the Effects of Variation Operators and Local Search Hybrids*, BioSystems, 72 (1–2): 57–73 (2003). [262](#), [263](#)
94. Clark, D.E. (editor), *Evolutionary Algorithms in Molecular Design*, Wiley (2000). [263](#)
95. Clark, D.E., Westhead, D.R., *Evolutionary Algorithms in computer-aided molecular design*, J. Comput. Aided Mol. Des. 10:337–358 (1996). [263](#)
96. Yang, J.-M. *An Evolutionary Approach to Molecular Docking*, Genetic and Evolutionary Computation Conference (GECCO 2003), 2372–2383 (2003). [263](#)
97. <http://www.expasy.uniprot.org/index.shtml> or <http://www.expasy.org/sprot/> or <http://www.expasy.org/prosite/> [263](#)
98. <http://www.rcsb.org/pdb/> [249](#), [263](#)
99. Miseta, A., Csutora, P., *Relationship Between the Occurrence of Cysteine in Proteins and the Complexity of Organisms*, Molecular Biology and Evolution, 17:1232–1239 (2000). [263](#)
100. Kimura, M., *Neutral theory of molecular evolution*, Cambridge University Press, (1983). ISBN 0-521-23109-4. [264](#)
101. Avise, J. C., *Molecular Markers, Natural History, and Evolution*, Second Edition, Sinauer Associates Publications (2004). ISBN 0-87893-041-8. [264](#), [267](#)
102. Richards, R.I., Sutherland, G.R., *Dynamic Mutations : A new class of mutations causing human disease*, Cell 70: 709–712 (1992). [265](#)
103. Toth, G., Gaspari, Z., and Jurka, J., *Microsatellites in different eucaryotic genomes: Survey and analysis*, Genome Res., 10: 967–981 (2000). [265](#)
104. Hancock, J.M., Santibanez-Koref, M.F., *Trinucleotide expansion diseases in the context of micro and minisatellite evolution*, EMBO J., 17, 55215524 (1998). [265](#)

105. Helgason, A., Yngvadóttir, B., Hrafnkelsson, B., Gulcher, J., Stefánsson, K., *An Icelandic example of the impact of population structure on association studies*, Nature Genetics 37, 90–95 (2005). [266](#)
106. Gorelick, R., Laubichler, M.D. *Decomposing Multilocus Linkage Disequilibrium*, Genetics 166: 1581–1583 (March 2004). [266](#)
107. Jourdan, L., Dhaenens, C., Talbi, El-Ghazali, *Discovering Haplotypes in Linkage Disequilibrium Mapping with an Adaptive Genetic Algorithm*, Applications of Evolutionary Computing, Springer Lecture Notes in Computer Science 2611, 66–75 (2003). [267](#)
108. Shah, S.C., Kusiak, A., *Data mining and genetic algorithm based gene/SNP selection*. Artif Intell Med., 31(3):183–96 (2004). [267](#)
109. Cotta, C., Moscato, P., *Inferring Phylogenetic Trees Using Evolutionary Algorithms* Parallel Problem Solving From Nature VII, J.J. Merelo et al. (eds.), Lecture Notes in Computer Science 2439, 720–729, Springer-Verlag Berlin, (2002). [267](#)
110. Cotta, C., Moscato, P., *A Memetic-aided Approach to Hierarchical Clustering from Distance Matrices: Application to Phylogeny and Gene Expression Clustering*, Biosystems, 72(1–2):75–97, (2003). [267](#)
111. Gregory, T.R., Macroevolution, hierarchy theory, and the C-value enigma, Paleobiology, 30(2):179–202 (2004). [267](#)
112. *Games and theories*, NewScientist, 14 June 2003, pp.48–51, (an interview with John Maynard Smith at the age of 83 and still an active researcher) (2003). [267](#)
113. *The Evolutionist*: Interview with the late John Maynard Smith: <http://www.lse.ac.uk/collections/evolutionist/jms.htm> [267](#)
114. Ray, T.S., *Selecting Naturally for Differentiation*, in: Koza, John R., Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick L. Riolo [eds.]. Genetic Programming 1997: Proceedings of the Second Annual Conference, Stanford University, 414–419, San Francisco, CA: Morgan Kaufmann (1997). [267](#)
115. Ray, T.S., *Evolving Complexity*, Artificial Life and Robotics 1(1): 21–26 (1997). [267](#)
116. Ray, T.S., Hart, J., *Tierra Tutorial*, Genetic and Evolutionary Computation Conference Tutorial Program, 367–394, Morgan Kaufmann, San Francisco. Presented at GECCO 1999 (Genetic and Evolutionary Computation Conference), Orlando, Florida, 13–17, (1999). [267](#)
117. Soule, T., *Operator Choice and Evolution of Robust Solutions*, Chap. 16 in Genetic Programming Theory and Practice, 257–270, Kluwer Academic Publishers (2003). [267](#)
118. Langdon, W.B., Poli, R., *Fitness Causes Bloat: Mutation*, in Proceedings of EuroGP 98, Lecture Notes in Computer Science, LNCS 1391, 37–48 (1998). [267](#)
119. Soule, T., Foster, J.A., *Removal Bias: A New Source of Code Growth in Tree Based Evolutionary Programming*, in IEEE International Conference on Evolutionary Computation, 178–186, Anchorage, Alaska, IEEE Press (1998). [267](#)
120. Luke, S., *Modification Point Depth and Genome Growth in Genetic Programming*, Evolutionary Computation, 11(1):67–106 (2003). [267](#)
121. Rosca, J., *A Probabilistic Model of Size Drift*, Chap. 8 in Genetic Programming Theory and Practice, 119–136, Kluwer Academic Publishers (2003). [267](#)

A. Toy Problem Illustration

A.1 The Solution Fan Computation

A toy problem designed and solved by this author, and described here, demonstrates the search for the solution fan (see Sect. 3). Multiple solutions are obtained with Genetic Programming (GP). Details allowing replication of results are presented in the next section that describes how the problem was manufactured purely to illustrate the numerical experience of “finding a solution fan”.

In Fig. 1, locomotives share a track and move a discrete number of cells in discrete units of time or epochs (like on a board game). Four locomotives are fast (move one unit per epoch) and the remaining locomotive is slow (moves one unit every two epochs). The objective is to produce one computer program that is applied without change at every epoch. This sets the points in the track to avoid any locomotive collisions for a desired fixed number of epochs n_{ET} , e.g. 3000 epochs. If no locomotives should crash into one another over this desired time period then the problem is considered solved.

The problem is tackled with Genetic Programming. The fitness measure is $f = n_{EA} - n_{ET}$ where n_{EA} is the number of collision free epochs. The fitness of an individual is evaluated as follows. The individual is run at every epoch of the simulation. As it evaluates, at the end of each epoch, it may look at the location and orientation of certain locomotives and also at the states of certain points at junctions. At different stages during its execution, the program may decide to change the points of one of the junctions of the track (the computer program does so as it is being evaluated). At the end of its execution it may have changed the settings of none, or of any number of junctions. The next epoch now moves the locomotives. The simulation ends if there is a crash or, failing that, when n_{ET} epochs have transpired and perfect fitness is achieved: $n_{EA} = n_{ET}$ (this solves the problem).

During the Genetic Programming run, a genotype (the computer program) may be discovered that can solve the problem. Then its expression or phenotype (a list or dump of the position of all locomotives on the track at every epoch) is saved to a file. Immediately, that genotype is punished with a very unfavourable fitness and the run continues. When Genetic Programming discovers yet another computer program that can solve the problem, its expression is compared line by line with each stored solution (all files that have been saved thus far). If it is not identical to any stored solution then it is saved (it dumps all locomotive positions at every epoch to a file that is saved). Its fitness is set to a very unfavourable value and the run continues “forever”. This process produces a great number of solutions to this problem.

Next comes what is interesting and relevant to this chapter. When these stored files (the solutions) are “animated” through visualization, it becomes immediately apparent that certain strategies or emergent “features” have been discovered by Genetic Programming. Generally speaking, these can only be

described qualitatively using linguistic labels (as in [24]). At the start of the simulation trains are facing each other in the track as in table 1 (only 5 out of 6 locomotives are used). However, when we animate, we notice that the computer program sets the junctions in such order to cause all locomotives to move in the same direction. This strategy reduces the chance of a collision. Another strategy that is apparent from observation of many discovered solutions is that a single slow locomotive takes a different tour to the rest of the locomotives. Consider solution S1 that is illustrated in the middle of Fig. 5. There are four panels in that figure: top left (TL), top right (TR), bottom left (BL) and bottom right (BR) and the circled cell in each panel is the starting cell for the tour (route), i.e. the starting point of travel. Then by viewing the animated solution we observe that the slow locomotive takes the tour: TL, BL and TR while the fast locomotives take the tour BR. The fast locomotives all travel at the same speed and never collide with one another. Now consider solution S2 as illustrated in bottom of Fig. 5. Here the fast locomotives take the tour: TL, BL, TR, and the slow locomotive takes the tour BR. In this case, however, the slow locomotive covers the whole track and all of the fast locomotives somehow stay out of its path at the right moments in time. This strategy could be recognized by the linguistic label “slow train avoidance”.

Many more solutions are discovered for this problem, for example as in Fig. 6 that isolates the slow locomotive from the fast locomotives – they do not share any part of the track. Note that a solution is defined as the phenotype. Different genotypes could be operationally equivalent.

These tactics or emergent features are important because although they are imprecise linguistic labels they help to organize taxonomy perhaps by clustering phenotypes by similarity and difference along the space that is defined by these emergent features.

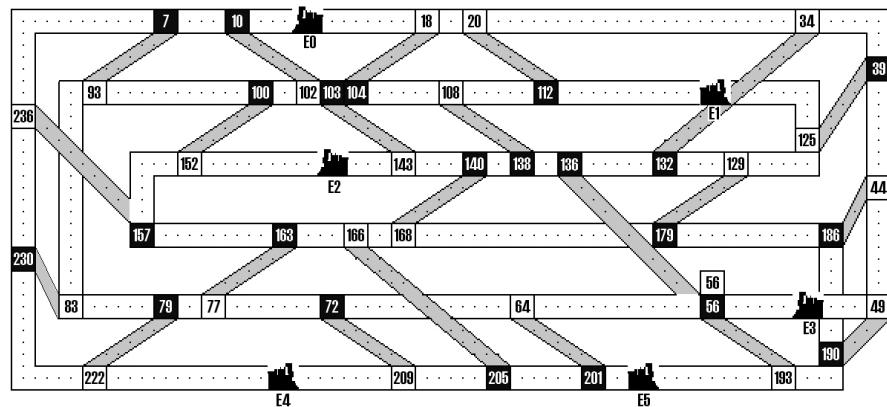
A.2 Toy Problem Details

The intention of the toy problem is definitely not to devise a solution of a real collision avoidance control system (for this could be achieved more efficiently in other ways).

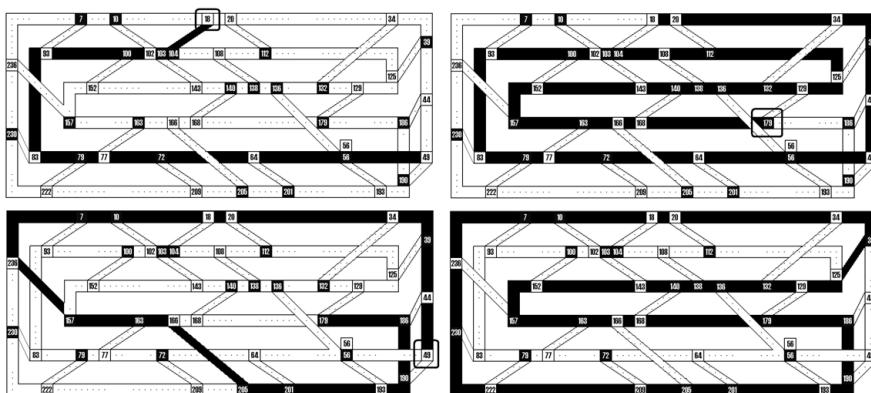
Each junction has a unique direction of travel. Locomotives can only enter from the track segment numbers that are indicated as white numbers, and exit at track segments that are indicated as black numbers (see Fig. 7). Also notice that locomotives can only enter the junctions from the correct angle of course. For example, only locomotives travelling against the numbering system may enter the junction that starts at track segment number 222 and which ends at track segment number 79 for example. Similarly, the same would be true for the junction involving nodes 18 and 102 but the locomotive would need to be travelling with the direction of the track numbering system to enter the junction involving nodes 20 and 112 for example.

The engines start their journeys from given locations and with different speeds and a given direction of travel (with or against the numbering system

PROBLEM:



SOLUTION S1:



SOLUTION S2:

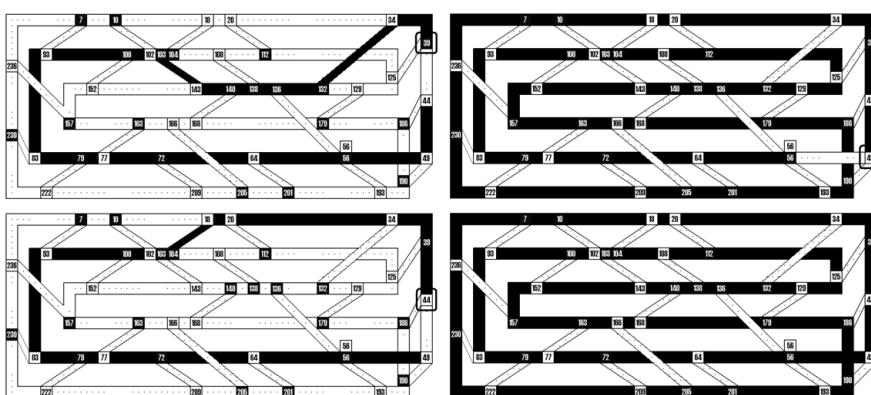


Fig. 5. Running a collision free railway (see text)

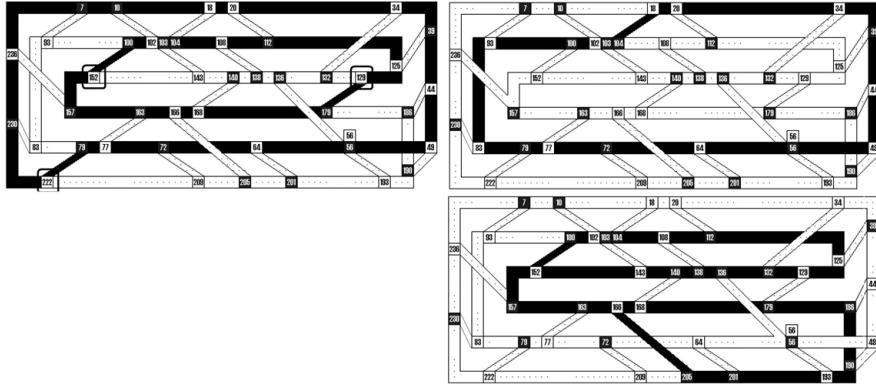


Fig. 6. *Left:* A very desirable and often extremely robust solution first positions the locomotives to follow one another (move in the same direction) and then proceeds to isolate the slow locomotive from the fast locomotives by segregating both groups in tours that are independent of one another (tours are indicated in black). *Right:* In an alternative solution which was observed to avoid a crash for 2121 epochs, slow and fast sets of locomotives move back and forth between the tours on the left and those on the *right* sides of this figure

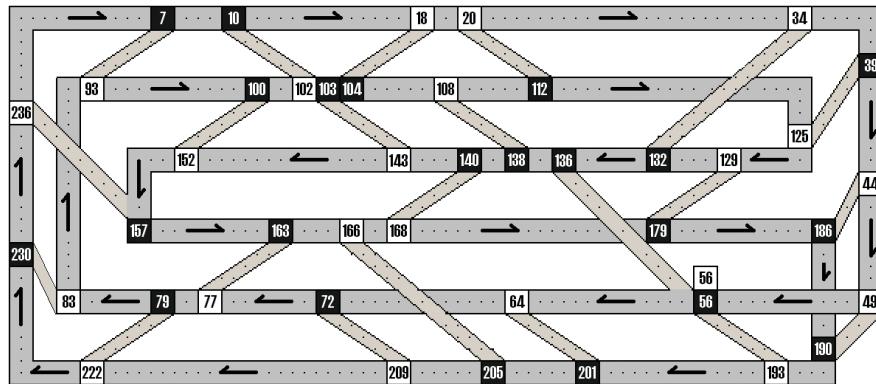


Fig. 7. The railway track. Arrows indicate the direction of the cell numbering system

of the track). The engine only occupies one section of track. This is illustrated in Fig. 5 and prescribed in Table 1. In the table, if locomotive i is positioned to move with increasing track numbering then $gtrain[i][0] = 1$ else $gtrain[i][0] = -1$. The array element $gtrain[i][2]$ indicates whether the locomotive is fast ("0") or slow ("1").

While transitioning from one part of the track to another an engine will need to cover a fixed number of track segments prior to emerging at the other side of the junction. Table 2 describes the junctions in full detail. Each junction has:

Table 1. Parameters for locomotives: array *gtrain* [locomotive] [property]

| <i>gtrain</i> [<i>i</i>] [<i>j</i>] | | <i>i</i> = 0 | <i>i</i> = 1 | <i>i</i> = 2 | <i>i</i> = 3 | <i>i</i> = 4 | <i>i</i> = 5 |
|---|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| direction of motion | <i>j</i> = 0 | -1 | 1 | 1 | 1 | -1 | -1 |
| starting cell position | <i>j</i> = 1 | 13 | 119 | 146 | 52 | 214 | 199 |
| speed: fast 0; slow 1 | <i>j</i> = 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| in-junction (counter) | <i>j</i> = 3 | 0 | 0 | 0 | 0 | 0 | 0 |

Table 2. Description of each junction: array *glink* [junction] [property] (see text)

| <i>glink</i> | <i>j</i> = 0 | <i>j</i> = 1 | <i>j</i> = 2 | <i>j</i> = 3 | <i>j</i> = 5 | <i>j</i> = 4 | <i>glink</i> | <i>j</i> = 0 | <i>j</i> = 1 | <i>j</i> = 2 | <i>j</i> = 3 | <i>j</i> = 5 | <i>j</i> = 4 | pt. set? |
|---------------|--------------|--------------|--------------|--------------|--------------|--------------|---------------|--------------|--------------|--------------|--------------|--------------|--------------|-------------|
| | acc. | start | end | sect. | swap? | pt. | | acc. | start | end | sect. | swap? | pt. | |
| | node | node | len. | | | set? | | node | node | len. | | | set? | |
| <i>i</i> = 0 | -1 | 18 | 104 | 3 | 0 | 0 | <i>i</i> = 11 | 1 | 108 | 138 | 3 | 1 | 0 | 0 |
| <i>i</i> = 1 | 1 | 20 | 112 | 3 | 0 | 0 | <i>i</i> = 12 | -1 | 125 | 39 | 6 | 0 | 0 | 0 |
| <i>i</i> = 2 | -1 | 34 | 132 | 6 | 1 | 0 | <i>i</i> = 13 | 1 | 129 | 179 | 3 | 1 | 1 | 1 |
| <i>i</i> = 3 | 1 | 44 | 186 | 2 | 0 | 0 | <i>i</i> = 14 | 1 | 143 | 103 | 3 | 1 | 0 | 0 |
| <i>i</i> = 4 | 1 | 49 | 190 | 2 | 0 | 0 | <i>i</i> = 15 | -1 | 152 | 100 | 3 | 1 | 0 | 0 |
| <i>i</i> = 5 | 1 | 56 | 136 | 6 | 0 | 0 | <i>i</i> = 16 | 1 | 166 | 205 | 6 | 1 | 1 | 1 |
| <i>i</i> = 6 | -1 | 64 | 201 | 3 | 0 | 0 | <i>i</i> = 17 | 1 | 168 | 140 | 3 | 1 | 0 | 0 |
| <i>i</i> = 7 | -1 | 77 | 163 | 3 | 1 | 0 | <i>i</i> = 18 | 1 | 193 | 56 | 3 | 0 | 0 | 0 |
| <i>i</i> = 8 | 1 | 83 | 230 | 2 | 0 | 0 | <i>i</i> = 19 | 1 | 209 | 72 | 3 | 0 | 0 | 0 |
| <i>i</i> = 9 | 1 | 93 | 7 | 3 | 0 | 0 | <i>i</i> = 20 | -1 | 222 | 79 | 3 | 0 | 0 | 0 |
| <i>i</i> = 10 | -1 | 102 | 10 | 3 | 0 | 0 | <i>i</i> = 21 | -1 | 236 | 157 | 5 | 1 | 0 | 0 |

1. An ID number.
2. Direction of entry (1 is with increasing track segment numbering and -1 is with decreasing track segment numbering (as was already discussed).
3. Point of entry.
4. Point of exit.
5. Length between entry and exit in number of discrete track segments (e.g note from Fig. 7 that for example junction ID 5 has to be longer than the others as it crosses another part of the track by means of a bridge; in general the lengths correspond to the diagram but not particularly: e.g. junction ID 12).
6. The next column indicates whether on arrival the locomotive now travels in a reverse direction from how it was going originally (in the numbering system). For example to enter junction ID 2 the locomotive must be travelling against the numbering system but emerges from the junction at node 132 travelling with the numbering system, and this is indicated by “1”.
7. The last column indicates whether or not the junction starts with its point set, indicated by “1”, to guide the locomotive off the main track.

Table 3. GP parameters

| Parameter | Description |
|--------------|--|
| Terminals | <i>TrainInfo</i> , <i>SignalInfo</i> (see text); +ve reals (0.01 to 1.0 in 0.01 steps); and -ve reals (-1.0 to -0.01 in 0.01 steps). |
| Functions | <i>JunctionSet(a, b)</i> (see text); protected % $avg(a, b) = (a + b)/2.0; +; -; *$ |
| Regeneration | steady-state; 70% cross-over; 10% copy; 20% mutation. No elitism. |
| Tournament | Breed Tourn Size = 4 Kill Tourn Size = 2 |
| S-Tree | Max. nodes = 1,000 Max. Init. Expr. Depth = 6 |
| Population | Size = 10,000 Mate radius = 10,000 |

GP functions and terminals:

- Train Information Terminal: *TrainInfo(0 : m – 1)*: This GP terminal is a type of atom which is originally randomly chosen to be one of m variants. The parameter m was chosen to be $m = 55$. The idea behind this choice was to bias for certain locomotive parameters (array *gtrain* has 4 parameters). Table 4 shows how the cell position of the locomotive and the in-junction counter (its location if inside the junction) are given more weight in the evolution. This terminal retrieves one current information parameter for a locomotive. It represents a particular locomotive and information tuple.
- Signal Information Terminal: *SignalInfo(0 : 219)*: This GP terminal is a type of atom which is originally randomly chosen to be one of 220 variants. This terminal retrieves one of three selected parameters for one of the 22 junctions as shown in Table 4. It represents a particular junction and information tuple.
- Junction Point Setting Function: *JunctionSet(a, b)*: This is a GP function of arity 2 which takes an action of point setting for one of the 22 available junctions. The first parameter a is used to choose the signal. If the second parameter b is greater than zero then the junction signal will be changed ('on' goes to 'off' and vice-versa). Otherwise no action is taken on the signal (see Table 4).

Table 4. Values adopted by the *TrainInfo*(0 : 54) terminal(direction of motion, cell position; speed; in-junction counter) and by the *SignalInfo*(0 : 219) terminal (start node; end node; junction setting). The choices for these weights are rather arbitrary. The function *JunctionSet(A, B)* (pseudo-code below) uses the numerical difference between its first argument and the numerical location of each junction (array elements *glink*[0..21][1]) to choose a junction. It then uses the sign of its second argument to determine whether it should flip the point set of that junction or leave it unaltered

| <i>TrainInfo</i> | | <i>JunctionSet(A,B)</i> |
|------------------|------------------------------------|-------------------------------------|
| Atom value | Corresponding array value | |
| $0 \leq a < 10$ | <i>gtrain</i> [$a/2$] [0] | C is jct closest to <i>fabs</i> (A) |
| $10 \leq a < 30$ | <i>gtrain</i> [$(a - 10)/4$] [1] | IF B > 0 THEN |
| $30 \leq a < 40$ | <i>gtrain</i> [$(a - 30)/2$] [2] | IF <i>glink</i> [C] [4] = 0 THEN |
| $40 \leq a < 55$ | <i>gtrain</i> [$(a - 40)/3$] [3] | <i>glink</i> [C] [4] = 1 |
| | | ELSE |
| | | <i>glink</i> [C] [4] = 0 |
| | | END IF |
| | | END IF |

| <i>SignalInfo</i> | | |
|--------------------|------------------------------------|--|
| Atom value | Corresponding array value | |
| $0 \leq a < 66$ | <i>glink</i> [$a/3$] [1] | |
| $66 \leq a < 110$ | <i>glink</i> [$(a - 66)/2$] [2] | |
| $110 \leq a < 220$ | <i>glink</i> [$(a - 110)/5$] [4] | |

The evolution of the fitness function in this GP problem is illustrated in Fig. 8.

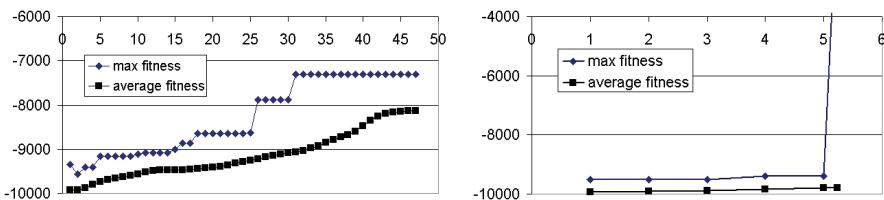


Fig. 8. Fitness versus generation (10,000 generates in steady state GP) plots for two parallel independent runs (pid). Typical progression of maximum and average fitness in a pid shows that fitness improves in jumps. The right pid illustrates an “eureka” moment, when GP suddenly discovers an outstanding solution (fitness = 0.0) at an early generation of the pid

Maximization of Combustion Efficiency: A Data Mining Approach

A. Kusiak and S. Shah

Abstract. Maximizing combustion efficiency with minimizing emissions is of importance to electric power industry. In this research, the impact of data transformation on boiler efficiency is investigated. The study showed that the data transformed with wavelet algorithms (Haar and Daubechies) provided better cross-validation accuracy, while moving average and wavelets resulted in similar prediction accuracy. The relationship between the length of the control horizon and prediction accuracy is studied. The study shows that the control horizon of a 3-hour to a half-week long provided acceptable prediction accuracy. An ensemble predictive model of the two control horizons is proposed to increase prediction accuracy. The research findings have established foundation for maximizing combustion efficiency by introduction of meta-controllers based on data mining algorithms.

Key words: *Combustion efficiency, data transformation, wavelets, control horizon, neighborhood analysis.*

1 Introduction

Fossil fuels, including coal, are the main source of energy production in the foreseeable future. In a coal-fired power plant, burning coal heats water that in turn produces steam. The steam energy powers turbines generating electricity. Boiler efficiency (defined as the theoretical energy output divided by the energy input) is typically in the range 75%–90%.

In this chapter, a data-mining approach to maximize combustion efficiency is proposed. Building a data-mining [1] meta-controller requires high accuracy predictive models. Since the process data is noisy and temporal, predictive models developed on “as-is” data are usually not accurate. Data transformation is used to de-noise the data and to account for process shifts (due to changing coal properties, boiler deterioration, and so on). Significant process shifts may result in frequent refreshing of the knowledge. This research focuses on the impact of data transformation (Sect. 3.2) and determination of

an acceptable control horizon of a predictive model Sect. (3.3). The literature review is presented in Sect. 2 while data preprocessing is discussed in (Sect. 3.1). The conclusions are presented in (Sect. 4).

2 Literature Review

Data transformation (arithmetic operators, joint features, discretization, de-noising methods, and so on) is useful in building robust data-mining models [2]. Kusiak [2] demonstrated significance of feature bundling in a semiconductor application to improve classification accuracy. The dynamic nature of the stock market is captured through exponential smoothing over different time intervals [3]. Burns et al. [4] applied genetic algorithm and decision-tree based wrapper to select significant parameters from “as-is” and transformed (with Haar wavelet and moving average) data. A brief overview of the data transformation schemes used in this research for de-noising is presented next.

Moving average (MA) transformation is averaging previous n observations to de-noise the incoming data. Based on computational experiments, a 20-minute moving average has been recommended for power plant applications considered in this research.

Wavelet algorithms are effective in data de-noising and data reduction [5]. The main prototype function, i.e., *mother wavelet*, yields a family of functions with different scales and amplitudes [6, 7]. The contracted, high frequency version of the prototype wavelet performs the primary analysis while a dilated low frequency version forms another analysis [6]. The wavelets may perform several passes to adjust the frequencies. The combinations of the two filters (analysis) are averaged and the original parameter signal is represented in terms of a wavelet expansion (i.e., a linear combination of wavelet coefficients). By adjusting the “viewing window”, wavelets capture the smallest variation in the input signal and maintain the time of discontinuity. Transformation of original (“as-is”) data into wavelet domain initiates the de-noising procedure [5]. The larger wavelet coefficients represent functional information, while smaller coefficients symbolize noise. Non-linear optimization (min-max function) of coefficients eliminates noise in the signal [5]. The final step of the de-noising procedure is transforming wavelet coefficients back to the original data domain. The commonly used wavelets are Haar and Daubechies [6, 7, 8].

The Haar wavelet is simple, compact, discontinuous, and it resembles a step function [7, 9]. Daubechies wavelets [8] are orthogonal. These wavelets have compact support with some sort of smoothness [9]. They follow the notation dbN, where db is the name of the wavelet, and N is a number representing the order. The first member of the Daubechies family db1 is similar to the Haar wavelet. The mathematical formulations for the Haar and Daubechies wavelets are discussed in [7, 8], and [9].

Sang and Li [10] explored predictability of network traffic in terms of the temporal prediction horizon (i.e., future traffic rate based on a given error

constraint) and minimum prediction error rate (based on predetermined time durations). The Auto-Regressive Moving Average (ARMA) and the Markov-Modulated Poisson Process (MMPP) models provided an upper bound for the optimal performance of the online traffic prediction. The increasing traffic prediction duration and differing traffic properties reduced the prediction accuracy while smoothing (low-pass filtering) and statistical multiplexing improved the predictability.

3 Experiments and Analysis

In this research, the impact of data transformation (Sect. 3.2) and control horizon (Sect. 3.3) are studied using data from a 250 MW commercial power plant. Data preprocessing is discussed in (Sect. 3.1).

3.1 Data Preprocessing

The original power plant data consisted of over quarter million instances of 68 parameters collected with a frequency of one minute. Examining the Megawatt-load and continuous boiler efficiency parameters isolated the potential shutdowns and uptimes ranging from days to few months. A derived parameter representing the combined inflow of the fuel (i.e., all feeder speed) was formulated. Domain expert's opinion and univariate analysis (of each parameter) identified outliers, abnormal readings, and sensor errors. Approximately initial five weeks (i.e., 53,280 minutes) of data was used as training and test datasets. The boiler efficiency was discretized into ten-quarter point intervals over the operating range of 86.5% to 87% (Table 1).

The parameters were categorized into controllable/non-controllable (i.e., C or N) and impact/response (i.e., I or R). These parameters constitute three

Table 1. Decision formulation: Boiler efficiency

| Parameter/Measurements | | Boiler Efficiency |
|------------------------|---------------------|---|
| Standard statistics | Mean | 87.28% |
| | Standard deviation | 2.81 |
| | Median | 86.71% |
| | Quantile Q3 | 86.96% |
| | Quantile Q1 | 86.47% |
| Preprocessing | Normal operations | 80%–90% |
| | Max values removed | > 95% |
| | Min values removed | < 75% |
| | Decision categories | LT_86, 86_86.25, 86.25_86.5, 86.5_86.75, 86.75_87, 87_87.25, 87.25_87.5, 87.5_87.75, 87.75_88, HT_88 |

parameter subsets namely, control-impact (i.e., CI), control-impact-response (i.e., CIR), and all parameters (i.e., ALL) (Appendix A) used in data analysis. For example, as condensate flow is directly controlled (C) to adjust boiler operations (I), it was included in the CI and CIR parameter subsets. The ALL parameter subset contain, for example, non-controllable outside air temperature (N), which indirectly affects the combustion process (R). The exclusion of individual feeder speeds from the analysis was due to the inclusion of combine fuel flow parameter (i.e., total input energy).

3.2 Data Transformation

Initially, data transformation with moving average was compared with Haar and db3 wavelets. Due to the software limitation, the training and testing dataset consisted of 5,760 instances, each representing control-impact-response parameters (i.e., CI and CIR parameter subsets). Haar and db3 wavelet algorithms were applied to each of the parameter using Matlab (the code is provided in Appendix B). The analysis of the original against transformed parameters showed moving average as a lagging indicator (i.e., tends to transform signals after the changes have occurred), db3 closely resembling the original signal, and Haar wavelets forming zigzagging step function. Figure 1 illustrates three data transformation schemes for the condensate flow parameter.

A decision-tree algorithm [11, 12] was applied to the transformed datasets and the 10-fold cross-validation [13] was performed. For CI and CIR parameter subsets, the Haar and db3 wavelets provided higher cross-validation accuracy than the moving average (Table 2). The Harr wavelet transformation yielded the highest average cross-validation accuracy of 83.09% and 83.36% for the CI

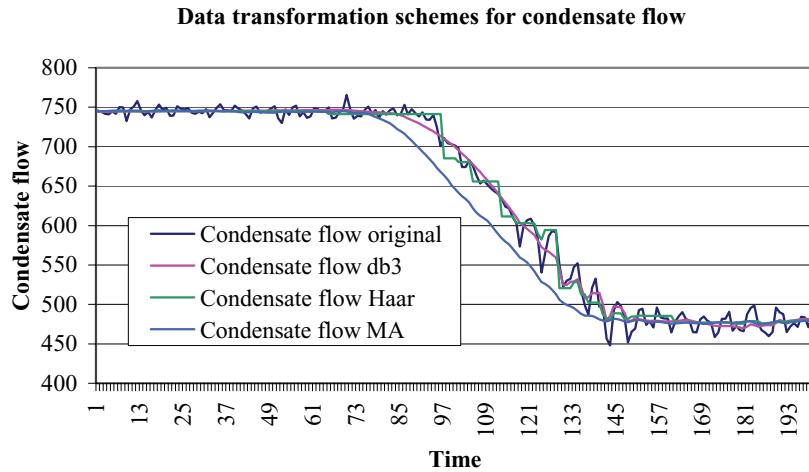


Fig. 1. Data transformation schemes: Condensate flow

Table 2. Comparison of wavelet and moving average transforms

| Boiler Efficiency | | Cross-validation Accuracy | | | Absolute Change | | % Change | |
|-------------------|------------------|---------------------------|--------|--------|-----------------|---------|----------|--------|
| Dataset | Parameter subset | MA | db3 | Haar | db3-MA | Haar-MA | db3 | Haar |
| Dataset 1 | BE_CI | 75.67% | 82.58% | 84.27% | 6.92 | 8.6 | 9.14% | 11.37% |
| | BE_CIR | 77.95% | 81.76% | 84.32% | 3.81 | 6.38 | 4.89% | 8.18% |
| Dataset 2 | BE_CI | 72.05% | 79.57% | 81.91% | 7.52 | 9.87 | 10.44% | 13.70% |
| | BE_CIR | 73.07% | 81.51% | 82.40% | 8.44 | 9.33 | 11.55% | 12.77% |
| Average | BE_CI | 73.86% | 81.08% | 83.09% | 7.22 | 9.235 | 9.79% | 12.54% |
| | BE_CIR | 75.51% | 81.64% | 83.36% | 6.125 | 7.855 | 8.22% | 10.48% |

and CIR parameter subsets, respectively. The absolute and percentage change in cross-validation accuracy between MA and Haar wavelets as well as MA and db3 wavelets is shown in Table 2. The Haar and db3 wavelets produced similar cross-validation accuracy (i.e., statistically equivalent and above 80%) (see Table 2 and Table 3). The Table 3 shows the statistical significance for different data transformation schemes applied to the CI and CIR datasets.

Two datasets with CIR parameters, dataset 1 and dataset 2, were alternatively utilized as the training and testing datasets. The average prediction accuracy for MA was 44.54%, while for Haar and db3 wavelets it was 43.06%, and 40.72%, respectively (Table 4). The range of prediction accuracy was between 37.10% and 51.63%.

The majority of prediction errors were in the neighborhood of the actual labels (Table 5). Due to the measurement error in each of the parameters used in the data analysis, the definition of prediction accuracy has been modified. Rather than using the diagonal elements of the confusion matrix, the sum of the predicted values in the immediate neighborhood of each diagonal element was used, i.e., the total over three categories (Predicted efficiency -.25%, Predicted efficiency, Predicted efficiency +.25%). The actual and modified (neighborhood) prediction accuracy for test dataset 2 is shown in Table 5.

Table 3. *t*-test results for different data transformation schemes

| Transformation Type | CI Parameter Subset | | CIR Parameter Subset | | |
|---------------------|--------------------------|-------------|----------------------|--------------------------|-------------|
| | P($T \leq t$) Two Tail | Significant | Transformation Type | P($T \leq t$) Two Tail | Significant |
| MA vs. db3 | 0.092 | Yes | MA vs. db3 | 0.241 | No |
| MA vs. Haar | 0.051 | Yes | MA vs. Haar | 0.205 | No |
| Db3 vs. Haar | 0.402 | No | db3 vs. Haar | 0.326 | No |

Table 4. Comparison of prediction accuracy for Haar, db3, and moving average

| Test | Actual Prediction Accuracy | | | Neighborhood Prediction Accuracy | | |
|-----------|----------------------------|--------|--------|----------------------------------|--------|--------|
| | db3 | Haar | MA | db3 | Haar | MA |
| Dataset 2 | 44.33% | 41.81% | 51.63% | 92.84% | 84.78% | 89.73% |
| Dataset 1 | 37.10% | 44.31% | 37.45% | 79.46% | 87.15% | 90.99% |
| Average | 40.72% | 43.06% | 44.54% | 86.15% | 85.96% | 90.36% |

Table 5. Modified accuracy for test dataset 2

| WV (db3) | Train Dataset 1 | | | | | | | | | | Test Dataset 2 | | | Prediction Accuracy | |
|-------------------|-----------------|-----------|-----------|------------|-------------|------------|------------|-----------|----------|----------|----------------|-----------------|---------------|---------------------|--|
| | LT_ | 86_ | 86.25 | 86.5_ | 86.75 | 87_ | 87 | 87.25 | 87.5_ | 87.75 | HT | Actual | Neighor- | hood | |
| | 86 | 6.25 | .86.5 | 86.75 | .87 | .25 | .87.5 | 87.75 | .88 | .88 | 88 | | | | |
| LT_86 | 1 | 8 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 1 | 9 | | |
| 86_86.25 | 0 | 11 | 3 | 0 | 3 | 24 | 0 | 0 | 0 | 0 | 0 | 11 | 14 | | |
| 86.25_86.5 | 0 | 9 | 0 | 7 | 8 | 27 | 0 | 0 | 0 | 0 | 0 | 0 | 16 | | |
| 86.5_86.75 | 0 | 7 | 34 | 425 | 522 | 84 | 0 | 0 | 0 | 0 | 0 | 425 | 981 | | |
| 86.75_87 | 0 | 8 | 10 | 765 | 1584 | 636 | 31 | 0 | 0 | 0 | 0 | 1584 | 2985 | | |
| 87_87.25 | 0 | 9 | 0 | 175 | 389 | 400 | 346 | 14 | 0 | 0 | 0 | 400 | 1135 | | |
| 87.25_87.5 | 1 | 1 | 0 | 0 | 0 | 12 | 129 | 54 | 0 | 0 | 129 | 195 | | | |
| 87.5_87.75 | 2 | 0 | 0 | 0 | 0 | 0 | 6 | 1 | 0 | 0 | 1 | 7 | | | |
| 87.75_88 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| HT_88 | 0 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| Correct predicted | | | | | | | | | | | | 2551 | 5342 | | |
| Boiler efficiency | | | | | | | | | | | | Total | 5754 | 5754 | |
| | | | | | | | | | | | | Accuracy | 44.33% | 92.84% | |

The overlapping decision boundaries naturally lead to a fuzzy logic decision-making algorithm. A membership function can be formulated, e.g., the membership function for the decision category 86_86.25 is shown next.

$$\begin{aligned}
 & \mu_{\text{most}}(\text{Boiler efficiency, i.e., BE}) \\
 &= 173 - 2 * \text{BE} \quad \text{if } \text{BE} \leq 86.50 \text{ and } \text{BE} > 86.25 \\
 &= 0.5 \quad \text{if } \text{BE} \leq 86.25 \text{ and } \text{BE} > 86.00 \\
 &= 2 * \text{BE} - 171.5 \quad \text{if } \text{BE} \leq 86.00 \text{ and } \text{BE} > 85.75
 \end{aligned}$$

Similar overlapping fuzzy functions are proposed for the other decision categories (see Fig. 2). Thus to determine the actual value of boiler efficiency in the interval [87.25% and 87.5%], fuzzy logic needs to consider the functions for decision categories 87_87.25, 87.25_87.5, and 87.5_87.75.

Though the cross-validation accuracy was higher for the data transformed using wavelets, the prediction accuracy (actual and neighborhood) was

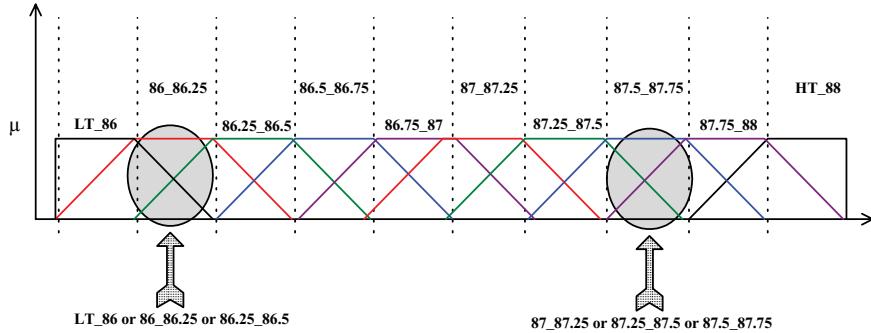


Fig. 2. An example membership function

statistically equivalent for all the data transformation schemes. Thus, any data transformation scheme is suitable for boiler efficiency predictions.

3.3 Control Horizon

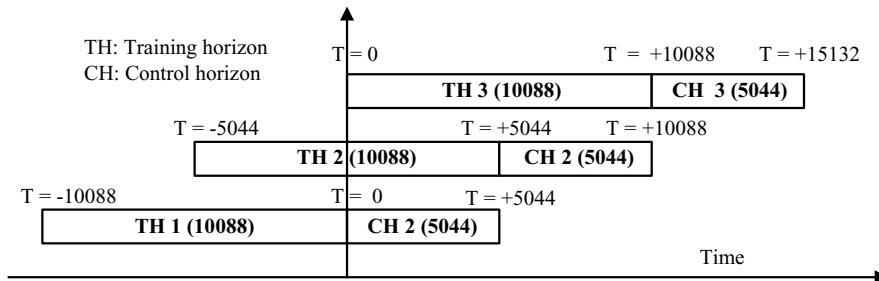
The length of the control horizon determines the refreshing rate of knowledge extracted for the combustion data. Based on prior experiments and the range of operating conditions, a one-week training horizon was chosen. Thus each of the 74 samples consisted of one week of training (ALL parameter subset with a 20-minute MA) data with subsequent control horizons of varying durations (i.e., control horizon of .5 week, .25 week, 1 day, .5 day, 6 hours, and 3 hours) (Table 6). For example, the W0-W1 training dataset in Table 6 includes one week of data starting from day 0, while the .5-week W1-W1.5 control horizon dataset has initiated at the beginning of week 2 and ended in the middle of week 2. The datasets were cascading for .5-week and .25-week control horizons (Fig. 3) while other control horizons were non-overlapping dataset (Table 6). The .5-week control horizon was represented by 9 samples while other control horizons were of 13 samples each.

The cross-validation and prediction accuracy for .5-week control horizon (9 samples) is presented in Fig. 4. Although the overall average cross-validation accuracy was higher (82.27% with standard deviation of 2.27), the prediction accuracy was moderate (56.56% with standard deviation of 10.65). The higher standard deviation for prediction accuracy is due to the dynamic nature of the combustion process. The prediction accuracy for .5-week control horizon is statistically different than the 10 fold cross-validation accuracy with $t = 2.160$ and $p = 0.00059$. The most misclassifications for the nine samples (.5-week control horizon) occurred in the neighborhood of each decision category. The average modified accuracy (in the neighborhood) from cross-validation and prediction for the .5-week control horizon was 99.42% and 92.87%, respectively. Removing sample 7 (potential outlier), the standard deviation for modified prediction accuracy was 3.5.

Table 6. Training and control horizons

| Training Horizon | | | Control Horizon | | | | | |
|------------------|-------|-------|-----------------|----------------|-------------|--------------|---------------------|---------------|
| Start | End | Start | End (.5 week) | End (.25 week) | End (1 day) | End (.5 day) | End (6 hours) | End (3 hours) |
| W0 | W1 | W1 | W1.5 | W1.25 | W1.14 | W1.07 | W1.04 | W1.02 |
| W0.25 | W1.25 | W1.25 | – | W1.5 | W1.39 | W1.32 | W1.29 | W1.27 |
| W0.5 | W1.5 | W1.5 | W2 | W1.75 | W1.64 | W1.57 | W1.54 | W1.52 |
| W0.75 | W1.75 | W1.75 | – | W2 | W1.89 | W1.82 | W1.79 | W1.77 |
| W1 | W2 | W2 | W2.5 | W2.25 | W2.14 | W2.07 | W2.04 | W2.02 |
| W1.25 | W2.25 | W2.25 | – | W2.5 | W2.39 | W2.32 | W2.29 | W2.27 |
| W1.5 | W2.5 | W2.5 | W3 | W2.75 | W2.64 | W2.57 | W2.54 | W2.52 |
| W1.75 | W2.75 | W2.75 | – | W3 | W2.89 | W2.82 | W2.79 | W2.77 |
| W2 | W3 | W3 | W3.5 | W3.25 | W3.14 | W3.07 | W3.04 | W3.02 |
| W2.25 | W3.25 | W3.25 | – | W3.5 | W3.39 | W3.32 | W3.29 | W3.27 |
| W2.5 | W3.5 | W3.5 | W4 | W3.75 | W3.64 | W3.57 | W3.54 | W3.52 |
| W2.75 | W3.75 | W3.75 | – | W4 | W3.89 | W3.82 | W3.79 | W3.77 |
| W3 | W4 | W4 | W4.5 | W4.25 | W4.14 | W4.07 | W4.04 | W4.02 |
| W3.5 | W4.5 | W4.5 | W5 | | | | | |
| | | | | | | | Total of 74 samples | |
| W4 | W5 | W5 | W5.5 | | | | | |

A similar procedure applied to other control horizons, .25 week, 1 day, .5 day, 6 hours, and 3 hours. Figure 5 provides the average accuracy and the corresponding standard deviations. A standard *t* test applied to each pair of control horizons showed no statistical difference (i.e., $p > 0.05$) (Table 7). Thus either of them is a candidate for the ideal control horizon. Based on the domain requirements and additional criterion of threshold drop in prediction accuracy, the ideal control horizon can be either the longest horizon of .5-week or the highest prediction accuracy control horizon (with lower variation) of 3 hours. The ensemble of the two models (based on .5-week and 3 hours control horizons) can be implemented for boiler efficiency prediction.

**Fig. 3.** Cascading training and control horizons

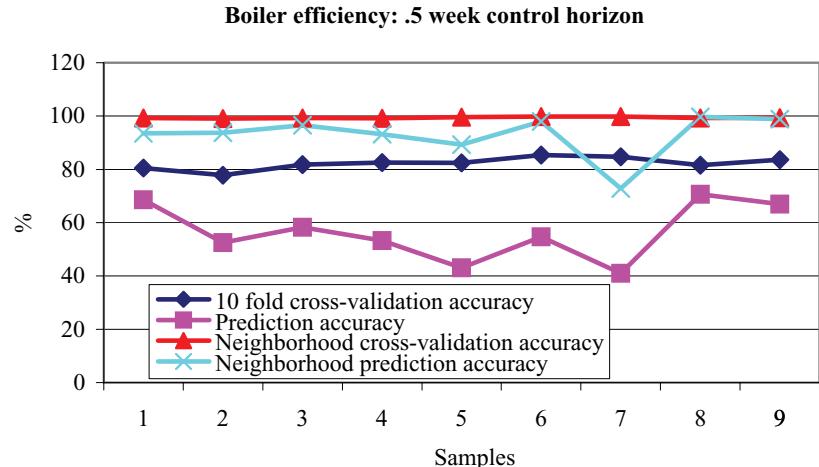


Fig. 4. Boiler efficiency accuracy for .5-week control horizon

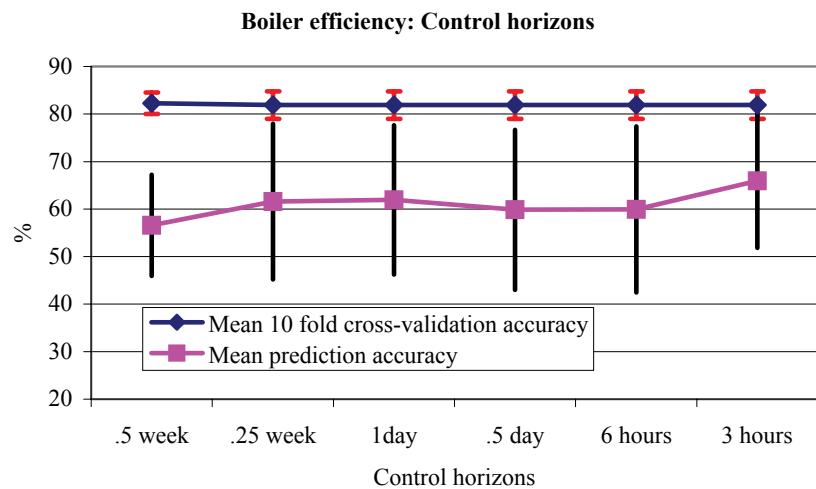


Fig. 5. Boiler efficiency accuracy for various control horizons

4 Conclusions

In this chapter, three data transformation schemes (moving average, Haar and Daubechies [db3] wavelets) were successfully applied to predict boiler efficiency. In general, the data transformed by wavelets resulted in better cross-validation accuracy than moving average. However, the prediction accuracy was statistically similar for all data transformation schemes. The length of control horizon that is acceptable for combustion process control was determined. The prediction accuracy was statistically lower than the cross-validation

Table 7. *t*-test comparisons

| t-test between each control horizon (p values reported) | | | | | | |
|---|---------------|----------|--------|--------|---------|---------------|
| | .5 week | .25 week | 1 day | .5 day | 6 hours | 3 hours |
| .5 week | – | 0.1656 | 0.1488 | 0.2397 | 0.2427 | 0.0534 |
| .25 week | 0.1656 | – | 0.9516 | 0.7939 | 0.8052 | 0.4688 |
| 1 day | 0.1488 | 0.9516 | – | 0.7445 | 0.7568 | 0.4986 |
| .5 day | 0.2397 | 0.7939 | 0.7445 | – | 0.9925 | 0.3247 |
| 6 hours | 0.2427 | 0.8052 | 0.7568 | 0.9925 | – | 0.3402 |
| 3 hours | 0.0534 | 0.4688 | 0.4986 | 0.3247 | 0.3402 | – |

accuracy for all control horizons. A new prediction accuracy measure accounting for predictions in the neighborhood of a correctly classified value was proposed. To better handle the uncertain nature of the parameters and the decision (efficiency) a fuzzy logic approach was proposed. The research presented in this chapter provides a foundation for development of robust predictive models.

Acknowledgement

Our special thanks to Kaiser Kevin, Bries Michael, Heiken Ryan, Ridnour Mitchell, Cichella Tim, Herring Anthony, Macias Mashala, Anderson Derek, Anderson Jeffrey, and Watson Jonathan for assisting in the research experiments.

References

1. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, Cambridge, MA, 1995 [283](#)
2. Kusiak, A., “Feature Transformation Methods in Data Mining”, IEEE Transactions on Electronics Packaging Manufacturing, Vol. 24, No. 3, pp. 214–221, 2001 [284](#)
3. Weigend, A., Chen, F., Figlewski, S., and Waterhouse, S.R., “Discovering Technical Trades in the T-Bond Futures Market”, In: Argawal, R., Stolorz, P., Piatetsky-Shapiro, G. (Eds.): Proc. Fourth Int. Conf. Knowledge Discovery and Data Mining, AAAI Press, New York, pp. 354–358, 1998 [284](#)
4. Burns, A., Kusiak, A., and Letsche, T., “Mining Transformed Data Sets”, in M. Gh. Negoita, R.J. Howlett, and L.C. Jain (Eds), Knowledge-Based Intelligent Information and Engineering Systems, LNAI 3213, Vol. I, Springer, Heidelberg, Germany, pp. 148–154, 2004 [284](#)
5. Li, T., Li, Q., Zhu, S., and Ogihara, M., “A Survey on Wavelet Applications in Data Mining”, SIGKDD Explorations, Vol. 4, No. 2, pp. 49–68, 2002 [284](#)

6. Graps, A., "An Introduction to Wavelets", IEEE Computational Sciences and Engineering, Vol. 2, No 2, pp. 50–61, 1995 [284](#)
7. Bratteli, O. and Jorgensen, P., Wavelets Through a Looking Glass, Birkhauser, Boston, MA, 2002 [284](#)
8. Daubechies, I., "Orthonormal bases of compactly supported wavelets", Communications on Pure and Applied Mathematics, Vol. 41, pp. 909–996, 1988 [284](#)
9. Aboufadel, E. and Schlicker, S., Discovering wavelets, John Wiley and Sons, Inc, New York, NY, 1999 [284](#)
10. Sang, A. and Li, S., "Predictability analysis of network traffic", Proceedings - IEEE INFOCOM, Vol. 1, pp 342–351, 2000 [284](#)
11. Quinlan, R., C 4.5 Programs for ml, San Meteo, Morgan Kaufmann, CA, 1992 [286](#)
12. Frank, E. and Witten, I.H., "Generating Accurate Rule Sets Without Global Optimization", in Shavlik, J., Eds., Machine Learning: Proceedings of the Fifteenth International Conference, Morgan Kaufmann Publishers, 1998 [286](#)
13. Stone, M., "Cross-validatory choice and assessment of statistical predictions", Journal of the Royal Statistical Society, Vol. 36, pp.111–147, 1974 [286](#)

Appendix A: Categories of Parameters and Parameter Subsets

| Parameter | Type | CI | CIR | ALL |
|------------------------------------|---------|-------|-------|-------|
| Air_Temp_From_PReheat_Coil_1 | C and I | X | X | X |
| Air_Temp_From_PReheat_Coil_2 | C and I | X | X | X |
| Air_HTR_9A_Gas_Out_Pressure | C and I | X | X | X |
| Air_HTR_9B_Gas_Out_Pressure | C and I | X | X | X |
| Condensate_Flow | C and I | X | X | X |
| Selected_9_BLR_FeedWater_F | C and I | X | X | X |
| RHTR_ATTemp_Water_Flow | C and I | X | X | X |
| SHTR_ATTemp_Water_Flow | C and I | X | X | X |
| BLR_FDW_Header_Pressure | C and I | X | X | X |
| Burner_TILT_Degress | C and I | X | X | X |
| Feeder_9ALL_Speed | C and I | X | X | X |
| Air_HTR_9A_Outlet_Pressure | C and R | | X | X |
| Air_PREHTR_Coil_9A_Outlet_Pressure | C and R | | X | X |
| Air_HTR_9B_Outlet_Pressure | C and R | | X | X |
| Air_PREHTR_Coil_9B_Outlet_Pressure | C and R | | X | X |
| ECON_Outlet_Pressure | C and R | | X | X |
| BLR_9_O2 | C and R | | X | X |
| HOT_Reheat_Steam_Temp | C and R | | X | X |
| Main_Steam_Temp | C and R | | X | X |
| R_5_Net_Megawatt_Load | C and R | | X | X |
| NO_9_Boiler_Steam_Flow | C and R | | X | X |
| Net_Unit_Heat_Rate_Actual | C and R | | X | |
| Main_Steam_Pressure_control | C and R | | X | X |
| 1ST_Stage_Shell_Pressure | C and R | | X | X |
| BE_D2 | C and R | X | X | X |
| Outside_Air_Temp | N and R | | | X |
| Circulating_Water_Inlet_Temp | N and R | | | X |
| Air_Temp_From_Air_HTR_9A | N and R | | | X |
| Air_Temp_From_Air_HTR_9B | N and R | | | X |
| | | | | |
| | | | | |
| FWH_F_TTD | N and R | | | X |
| Hot_RH_Pressure_from_Boiler | N and R | | | X |

Appendix B: Matlab Code

```

data = "Week1feederspeed1.txt"
sig = data;
[thr, sorh, keepapp] = ddencmp ("den", "wv", sig);
sigd = wdencmp("gbl", sig, "db3", 5, thr, sorh, keepapp);
save("sigd", "sigd", "-ascii");
subplot(1,2,1); plot (sig); subplot(1,2,2); plot(sigd);

```

where:

sig – represents the signal from the Excel data;
 sigd – represents the de-noised signal;
 thr – represents a threshold level, correlates to the wavelet coefficients;
 sorh – determines soft or hard thresholding;
 keepapp – allows user to keep the approximation coefficients;
 ddencmp – Matlab function that supplies the default values for de-nosing or compression, reads in the values “den” signifying denoising, “wv” signifying the use of wavelets for denoising, and reads in the signal “sig”;
 wdencmp – Matlab function used explicitly for wavelet de-noising purposes, reads in a global thresholding option “gbl”, the signal, the specific wavelet at a refinement level, and the default values returned from “ddencmp”.