



IntechOpen

Machine Learning

Edited by Yagang Zhang



MACHINE LEARNING

Edited by
YAGANG ZHANG

Machine Learning

<http://dx.doi.org/10.5772/217>

Edited by Yagang Zhang

© The Editor(s) and the Author(s) 2010

The moral rights of the and the author(s) have been asserted.

All rights to the book as a whole are reserved by INTECH. The book as a whole (compilation) cannot be reproduced, distributed or used for commercial or non-commercial purposes without INTECH's written permission.

Enquiries concerning the use of the book should be directed to INTECH rights and permissions department (permissions@intechopen.com).

Violations are liable to prosecution under the governing Copyright Law.



Individual chapters of this publication are distributed under the terms of the Creative Commons Attribution 3.0 Unported License which permits commercial use, distribution and reproduction of the individual chapters, provided the original author(s) and source publication are appropriately acknowledged. If so indicated, certain images may not be included under the Creative Commons license. In such cases users will need to obtain permission from the license holder to reproduce the material. More details and guidelines concerning content reuse and adaptation can be found at <http://www.intechopen.com/copyright-policy.html>.

Notice

Statements and opinions expressed in the chapters are those of the individual contributors and not necessarily those of the editors or publisher. No responsibility is accepted for the accuracy of information contained in the published chapters. The publisher assumes no responsibility for any damage or injury to persons or property arising out of the use of any materials, instructions, methods or ideas contained in the book.

First published in Croatia, 2010 by INTECH d.o.o.

eBook (PDF) Published by IN TECH d.o.o.

Place and year of publication of eBook (PDF): Rijeka, 2019.

IntechOpen is the global imprint of IN TECH d.o.o.

Printed in Croatia

Legal deposit, Croatia: National and University Library in Zagreb

Additional hard and PDF copies can be obtained from orders@intechopen.com

Machine Learning

Edited by Yagang Zhang

p. cm.

ISBN 978-953-307-033-9

eBook (PDF) ISBN 978-953-51-5899-8

We are IntechOpen, the world's leading publisher of Open Access books

Built by scientists, for scientists

4,200+

Open access books available

116,000+

International authors and editors

125M+

Downloads

151

Countries delivered to

Top 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.

For more information visit www.intechopen.com



Preface

The goal of this book is to present the key algorithms, theory and applications that from the core of machine learning. Learning is a fundamental activity. It is the process of constructing a model from complex world. And it is also the prerequisite for the performance of any new activity and, later, for the improvement in this performance. Machine learning is concerned with constructing computer programs that automatically improve with experience. It draws on concepts and results from many fields, including artificial intelligence, statistics, control theory, cognitive science, information theory, etc. The field of machine learning is developing rapidly both in theory and applications in recent years, and machine learning has been applied to successfully solve a lot of real-world problems.

Machine learning theory attempts to answer questions such as "How does learning performance vary with the number of training examples presented?" and "Which learning algorithms are most appropriate for various types of learning tasks?" Machine learning methods are extremely useful in recognizing patterns in large datasets and making predictions based on these patterns when presented with new data. A variety of machine learning methods have been developed since the emergence of artificial intelligence research in the early 20th century. These methods involve the application of one or more automated algorithms to a body of data. There are various methods developed to evaluate the effectiveness of machine learning methods, and those methods can be easily extended to evaluate the utility of different machine learning attributes as well.

Machine learning techniques have the potential of alleviating the complexity of knowledge acquisition. This book presents today's state and development tendencies of machine learning. It is a multi-author book. Taking into account the large amount of knowledge about machine learning and practice presented in the book, it is divided into three major parts: Introduction, Machine Learning Theory and Applications. Part I focuses on the Introduction of machine learning. The author also attempts to promote a new thinking machines design and development philosophy. Considering the growing complexity and serious difficulties of information processing in machine learning, in Part II of the book, the theoretical foundations of machine learning are considered, mainly include self-organizing maps (SOMs), clustering, artificial neural networks, nonlinear control, fuzzy system and knowledge-based system (KBS).Part III contains selected applications of various machine learning approaches, from flight delays, network intrusion, immune system, ship design to CT, RNA target prediction, and so on.

The book will be of interest to industrial engineers and scientists as well as academics who wish to pursue machine learning. The book is intended for both graduate and postgraduate students in fields such as computer science, cybernetics, system sciences, engineering, statistics, and social sciences, and as a reference for software professionals and practitioners. The wide scope of the book provides them with a good introduction to many basic approaches of machine learning, and it is also the source of useful bibliographical information.

Editor:

Yagang Zhang

Contents

Preface	VII
PART I INTRODUCTION	
1. Machine Learning: When and Where the Horses Went Astray? Emanuel Diamant	001
PART II LEARNING THEORY	
2. SOMs for machine learning Iren Valova, Derek Beaton and Daniel MacLean	019
3. Relational Analysis for Clustering Consensus Mustapha Lebbah, Younès Bennani, Nistor Grozavu and Hamid Benhadda	045
4. A Classifier Fusion System with Verification Module for Improving Recognition Reliability Ping Zhang	061
5. Watermarking Representation for Adaptive Image Classification with Radial Basis Function Network Chi-Man Pun	077
6. Recent advances in Neural Networks Structural Risk Minimization based on multiobjective complexity control algorithms D.A.G. Vieira, J.A. Vasconcelos and R.R. Saldanha	091
7. Statistics Character and Complexity in Nonlinear Systems Yagang Zhang and Zengping Wang	109
8. Adaptive Basis Function Construction: An Approach for Adaptive Building of Sparse Polynomial Regression Models Gints Jekabsons	127
9. On The Combination of Feature and Instance Selection Jerffeson Teixeira de Souza, Rafael Augusto Ferreira do Carmo and Gustavo Augusto Campos de Lima	157
10. Fuzzy System with Positive and Negative Rules Thanh Minh Nguyen and Q. M. Jonathan Wu	173

11. Automatic Construction of Knowledge-Based System using Knowware System 189
 Sio-Long Lo and Liya Ding
12. Applying Fuzzy Bayesian Maximum Entropy to Extrapolating Deterioration in Repairable Systems 217
 Chi-Chang Chang, Ruey-Shin Chen and Pei-Ran Sun

PART III APPLICATIONS

13. Alarming Large Scale of Flight Delays: an Application of Machine Learning 239
 Zonglei Lu
14. Machine Learning Tools for Geomorphic Mapping of Planetary Surfaces 251
 Tomasz F. Stepinski and Ricardo Vilalta
15. Network Intrusion Detection using Machine Learning and Voting techniques 267
 Tich Phuoc Tran, Pohsiang Tsai, Tony Jan and Xiaoying Kong
16. Artificial Immune Network: Classification on Heterogeneous Data 291
 Mazidah Puteh, Abdul Razak Hamdan, Khairuddin Omar
 and Mohd Tajul Hasnan Mohd Tajuddin
17. Modified Cascade Correlation Neural Network and its Applications to Multidisciplinary Analysis Design and Optimization in Ship Design 301
 Adeline Schmitz, Frederick Courouble, Hamid Hefazi and Eric Besnard
18. Massive-Training Artificial Neural Networks (MTANN) in Computer-Aided Detection of Colorectal Polyps and Lung Nodules in CT 343
 Kenji Suzuki, Ph.D.
19. Automated detection and analysis of particle beams in laser-plasma accelerator simulations 367
 Daniela M. Ushizima, Cameron G. Geddes, Estelle Cormier-Michel, E.Wes Bethel,
 Janet Jacobsen, Prabhat, Oliver R ubel, GuntherWeber, Bernd Hamann,
 Peter Messmer and Hans Haggen
20. Specificity Enhancement in microRNA Target Prediction through Knowledge Discovery 391
 Yanju Zhang, Jeroen S. de Bruin and Fons J. Verbeek
21. Extraction Of Meaningful Rules In A Medical Database 411
 Sang C. Suh, Nagendra B. Pabbisetty and Sri G. Anaparthi
22. Establishing and retrieving domain knowledge from semi-structural corpora 427
 Hsien-chang WANG, Pei-chin YANG and Chen-chieh LI

Machine Learning: When and Where the Horses Went Astray?

Emanuel Diamant
VIDIA-mant
Israel

1. Introduction

The year of 2006 was exceptionally cruel to me – almost all of my papers submitted for that year conferences have been rejected. Not “just rejected” – unduly strong rejected. Reviewers of the ECCV (European Conference on Computer Vision) have been especially harsh: “This is a philosophical paper... However, ECCV neither has the tradition nor the forum to present such papers. Sorry...” O my Lord, how such an injustice can be tolerated in this world? However, on the other hand, it can be easily understood why those people hold their grudges against me: Yes, indeed, I always try to take a philosophical stand in all my doings: in thinking, paper writing, problem solving, and so no. In my view, philosophy is not a swear-word. Philosophy is a keen attempt to approach the problem from a more general standpoint, to see the problem from a wider perspective, and to yield, in such a way, a better comprehension of the problem’s specificity and its interaction with other world realities. Otherwise we are doomed to plunge into the chasm of modern alchemy – to sink in partial, task-oriented determinations and restricted solution-space explorations prone to dead-ends and local traps.

It is for this reason that when I started to write about “Machine Learning”, I first went to the Wikipedia to inquire: What is the best definition of the subject? “Machine Learning is a subfield of Artificial Intelligence” – was the Wikipedia’s prompt answer. Okay. If so, then: “What is Artificial Intelligence?” – “Artificial Intelligence is the intelligence of machines and the branch of computer science which aims to create it” – was the response. Very well. Now, the next natural question is: “What is Machine Intelligence?” At this point, the kindness of Wikipedia has been exhausted and I was thrown back, again to the Artificial Intelligence definition. It was embarrassing how quickly my quest had entered into a loop – a little bit confusing situation for a stubborn philosopher.

Attempts to capitalize on other trustworthy sources were not much more productive (Wang, 2006; Legg & Hutter, 2007). For example, Hutter in his manuscript (Legg & Hutter, 2007) provides a list of 70-odd “Machine Intelligence” definitons. There is no consensus among the items on the list – everyone (and the citations were chosen from the works of the most prominent scholars currently active in the field), everyone has his own particular view on the subject. Such inconsistency and multiplicity of definitions is an unmistakable sign of

philosophical immaturity and a lack of a will to keep the needed grade of universality and generalization.

It goes without saying, that the stumbling-block of the Hutter's list of definitions (Legg & Hutter, 2007) is not the adjectives that were used- after all the terms "Artificial" and "Machine" are consensually close in their meaning and therefore are commonly used interchangeably. The real problem - is the elusive and indefinable term „Intelligence".

I will not try the readers' patience and will not tediously explain how and why I had arrived at my own definition of the matters that I intend to scrutinize in this paper.

I hope that my philosophical leanings will be generously excused and the benevolent readers will kindly accept the unusual (reverse) layout of the paper's topics. For the reasons that would be explained in a little while, the main and the most general paper's idea will be presented first while its compiling details and components will be exposed (in a descending order) afterwards. And that is how the proposed paper's layout should look like:

- **Intelligence is the system's ability to process information.** This statement is true both for all biological natural systems as for artificial, human-made systems. By "information processing" we do not mean its simplest forms like information storage and retrieval, information exchange and communication. What we have in mind are the high-level information processing abilities like information analysis and interpretation, structure patterns recognition and the system's capacity to make decisions and to plan its own behavior.
- **Information in this case should be defined as a description** - A language and/or an alphabet-based description, which results in a reliable reconstruction of an original object (or an event) when such a description is carried out, like an execution of a computer program.
- **Generally, two kinds of information must be distinguished:** Objective (physical) information and subjective (semantic) information. By physical information we mean the description of data structures that are discernable in a data set. By semantic information we mean the description of the relationships that may exist between the physical structures of a given data set.
- **Machine Learning** is defined as the best means for appropriate information retrieval. Its usage is endorsed by the following fundamental assumptions: 1) Structures can be revealed by their characteristic features, 2) Feature aggregation and generalization can be achieved in a bottom-up manner where final results are compiled from the component details, 3) Rules, guiding the process of such compilation, could be learned from the data itself.
- **All these assumptions validating Machine Learning applications are false.** (Further elaboration of the theme will be given later in the text). Meanwhile the following considerations may suffice:
 - Physical information, being a natural property of the data, can be extracted instantly from the data, and any special rules for such task accomplishment are not needed. Therefore, Machine Learning techniques are irrelevant for the purposes of physical information retrieval.
 - Unlike physical information, semantics is not a property of the data. Semantics is a property of an external observer that watches and scrutinizes the data. Semantics is assigned to physical data structures, and therefore it can not be learned straightforwardly from the data. For this reason, Machine Learning techniques are

useless and not applicable for the purposes of semantic information extraction. Semantics is a shared convention, a mutual agreement between the members of a particular group of viewers or users. Its assignment has to be done on the basis of a consensus knowledge that is shared among the group members, and which an artificial semantic-processing system has to possess at its disposal. Accommodation and fitting of this knowledge presumes availability of a different and usually overlooked special learning technique, which would be best defined as **Machine Teaching** - a technique that would facilitate externally-prepared-knowledge transfer to the system's disposal.

These are the topics that I am interested to discuss in this paper. Obviously, the reverse order proposed above, will never be reified - there are paper organization rules and requirements, which none never will be allowed to override. They must be, thus, reverently obeyed. And I earnestly promise to do this (or at least to try to do this) in this paper.

2. When the State of the Art is Irrelevant

One of the commonly accepted rules prescribes that the Introduction Section has to be succeeded by a clear presentation of a following subject: What is the State of the Art in the field and what is the related work done by the other researchers? Unfortunately, I'm unable to meet this requirement, because (to the best of my knowledge) there is no relevant work in the field that can be used for this purpose. Or, let us put this in a more polite way: The work presented in this paper is so different from other mainstream approaches that it would be unwise to compare it with the rest of the work in the field and to discuss arguments in favour or against their endless disagreements and discrepancies. However, to avoid any possible allegations in disrespectfulness, I would like to provide here some reflections on the departure points of my work, which (I hope) are common to many friends and foes in the domain.

My first steps in the field were inspired by David Marr's ideas about the "Primal" and the "Two-and-a-half" image representation sketch, which is carrying out the information content of an image (Marr, 1978; Marr, 1982). Image understanding was always the most relevant and the most palpable manifestation of human intelligence, and so, those who are busy with intelligence replications in machines, are due to cope with image understanding and image processing issues.

"You see, - had I proudly agitated my employers, trying to convince them to fund my image-processing enterprises, - meagre lines of a painter's caricature provide you with all information needed to comprehend the painter's intention and to easily recognise the objects drawn in the picture. Edges are the information bearers! Edge exploration and processing will help us to reach advances in pattern recognition and image understanding." My employers were skeptic and penny-pinching, but nevertheless, I was allowed to do some work. However, very soon it had become clear that my problems are far from being information retrieval issues - my real problem was to run (approximately in a real-time fashion) a 3-by-3 (or 5-by-5) operator over a 256-by-256 pixel image. And only then, when the run is somehow successfully completed, I was doomed to find myself inflated with a multitude of edges: cracked, disjoint, and inconsistent. On one hand, an entire spectrum of dissimilar edge pieces, and on the other hand - a striking deficit of any hints regarding how to arrange them into something handy and meaningful. At least, to choose among them (to

discriminate, to segment, to threshold) those that would be suitable for further processing. Even though, it was at all not sure that anybody knows what such a processing should be. It was not only my nightmare. Many people have swamped in this bog. Many are still trying to tempt the fate – even today, the flow of edge extraction and segmentation publications does not dry up, and new machine learning techniques are reportedly proposed to cure the problem (Ghosh et al., 2007; Awad & Man, 2008; Qiu & Sun, 2009).

Human vision physiology studies, which have been always seen as an endless source of computer vision R&D inspiration, have also proved to be of a little help here. Treisman's feature-integration theory (Treisman & Gelade, 1980) and Biederman's recognition-by-components theory (Biederman, 1987) – the cornerstones of contemporary vision science – were fitting well the bottom-up image processing philosophy, (where initial feature gathering is followed by further feature consolidation), but they have nothing to say about how this feature aggregation and integration (into meaningful perceptible objects) has to be realized. They only say that this process has to be done in a top-down fashion, in opposite to the bottom-up processing of the initial features.

To overcome the problem, a great variety of so-called "binding" theories have been proposed (Treisman, 1996; Treisman, 2003). However, all of them turned out as inappropriate. In a desperate attempt to resolve this irresolvable contradiction, even a theory of a mysterious homunculus has been proposed – a "little man inside the head" that perceives the world through our senses and then unmistakably fulfills all the needed (intelligent) actions (Crick & Koch, 2000). But the theory of the homunculus has not taken roots. Human level intelligence has been and continues to be a challenge, and nothing in the field has changed since the 50s of the past century, when the first steps of Artificial Intelligence exploration have been carried out (Turing, 1950; McCarthy et al., 1955).

3. In Search for a Better Fortune

I am not trying to claim that I am more clever or wise than others. All the stupid things that others have persistently tried to do, I have repeatedly tried as well. But in one thing, however, I was certainly different from the others – I have never neglected my final goal: To grasp the information content of an image. Together with other image-processing "partisans" and "camarados" I fought my pixel-oriented battles, but a dream about object-oriented image processing was always blooming in my heart.

As you can understand, nothing worthy had come out from that. Nevertheless, some of the things that I was lucky to make happen (at that time) are worth to be mentioned here. For example, I have invented a notion of "Single Pixel Information Content" and a way for its quantitative evaluation (Diamant, 2003). I have also invented a notion of "Specific Information Density of an Image", and, relying on the pixel's information content (measure), I have attempted to investigate the effect of "Image Information Content Conservation". That is, when an image scale is successively reduced, Image Specific Information Density remains unchanged (or even slightly grows up). Then, at some level of reduction, it rapidly declines. This scale, actually the scale one step preceding the drop of Information Density, I thought, should be the most advantageous (scale) to start image information content explorations.

A paper, containing quantitative results and a proof of this idea, has been submitted to the British Machine Vision Conference (Diamant, 2002), but, (as usually), was decisively

rejected. Never mind, these investigations have led to an important insight that image information content excavation has to be commenced at an optimal, low-dimensional image representation scale.

I am proud to inform the interested readers that similar investigations have been performed recently (and similar results have been attained) by MIT researchers (Torralba, 2009). However, that was done about seven years later, and only in qualitative experiments conducted on human participants (but not as a quantitative work).

Never mind, the idea of initial low-dimensional image exploration was in some way consistent with a naïve psychological vision conjecture about how humans look at a scene. Since biological vision research was always busy with only foveated vision studies, one principal aspect of human vision was always remained neglected: How does the brain know where to look in a scene? We do not search our field of view in a regular, raster-scan manner. On the contrary, we do this in an unpredictable, but certainly a not-random manner (Koch et al., 2007; Shomstein & Behrmann, 2008). If so, how does the brain know where to place the eye's fovea – (the main means for visual information gathering) – before it knows in advance where such information is to be found? Certainly, the brain must have a prior knowledge about the scene layout, about the general map of a scene. Certainly, the scale of this map must be several orders lower than the fovea resolution scale, and it is clear that these information gathering maps are being used simultaneously and interchangeably.

Such considerations have inevitably led us to a conclusion that other theories, currently unknown to us, which would be capable of explaining such multiscale brain performance have to be urgently searched for. Indeed, very soon I came upon an appropriate theory. And even not a single one, but a whole bundle of theories.

In the middle of the 60s of the previous century, three almost simultaneous, but absolutely independently developed, theories have sprung up: Solomonoff's theory of Inference (Solomonoff, 1964), Kolmogorov's Complexity theory (Kolmogorov, 1965), and Chaitin's Algorithmic Information theory (Chaitin, 1966). Since among the three, Kolmogorov's theory is the most known one, I will first and mainly refer to it in our further discussion.

Just as Shannon's Information theory (Shannon, 1948) published almost 20 years ahead, Kolmogorov's theory was aimed at providing means for measuring "information". However, while Shannon's theory was dealing only with the average amount of information conveyed by an outcome of a random source, Kolmogorov's theory was busy with information contained in a particular isolated object. Thus, Kolmogorov's theory was far more suitable to deal with human vision peculiarities.

However, I do not intend to bother the readers with explanations about Kolmogorov's theory merits. Such expanded enlightenment could be found else where, for example (Li & Vitanyi, 2008; Grunvald & Vitanyi, 2008). My humble intention is, relying on the insights of the Kolmogorov's theory, to provide some useful illuminations, which can be deduced from the theory and applied to the practice of image information content excavation.

An essential part of my work has been already done in the past years, and has been even published on several occasions (Diamant, 2004; Diamant, 2005a; Diamant, 2005b). (The publications could be easily found at some freely accessible web repositories, like CiteSeer, Eprintweb, ArXiv, etc. And also on my personal web site: <http://www.vidia-mant.info>). However, for the consistency of our discussion, I would like to repeat here the main points of these previous publications.

The key point is that **information is a description**, a certain alphabet-based or language-based description, which Kolmogorov's theory regards as a program that, being executed, trustworthy reproduces the original object (Vitanyi, 2006). In an image, such objects are visible data structures from which an image is comprised of. So, a set of reproducible descriptions of image data structures is the information contained in an image.

The Kolmogorov's theory prescribes the way in which such descriptions must be created: At first, the most simplified and generalized structure must be described. Recall the Occam's Razor principle: Among all hypotheses consistent with the observation choose the simplest one that is coherent with the data, (Sadrzadeh, 2008). Then, as the level of generalization is gradually decreased, more and more fine-grained image details (structures) become revealed and depicted. This is the second important point, which follows from the theory's pure mathematical considerations: Image **information is a hierarchy of decreasing level descriptions** of information details, which unfolds in a coarse-to-fine top-down manner. (Attention, please! Any bottom-up processing is not mentioned here! There is no low-level feature gathering and no feature binding!!! The only proper way for image information elicitation is a top-down coarse-to-fine way of image processing!)

The third prominent point, which immediately pops-up from the two just mentioned above, is that the top-down manner of image **information elicitation does not require incorporation of any high-level knowledge** for its successful accomplishment. It is totally free from any high-level guiding rules and inspirations. (The homunculus have lost his job and is finally fired).

That is why I call the information, which unconditionally can be found in an image, – the **Physical Information**. That is, information that reflects objective (physical) structures in an image and is totally independent of any high level interpretation of the interrelations between them.

What immediately follows from this is that high-level image semantics is not an integrated part of image information content (as it is traditionally assumed). It cannot be seen more as a natural property of an image. **Semantic Information**, therefore, must be seen as a property of a human observer that watches and scrutinizes an image. That is why we can say now: **Semantics is assigned to an image by a human observer**. That is strongly at variance with the contemporary views on the concept of semantic information.

As it was mentioned above, I have no intention to argue with the mainstream experts, conference chairs, keynotes speakers and invited talks presenters about the validity of my contemplations, about my philosophical inclinations or research duties and preferences. These respected gentlemen would continue to teach you **how to extract semantic information from an image or how it should be derived from low-level information features**.

(I do not provide here examples of such claims. I hope, the readers are well enough acquainted with the state of the art in the field and its mainstream developments, to be able to recall the appropriate cases by themselves. I also hope that readers of this paper are ready to change their minds – fifty or so years of Machine Learning triumphal marching in the head of the Artificial Intelligence parade have not got us closer to the desired goal of Intelligent Machines deployment and use. Partially and loosely defined (or it would be right to say, undefined) departure points of this enterprise were the main reasons responsible for this years-long wandering in the desert far away from the promised land.)

4. “Repetitio est Mater Studiorum”

(For those who are not fluent enough in Latin, the translation of this proverb would be: Reiteration is the mother of learning). Okay, I am really sorry that instead of dealing with the declared subject of this paper (that is, Machine Learning and all its corresponding issues), I have to return again and again to topics that have been already discussed in the past and even published at some previous occasions. (But that is the bad luck of an image-processing partisan). Therefore, with all apologies to be due, I will continue our discourse with some extended citations seized from my previously published papers.

4.1 Image Physical information Processing

The first citation is related to physical information processing issues and is taken from a five years old paper (Diamant, 2004). The citation subject is – an algorithmic implementation of image physical information mining principles.

The algorithm's block-scheme looks as follows:

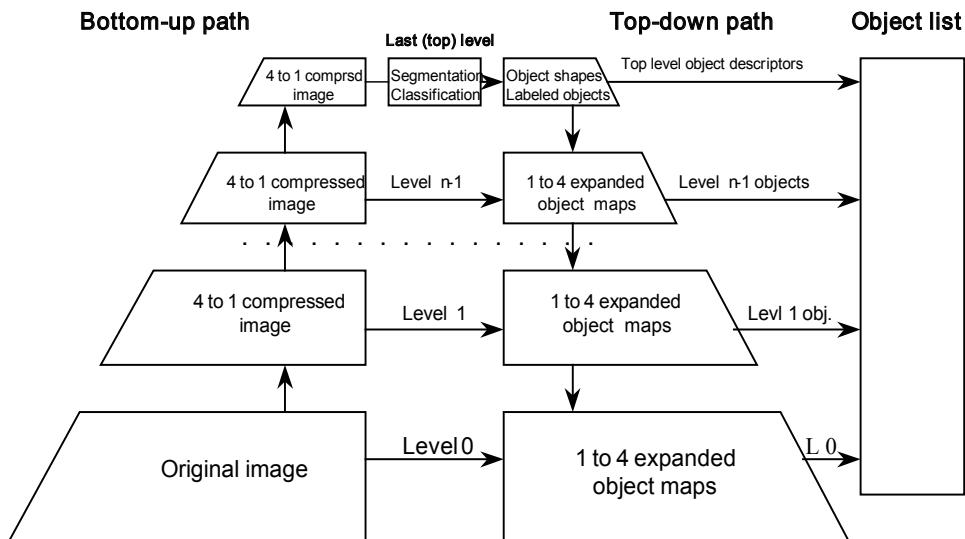


Fig. 1. The block-diagram of physical information elucidation.

As can be seen at Fig. 1, the proposed schema is comprised of three main processing paths: the bottom-up processing path, the top-down processing path and a stack where the discovered information content (the generated descriptions of it) is actually accumulated. The algorithm's structure reflects the principles of information representation, which have been already defined previously.

As it is shown in the schema, the input image is initially squeezed to a small size of approximately 100 pixels. The rules of this shrinking operation are very simple and fast: four non-overlapping neighbor pixels in an image at level L are averaged and the result is assigned to a pixel in a higher ($L+1$)-level image. This is known as “four children to one parent relationship”. Then, at the top of the shrinking pyramid, the image is segmented, and

each segmented region is labeled. Since the image size at the top is significantly reduced and since in the course of the bottom-up image squeezing a severe data averaging is attained, the image segmentation/labeling procedure does not demand special computational resources. Any well-known segmentation methodology will suffice. We use our own proprietary technique that is based on a low-level (single pixel) information content evaluation (Diamant, 2003), but this is not obligatory.

From this point on, the top-down processing path is commenced. At each level, the two previously defined maps (average region intensity map and the associated label map) are expanded to the size of an image at the nearest lower level. Since the regions at different hierarchical levels do not exhibit significant changes in their characteristic intensity, the majority of newly assigned pixels are determined in a sufficiently correct manner. Only pixels at region borders and seeds of newly emerging regions may significantly deviate from the assigned values. Taking the corresponding current-level image as a reference (the left-side unsegmented image), these pixels can be easily detected and subjected to a refinement cycle. In such a manner, the process is subsequently repeated at all descending levels until the segmentation/classification of the original input image is successfully accomplished.

At every processing level, every image object-region (just recovered or an inherited one) is registered in the objects' appearance list, which is the third constituting part of the proposed scheme. The registered object parameters are the available simplified object's attributes, such as size, center-of-mass position, average object intensity and hierarchical and topological relationship within and between the objects ("sub-part of...", "at the left of...", etc.). They are sparse, general, and yet specific enough to capture the object's characteristic features in a variety of descriptive forms.

Examples of algorithm's performance and some concrete palpable results are provided in several previously published papers (Diamant, 2005a; Diamant, 2005b).

In our current discussion it is worth to be mentioned that: First, image segmentation (physical image structures delineation, physical image information elicitation) is performed in a top-down manner, not in a conventional bottom-up mode. Second, the suggested image segmentation principle does not require any knowledge about high-level rules, which are used to support the segmentation process and which are an obligatory part of any bottom-up proceeding procedure. Third, canceling the necessity of these high-level rules, makes all Machine Learning techniques useless and invalidates all efforts that are specially carried out to meet this sacred requirement! In this way, Machine Learning loses its role as the main performer in physical information recovery enterprises.

4.2 Image Semantic Information Processing

The context of this sub-section is also an extended quotation from a recently published paper (Diamant, 2008). The key point of this quotation is a semantic information processing architecture based on the same information-defining rules and the same (top-down) information representation principles that were already introduced in Section 3. The block-schema of such a semantic information processing architecture is borrowed from the above mentioned paper (Diamant, 2008), and is depicted in the Fig. 2.

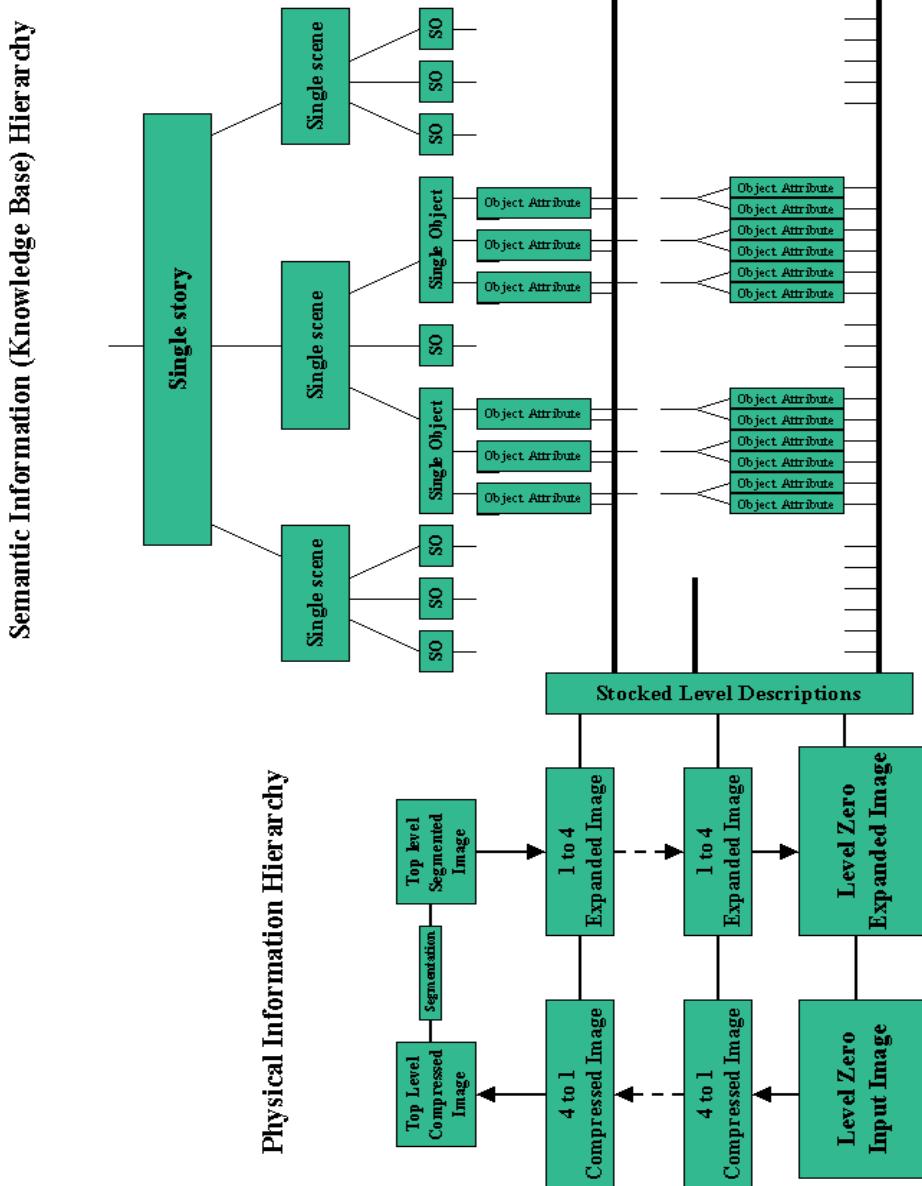


Fig. 2. Physical and Semantic Information processing hierarchies.

Scrutinizing of this general image information processing architecture must be preceded by some remarks: Semantic information, which (as we understand now) is a property of an external observer, is separated and dissociated from the physical information processing, in our case an image. Therefore it must be treated (or modeled) in accordance with observer-specific (his/her) cognitive information processing rules.

It is well known that human cognitive abilities (including the aptness for image interpretation and the capacity to assign semantics to an image) are empowered by the existence of a huge knowledge base about the things in the surrounding world kept in human brain.

This knowledge base is permanently upgraded and updated during the human's life span. So, if we intend to endow our design with some cognitive capabilities we have to provide it with something equivalent to this (human) knowledge base.

It goes without saying that this knowledge base will never be as large and developed as its human prototype. But we are not sure that such a requirement is valid here. After all, humans are also not equal in their cognitive capacities, and the content of their knowledge bases is very diversified as well. (The knowledge base of an aerial photographs interpreter is certainly different from the knowledge base of an X-ray images interpreter, or an IVUS images interpreter, or PET images). The knowledge base of our visual thinking machine has to be small enough to be effective and manageable, but sufficiently large to ensure the machine acceptable performance. Certainly, for our feasibility study we can be satisfied even with a relatively small, specific-task-oriented knowledge base.

The next crucial point is the knowledge representation issue. To deal with it, we first of all must arrive at a common agreement about what is the meaning of the term "knowledge". (A question that usually does not have a single answer.) We state that in our case a suitable definition of it would be: "**Knowledge is memorized information**". Consequently, we can say that knowledge (like information) must be a hierarchy of descriptive items, with the grade of description details growing in a top-down manner at the descending levels of the hierarchy.

One more point that must be mentioned here, is that these descriptions have to be implemented in some alphabet (as it is in the case of physical information) or in a description language (which better fits the semantic information case). Any farther argument being put aside, we will declare that the most suitable language in our case is the natural human language. After all, the real knowledge bases that we are familiar with are implemented in natural human languages.

The next step, then, is predetermined: if natural language is a suitable description implement, the suitable form of this implementation is a narrative, a story tale (Tuffield et al., 2005). If the description hierarchy can be seen as an inverted tree, then the branches of this tree are the stories that encapsulate human's experience with the surrounding world. And the leaves of these branches are single words (single objects) from which the story parts (single scenes) are composed of.

The descent into description details, however, does not stop here, and each single word (single object) can be farther decomposed into its attributes and rules that describe the relations between the attributes.

At this stage the physical information reappears. Because the words are usually associated with physical objects in the real world, words' attributes must be seen as memorized physical information (descriptions). Once derived (by the human visual system) from the observable world and learned to be associated with a particular word, these physical information descriptions are soldered in into the knowledgebase. Object recognition, thus, turns out to be a comparison and similarity test between currently acquired physical information and the one already retained in the memory. If the similarity test is successful, starting from this point in the hierarchy and climbing back up on the knowledgebase ladder

we will obtain: first, the linguistic label for a recognized object; second, the position of this label (word) in the context of the whole story; and third, the ability to verify the validity of an initial guess by testing the appropriateness of the neighboring parts composing the object, that is, the context of a story. In this way, object's meaningful categorization can be reached, and the first stage of image annotation can be successfully accomplished, providing the basis for farther meaningful (semantic) image interpretation.

One question has remained untouched in our discourse: How does this artificial knowledgebase have to be initially created and brought into our thinking machine disposal? This subject deserves a special discussion.

4.3 Can Semantic Knowledge be Learned?

There is no need to reiterate the dictums of the today's Internet revolution, where access and exchange of semantic information is viewed as a prime and an ultimate goal. Machines are supposed to handle the documents' semantic content, and Machine Learning techniques, thus, supporting this knowledge mining venture are supposed to be the leading force, the centre forward of this exciting enterprise. Semantic Knowledge mining is now the hottest topic of every conference discussion, most recent research projects and many other applied science initiatives. However, in the light of our new definition of information, which was recently introduced in my work and re-introduced in the Section 3 of this paper, I am really skeptic about the Machine Learning ability to meet this challenge.

Again, some philosophy would not be out of place here. At first, it must be reiterated that semantics is not a natural property of an image (or a natural property of the data, if you would like a more general view on the subject). Semantics is a property of a human observer that watches and scrutinizes the data, and this property is shared among the observer and other members of his community. By the way, this community does not have to embrace the whole mankind, it can be even a very small community of several people or so, which, nevertheless, were lucky to establish a common view on a particular subject and a common understanding of its meaning. That is the reason why this particular (privet) knowledge can not be attained in any reasonable way, including Machine Learning techniques and tricks.

On the other hand, an intelligent information-processing system has to have at its disposal a memorized knowledgebase hierarchy (implemented, as we postulate, as a collection of typical stories) against which the physical information of its input sensors is matched and associated. Finding the suitable story whose attributes most closely match the sensors' physical information is equivalent to finding the interpretation for the input sensor data (the input physical information). That is the novelty of our proposed architecture. That is the most important feature of our design approach: The knowledgebase hierarchy is used for a linguistic input interpretation, but this knowledge is not derived (by the system) from the input data. It is not learned from the data. On the contrary, in accordance with the top-down information unfolding principle, the knowledge-base hierarchy (as a whole) has to be transferred to the system disposal from the outside. The system doesn't learn the knowledgebase, it is taught to use the knowledgebase (In our case, a pool of task related stories and their ramifications putted at system disposal in advance).

Thus, providing the system with the needed new knowledge each time when the system is due for a new task accomplishment is becoming a natural duty of Artificial Intelligence (Machine Intelligence) system designer. This shift from Machine Learning to Machine Teaching paradigm should become the key point of intelligent system design and

development roadmap. But unfortunately, that has not happen although it has been about three years since the idea was at first articulated and even occasionally published (Diamant, 2006b).

4.4 Some additional remarks

That is a very important and an interesting twist in the philosophy of intelligent artificial systems design. It does not result from the understanding of the principals of biological systems intelligence or other proudly declared biological inspirations. On the contrary, it results from pure mathematical considerations of the Kolmogorov's complexity theory. Only now, drawing on the insights of Kolmogorov's theory, we can seize the interpretation of the facts that we usually come across in our natural (biological) surrounding.

It is a very subtle issue among the topics of machine intelligence that I would like to address. "Biologically inspired" means that we borrow from the nature some fruitful ideas, which we intend to replicate in our artificial designs. That is, we presume that we understand or at least are very close to the state of understanding how some biological mechanisms operate, performing their natural duties. But that is not true!. We don't have even a slightest hint about how the nature works. What we have are gambling guesses, intuitive feelings, speculations, and - nothing more than that.

Another important remark in this regard, is that Nature is not an Engineer. It does not invent new mechanisms and new solutions for its problem-solving. On the contrary, it gradually adjusts and adapts what it already has on the hand. Although the final results are really remarkable, it takes a lot of time to reach them in the course of natural evolution, millions and billions of years. Despite all this, the nature has never reached some very important human-life-shaping revelations - for example, the wheel (as a means for transportation), the cooked food, the writing and numbering practice, etc.

The inventors of "Genetic Programming" provide very interesting quotations from Turing's early works considering Machine Intelligence (Koza et al., 1999; Koza et al., 2002). In his 1948 essay "Intelligent Machines" Alan Turing has identified three broad approaches by which machine intelligence could be achieved: "One approach... is a search through the space of integers representing candidate computer programs, (a logic-driven search)... Another approach is the "cultural search" which relies on knowledge and expertise acquired over a period of years from others. This approach is akin to present-day knowledge-based systems... The third approach is "genetical or evolutionary search"..." (Koza, et al., 1999). From the three, the inventors of Genetic Programming pick up the idea of biological relevance to the problem of machine intelligence acquisition. However, from our point of view (from the point of view inspired by Kolmogorov's theory) this can not be true. Genetic Programming and Neural Networking are pure bottom-up information-processing approaches. As we know today, the right way of information retrieval is a top-down coarse-to-fine approach. Therefore, it might be more intelligent to embrace the first Turing's alternative - the logic-driven approach. That is, relying on pure logical and engineering approaches to find out the best ways of intelligence reification, and only then to verify our hypothetical solutions against known (or unknown) biological evidences and facts. That is exactly what we are intended to do now.

The first issue is the bottom-up versus top-down information-processing alternatives. Despite the traditional dominance of the bottom-up approach, evidence for top-down preliminary processing in biological vision systems is present in research literature since the

early 80s of the previous century (Navon, 1977; Chen, 1982). Unfortunately, they were overlooked both by biological and computer vision communities.

The next phenomenon which is usually misunderstood (and consequently mistreated) is the knowledge transfer (in human and animal world), which is usually mistakenly defined as a Learning process. We have proposed a more suitable definition – a Teaching process. Indeed, it turns out that in nature, teaching is a universal and a wide-spread phenomenon. Only recently this fact has become recognized and earned its careful investigation (Csibra, 2007; Hoppitt et al., 2008). Teaching in nature does not mean human-like mentoring – animals do not possess spoken language capabilities. Teaching in nature assumes specific semantic knowledge transfer, specific information relocation from a teacher to a pupil, from one community member to another. And examples of this knowledge transfer are really abundant in our surrounding, if only we are ready to look at them and see them in a proper way.

In this regard, dancing bees that convey to the rest of the hive the information about melliferous sites (Zhang et al., 2005), ants that learn in tandem (Franks & Richardson, 2006), and even bacteria developing their antibiotic resistance as a result of a so-called horizontal gene transfer when a single DNA fragment of one bacteria is disseminated among other colony members (Lawrence & Hendrickson, 2003), all these examples convincingly support our claim that meaningful information (the semantic knowledge base) is always transferred to the individual information processing system from the outside, from the external world. The system does not learn it in a traditionally assumed Machine Learning manner.

Another benefit which biological science can gain from our logically-driven (engineering) approach is the issue of astrocyte-neuron communication. Only defining information as a description message you can explain how astrocytes, (the dominant glial cells), “listen and talk” with neuronal and synaptic networks. In their paper, Volterra & Meldolesi wrote that: “One reason that the active properties of astrocytes have remained in the dark for so long relates to the differences between the excitation mechanisms of these cells and those of neurons. Until recently, the electrical language of neurons was thought to be the only form of excitation in the brain. Astrocytes do not generate action potentials, they were considered to be non-excitable and, therefore, unable to communicate. The finding that astrocytes can be excited non-electrically has expanded our knowledge of the complexity of brain communication to an integrated network of both synaptic and non-synaptic routes” (Volterra & Meldolesi, 2005). That is, traditional belief that a spiking neuron burst is a valid form of information exchange and representation does not hold any more, and has to be replaced by a chemical molecular-language-based description-massages transfer mechanism.

A very important issue of our discussion about semantic information processing is the issue of knowledge representation. As it was already mentioned above, and it also stems from the insights of Kolmogorov’s theory, the best form of knowledge representation has to be a language-based description, a narrative, a story. I do not intend to expand here on the implementation details of this issue. I would like to continue to maintain our discussion on a philosophical level. What follows from this is that we have always to consider intelligence as being carried out in a language, in a linguistic structure. That is, although the block-schema depicted in Fig. 2 outlines only visual information incorporation into the semantic processing hierarchy, you can easily imagine physical information of other modalities (hearing, haptics, olfactory senses information) being subjected (usually in parallel with information from other sensors) as attributes of semantic (linguistic) objects into the

knowledgebase processing hierarchy. (That will again explain you why functional Magnetic Resonance Imaging shows you that visual stimuli are processed in audio stimuli processing zones, which are naturally associated with speech processing. The simple explanation for this is that all modalities are finally processed in the language processing zone, as it is proposed by our approach.)

The next important issue, which naturally follows the preceding ones, is the narrative story form of knowledge representation accepted for the discussed case of semantic information processing. Linguistic tagging, that means labeling image objects with words, is a well known and widely used methodology of image semantics retrieval supported by a special class of Machine Learning techniques (Barnard et al., 2003; Duygulu et al., 2008; Blondin Masse et al., 2008). This way of thinking naturally stems from another wide-spread assumption that ontology (the basis of semantic reasoning and elaboration) is a vocabulary, a thesaurus, a dictionary. What follows from our descriptive form of knowledge representation is that ontology has to be treated as a story, a narrative, which naturally describes the system's behavior in various real-life-encountered situations. However, this very important aspect of intelligent systems design philosophy leads us far away from the main theme of our discussion – the philosophy of Machine Learning. And for that reason I will quit at this point, and not continue further.

5. Conclusions

In this paper I have attempted to promote a new Thinking Machines design and development philosophy. The central point of my approach is a new definition of information, that is, a notion of information as a language-based description. Then, above it the notion of intelligence can be placed, defining intelligence as the system's ability to process information. The principles of information mining should be placed in the lower part of the construction. Thus, it seems to me, a proper frame for a rational Artificial or Machine Intelligence devices research and development enterprise can be established.

Essentially, the declared focus of the paper's subject is the issue of Machine Learning, which is assumed to be a bundle of techniques used to support all information-processing machinery. But, as you know, Machine Learning as by now (and already for a very long time) is treated as an independent and stand alone discipline, totally detached from its original destination – Thinking Machines research and development (Turing, 1950). The roadmap for this challenge was formulated at the Dartmouth College meeting in the summer of 1956 (McCarthy, et al. 1955). The date of this meeting is considered today as the Artificial Intelligence (AI) birthday. (The very name of AI was coined at this time by John McCarthy, one of the authors of the Dartmouth Proposal).

At first, the excitement and hopes were really high, and the goals have seemed to be reachable in a short while. In the Panel Discussion at the Artificial General Intelligence (AGI) Workshop in 2006, Steve Grand has recalled that "Rodney Brooks has a copy of a memo from Marvin Minsky (another father of the Dartmouth Proposal), in which he suggested charging an undergraduate for a summer project with the task of solving vision. I don't know where that undergraduate is now, but I guess he hasn't finished yet" (Panel Discussion, 2006).

Indeed, problems of Vision, as well as all other AI troubles, have turned out to be much more complicated and problematic than it looked out at the beginning. Within a decade or

so, it became clear that AI tribulations are immense, maybe even intractable. As a consequence, the AI community to a large extent has abandoned its original dream, and turned to more “practical” and “manageable” problems (Wang & Goertzel, 2006). “AI has evolved to being a label on a family of relatively disconnected efforts” (Brachman, 2005). Exactly the same were the milestones of Machine Learning. Machine Learning, which was always perceived as an indispensable component of intelligence, has undergone all the metamorphoses as its general domain. At first, there was a generous offer to let the system by itself (in an autonomous manner) to find out the best way to mimic Intelligence. Why to trouble oneself trying to grasp the principles of intelligence? Let us give the machine the chance to do this by itself. (I can not to withstand the temptation to provide an example of such a fatal misunderstanding: IGI Global Publisher (formerly Idea Group Inc.) has published a Call for Chapter Proposals for a future book “Intelligent Systems for Machine Olfaction: Tools and Methodologies” (Can be found at the publisher site: <http://www.igi-global.com/requests/details.asp?ID=610>). You can read in the Introduction part of it: “Intelligent systems are those that, given some data, are able to learn from that data. This ability makes it possible for complex systems to be modeled and/or for performance to be predicted. In turn it is possible to control their functionality through learning/training, without the need for a priory knowledge of the system’s structure”. Once more, I apologize for such a so long quotation.)

Then, when the first idealistic objective has failed, Machine Learning was broken into pieces, disintegrated and fragmented to many partial tasks and goals. Now the question in the paper’s title – “When and Where the Horses Went Astray?” – can be answered beyond any doubts: It has happened about 50 years ago!

From the standpoint that we possess today, we can even spell out the fundamental flaws which are responsible for this derailment: First, the bottom-up philosophy of information retrieval. (As we know today, the right way of information treatment is the top-down coarse-to-fine line of information processing). Second, is the lack of a proper definition of information, leading, consequently, to a lack of a clear distinction between physical and semantic information. (This failure had a tremendous impact on the Machine Learning disruption). The same can be said about the third misleading factor – misunderstanding of the very nature of semantic information, which has led to an endless, infamous race for knowledge and semantic meaning extraction directly from the raw data. (Which is, obviously, a philosophical lapse).

For the same reasons, the basic notion of intelligence has been overlooked and defined erroneously. I hope, in this paper I was lucky to repair some of these misconceptions.

6. References

- Awad, A. & Man, H. (2008). Similar Neighbourhood Criterion for Edge Detection in Noisy and Noise-Free Images, *Proceedings of the International Multiconference on Computer Science and Information Technology*, pp. 483-486, Wisla, Poland, October 2008.
- Barnard, K.; Duygulu, P.; Forsyth, D.; de Freitas, N.; Bley, D. & Jordan, M. (2003). Matching Words and Pictures, *Journal of Machine Learning Research*, Vol. 3, pp. 1107-1135.
- Biederman, I. (1987). Recognition-by-Components: A Theory of Human Image Understanding, *Psychological Review*, Vol. 94, No. 2, 1987, pp. 115-147.

- Blondin Masse, A.; Chicoisne, G.; Gargouri, Y.; Harnad, S.; Picard, O. & Marcotte, O. (2008). How Is Meaning Grounded in Dictionary Definitions? Available: <http://arxiv.org/abs/0806.3710>.
- Brachman, R. (2005). Getting Back to "The Very Idea". *AI Magazine*, Vol. 26, pp. 48-50, Winter 2005.
- Chaitin, G. (1966). On the length of programs for computing finite binary sequences. *Journal of the ACM*, Vol. 13, pp. 547-569, 1966.
- Chen, L. (1982). Topological structure in visual perception, *Science*, 218, pp. 699-700, 1982.
- Crick, F. & Koch, C. (2000). The Unconscious Homunculus, In: *The Neuronal Correlates of Consciousness*, Metzinger, T. (Ed.), pp. 103-110, MIT Press: Cambridge, MA, 2000.
- Csibra, G. (2007). Teachers in the wild. *Trends in Cognitive Science*, Vol. 11, No. 3, pp. 95-96, March 2007.
- Diamant, E. (2002). Image Segmentation Scheme Ruled by Information Density Optimization, Submitted to *British Machine Vision Conference (BMVC-2002)* and decisively rejected there. Available: <http://www.vidia-mant.info>.
- Diamant, E. (2003). Single Pixel Information Content, *Proceedings SPIE*, Vol. 5014, pp. 460-465, IST/SPIE 15th Annual Symposium on Electronic Imaging, Santa Clara, CA, January 2003.
- Diamant, E. (2004). Top-Down Unsupervised Image Segmentation (it sounds like an oxymoron, but actually it isn't), *Proceedings of the 3rd Pattern Recognition in Remote Sensing Workshop (PRRS'04)*, Kingston University, UK, August 2004.
- Diamant, E. (2005a). Searching for image information content, its discovery, extraction, and representation, *Journal of Electronic Imaging*, Vol. 14, Issue 1, January-March 2005.
- Diamant, E. (2005b). Does a plane imitate a bird? Does computer vision have to follow biological paradigms?, In: De Gregorio, M., et al, (Eds.), *Brain, Vision, and Artificial Intelligence*, First International Symposium Proceedings. LNCS, Vol. 3704, Springer-Verlag, pp. 108-115, 2005. Available: <http://www.vidia-mant.info>.
- Diamant, E. (2006a). In Quest of Image Semantics: Are We Looking for It Under the Right Lamppost?, Available: <http://arxiv.org/abs/cs.CV/0609003>; <http://www.vidia-mant.info>.
- Diamant, E. (2006b). Learning to Understand Image Content: Machine Learning Versus Machine Teaching Alternative, *Proceedings of the 4th IEEE Conference on Information Technology: Research and Education (ITRE-2006)*, Tel-Aviv, October 2006.
- Diamant, E. (2007). The Right Way of Visual Stuff Comprehension and Handling: An Information Processing Approach, *Proceedings of The International Conference on Machine Learning and Cybernetics (ICMLC-2007)*, Hong Kong, August 2007.
- Diamant, E. (2008). Unveiling the mystery of visual information processing in human brain, *Brain Research*, Vol. 1225, 15 August 2008, pp. 171-178.
- Duygulu, P.; Bastan, M. & Ozkan, D. (2008). Linking image and text for semantic labeling of images and videos, In: *Machine Learning Techniques for Multimedia*, M. Cord & P. Cunningham (Eds.), Springer Verlag, 2008.
- Floridi, L. (2003). From Data to Semantic Information, *Entropy*, Vol. 5, pp. 125-145, 2003.
- Franks, N. & Richardson, T. (2006). Teaching in tandem-running ants, *Nature*, 439, p. 153, January 12, 2006.

- Gerchman, Y. & Weiss, R. (2004). Teaching bacteria a new language. *Proceedings of The National Academy of Science of the USA (PNAS)*, Vol. 101, No. 8, pp. 2221-2222, February 24, 2004.
- Ghosh, K.; Sarkar, S. & Bhaumik, K. (2007). The Theory of Edge Detection and Low-level Vision in Retrospect, In: *Vision Systems: Segmentation and Pattern Recognition*, G. Obinata and A. Dutta, (Eds.), I-Tech Publisher, Viena, June 2007.
- Goertzel, B. (2006). Panel Discussion: What are the bottlenecks, and how soon to AGI?, *Proceedings of the Artificial General Intelligence Workshop (AGI 2006)*, Washington DC, May 2006.
- Grunvald, P. & Vitanyi, P. (2008). Algorithmic Information Theory, In: *The Handbook of the Philosophy of Information*, P. Adriaans, J. van Benthem (Eds.), pp. 281-320, North Holland, 2008. Available: <http://arxiv.org/abs/0809.2754>.
- Hoppitt, W.; Brown, G.; Kendal, R.; Rendell, L.; Thornton, A.; Webster, M. & Laland, K. (2008). Lessons from animal teaching. *Trends in Ecology and Evolution*, Vol. 23, No. 9, pp. 486-493, September 2008.
- Hutter, M. (2007). Algorithmic Information Theory: A brief non-technical guide to the field, Available: <http://arxiv.org/abs/cs/0703024>.
- Koch, C.; Cerf, M.; Harel, J.; Einhäuser, W. (2007). Predicting human gaze using low-level saliency combined with face detection, *Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems (NIPS 2007)*, Vancouver, Canada, December 2007. Available: <http://papers.klab.caltech.edu/view/year/2007.html>.
- Kolmogorov, A. (1965). Three approaches to the quantitative definition of information, *Problems of Information and Transmission*, Vol. 1, No. 1, pp. 1-7, 1965.
- Koza, J.; Bennett, F.; Andre, D. & Keane, M. (1999). Genetic Programming: Turing's Third Way to Achieve Machine Intelligence. *EUROGEN Workshop* in Jyväskylä, Finland, May 1999. Available: <http://www.genetic-programming.com/jkpdf/eurogen1999>.
- Koza, J.; Bennett, F.; Andre, D. & Keane, M. (2002). Genetic Programming: Biologically Inspired Computation that Exhibits Creativity in Solving Non-Trivial Problems. In: *Evolution as Computation: DIMACS Workshop*, Princeton, 2002. Available: <http://gridley.res.carleton.edu/~kachergg/docs/geneticProgramming.pdf>.
- Lawrence, J. & Hendrickson, H. (2003). Lateral gene transfer: when will adolescence end?, *Molecular Microbiology*, vol. 50, no. 3, pp. 739-749, 2003.
- Legg, S. & Hutter, M. (2007). Universal Intelligence: A Definition of Machine Intelligence, Available: <http://arxiv.org/abs/0706.3639>.
- Li, M. & Vitanyi, P. (2008). *An Introduction to Kolmogorov Complexity and Its Applications*, Third Edition, New York: Springer-Verlag, 2008.
- McCarthy, J.; Minsky, M.; Rochester, N. & Shannon, C. (1955). A proposal for the Dartmouth summer research project on Artificial Intelligence, *AI Magazine*, Vol. 27, No. 4, 2006. Avail.: <http://www.aaai.org/ojs/index.php/aimagazine/article/viewFile/1904/1802>.
- Marr, D. (1978). Representing visual information: A computational approach, *Lectures on Mathematics in the Life Science*, Vol. 10, pp. 61-80, 1978.
- Marr, D. (1982). *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, Freeman, San Francisco, 1982.
- Navon, D. (1977). Forest Before Trees: The Precedence of Global Features in Visual Perception, *Cognitive Psychology*, 9, pp. 353-383, 1977.

- Panel Discussion, (2006). Panel Discussion: What are the bottlenecks, and how soon to AGI?, Proceedings of the AGI Workshop, Washington DC, USA, May 2006.
- Qiu, P. & Sun, J. (2009). Using Conventional Edge Detectors and Post-Smoothing for Segmentation of Spotted Microarray Images, *Journal of Computational and Graphical Statistics*, Vol.18, No. 1, pp. 147-164, 2009.
- Saba, W. (2008). Commonsense Knowledge, Ontology and Ordinary Language. *International Journal of Reasoning-based Intelligent Systems*, Vol. n., No. m., pp. 43-60, 2008. Available: <http://arxiv.org/abs/0808.1211>.
- Sadrzadeh, M. (2008). Occam's razor and reasoning about information flow, Available: <http://arxiv.org/abs/cs/0808.1354>.
- Shannon, C. E. (1948). The mathematical theory of communication, *Bell System Technical Journal*, Vol. 27, pp. 379-423 and 623-656, July and October 1948.
- Shomstein, S. & Behrmann, M. (2008). Object-based attention: Strength of object representation and attentional guidance. *Perception & Psychophysics*, Vol. 70, No. 1, pp. 132-144, January 2008.
- Solomonoff, R. J. (1964). A formal theory of inductive inference. *Information and Control*, Part 1: Vol. 7, No. 1, pp. 1-22, March 1964; Part 2: Vol. 7, No. 2, pp. 224-254, June 1964.
- Torralba, A. (2009). How many pixels make an image? *Visual Neuroscience*, Vol. 26, Issue 1, pp. 123-131, 2009. Available: <http://web.mit.edu/torralba/www/>.
- Treisman, A. (1996). The binding problem. *Current Opinion in Neurobiology*, Vol. 6, pp.171-178, 1996.
- Treisman, A. (2003). Consciousness and perceptual binding. Available: <http://www.csmb.princeton.edu/conte/pdfs/project2/Proj2Pub5anne.pdf>.
- Treisman, A. & Gelade, G. (1980). A feature-integration theory of attention, *Cognitive Psychology*, Vol. 12, pp. 97-136, Jan. 1980.
- Tuffield, M.; Shadbolt, N. & Millard, D. (2005). Narratives as a Form of Knowledge Transfer: Narrative Theory and Semantics, Proceedings of the 1st AKT (Advance Knowledge Technologies) Symposium, Milton Keynes, UK, June 2005.
- Turing, A. (1950). Computing machinery and intelligence. *Mind*, Vol. 59, pp. 433-460. Available: <http://scholar.google.co.il/>.
- Vitanyi, P. (2006). Meaningful Information, *IEEE Transactions on Information Theory*, Vol. 52, No. 10, pp. 4617-4624, October 2006. Availbl: <http://www.cwi.nl/~paulv/papers>.
- Volterra, A. & Meldolesi, J. (2005). Astrocytes, from brain glue to communication elements: the revolution continues, *Nature Reviews, Neuroscience*, vol. 6, No. 8, pp. 626-640.
- Wang, P. (2006). The Logic of Intelligence. In: *Artificial General Intelligence*, Wang, P. & Goertzel, B. (Eds.), pp. 31-62. Springer Verlag, May 2006. Available: <http://nars.wang.googlepages.com/nars%3Application>.
- Wang, P. & Goertzel, B. (2006). Introduction: Aspects of Artificial General Intelligence. In: *Artificial General Intelligence*, Wang, P. & Goertzel, B. (Eds.), Springer Verlag, 2006. Available: <http://nars.wang.googlepages.com/nars%3Application>.
- Zhang, S.; Bock, F.; Si, A.; Tautz, J. & Srinivasan, M. (2005). Visual working memory in decision making by honey bees, *Proceedings of The National Academy of Science of the USA (PNAS)*, vol. 102, no. 14, pp. 5250-5255, April 5, 2005.

SOMs for machine learning

Iren Valova, Derek Beaton and Daniel MacLean
University of Massachusetts Dartmouth
 USA

1. Introduction

In this chapter we offer a survey of self-organizing feature maps with emphasis on recent advances, and more specifically, on growing architectures. Several of the methods are developed by the authors and offer unique combination of theoretical fundamentals and neural network architectures. Included in this survey of dynamic architectures, will also be examples of application domains, usage and resources for learners and researchers alike, to pursue their interest in SOMs.

The primary reason for pursuing this branch of machine learning, is that these techniques are unsupervised – requiring no a priori knowledge or trainer. As such, SOMs lend themselves readily to difficult problem domains in machine learning, such as clustering, pattern identification and recognition and feature extraction. SOMs utilize competitive neural network learning algorithms introduced by Kohonen in the early 1980's. SOMs maintain the features (in terms of vectors) of the input space the network is observing. This chapter, as work emphasizing dynamic architectures, will be incomplete without presenting the significant achievements in SOMs including the work of Fritzke and his growing architectures.

To exemplify more modern approaches we present state-of-the art developments in SOMs. These approaches include parallelization (ParaSOM – as developed by the authors), incremental learning (ESOINN), connection reorganization (TurSOM – as developed by the authors), and function space organization (mnSOM). Additionally, we introduce some methods of analyzing SOMs. These include methods for measuring the quality of SOMs with respect to input, neighbors and map size. We also present techniques of posterior recognition, clustering and input feature significance. In summary, this chapter presents a modern gamut of self-organizing neural networks, and measurement and analysis techniques.

2. Overview of competitive learning

2.1 Unsupervised and competitive learning

Very broadly defined, neural networks learn by example and mimic human brain in its decision or object identification capabilities. The concept of artificial neural networks (ANN) is based on two different views of the human brain activity, both of which rely on the functionality of a single neuron. The neurons are perceived as adding devices, which react,

or fire, once the incoming signals sum reaches a threshold level. Fig.1 illustrates the functionality of a single neuron, which receives signals from other neurons it is connected to via weighted synapses. Upon reaching the firing level, the neuron will broadcast a signal to the units connected to its output.

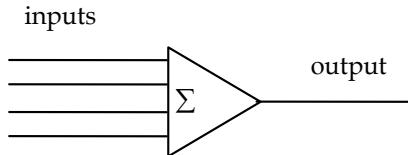


Fig. 1. Artificial neuron functionality

Returning to the two major types of ANN, one view generally relies on the individual neuron and its ability to respond, or fire, given sufficient stimulus. The topology of neurons and the connections among them is not the goal of this type of ANN, but rather the output produced by the respective neuron.

The second view banks on the neurons functioning as a team. As such, it takes into account the concept of map formed by neuron positions, much like the visual cortex map, producing a two dimensional image of the perceived visual field. This type of ANN produces a topology of neurons, connected by weighted synapses and features the natural grouping of the input data (Fig.2). This translates into input density map and necessitates the development of evaluation procedures on the formed clusters for the purpose of identifying or matching patterns in the data.

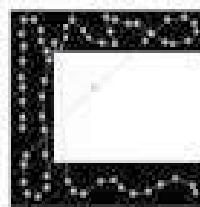


Fig. 2. The black area denotes input space distribution, where the neurons have organized to cover that input topology as a team

The taxonomy of learning methods and algorithms for ANN is multifaceted and includes many hierarchical classifications (Fig.3). In this chapter we are concerned with unsupervised learning that is also competitive. Learning in ANN is the process of connection weight adjustment, which, in turn guides the neuron to a better position in terms of input data configuration.

In the case of supervised learning, the weight adjustment will be guided by the teaching signal and the penalty/reward of the error in the ANN response. Unsupervised learning methods do not benefit from teacher signal guidance. The neurons compete to match the input as closely as possible, usually based on Euclidean distance. The neuron closest to the considered input exemplar is the winner taking it all, i.e. adjusting its weight to improve its position and thus move closer to the input.

We are describing the extreme case of competition, i.e. winner-take-all. Depending on the learning algorithm and the ANN application, while the winning neuron will be selected, a neighbourhood of influence may be established, whereby the winning neuron neighbours will also move in the same direction albeit at a lesser distance.

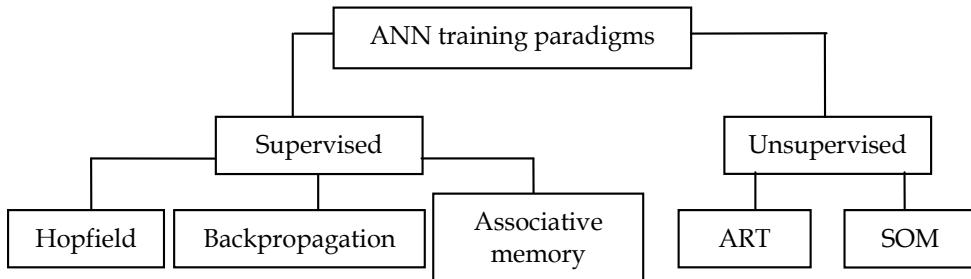


Fig. 3. Taxonomy of ANN learning methods

Unsupervised learning (UL) is generally based on competition. UL seeks to map the grouping or patterns in the input data. This can be accomplished either by neurons resonating with the input exemplar (e.g. adaptive resonance theory) or by neurons winning the distance from the input exemplars competition. It must be noted that there are ANN models that learn in unsupervised manner, but are not based on competition. Among those, the principal component network should be mentioned here as a prelude to later sections in this chapter.

2.2 Kohonen's SOM

The brains of higher animals are organized by function, e.g. the visual cortex processes the information received through the optical nerve from the eyes, the somatosensory cortex maps the touch information from the surface of the body, etc. Inspired by the mapping abilities of the brain, the self-organizing feature map (SOM) was introduced in early 1980's by Teuvo Kohonen (Kohonen, 1995). SOMs are used to topologically represent the features of the input based on similarity usually measured by Euclidean distance. SOMs are useful tools in solving visualization and pattern recognition tasks as they map a higher dimension input space into a one- or two-dimensional structure. SOMs are initialized usually randomly (Fig.4b), in a topology with fixed number of neurons, that can be ordered in a chain (i.e. each neuron has at most two neighbors) or in a two-dimensional grid of rectangular (Fig.4c) or hexagonal nature, where the neurons have at most four neighbors.



Fig. 4. Input space: a) distribution; b) with randomly initialized neurons; c) two-dimensional rectangular grid

Before a brief overview of the SOM algorithm, let us take the reader through the concept of Kohonen's ANN. The input space of Fig.4a is used along with the random initialization in Fig.4b. As every input vector (in this case a two-dimensional x, y representation of the location of each dot comprising the black area in Fig.4a) is presented to the network, the closest neuron responds as a winner of the Euclidean distance-based competition and updates its weight vector to be closer to the input just analyzed. So do its neighbors dependent on the neighborhood radius, which can be reduced as the time progresses. The rate of reduction is determined by the learning rate. As inputs are presented in a random order, the neurons move closer to the input vectors they can represent and, eventually, the movement of neurons becomes negligible. Usually, that is when the map is considered to have converged. This process is illustrated in Fig.5a for one-dimensional SOM and Fig.5b for rectangular SOM.

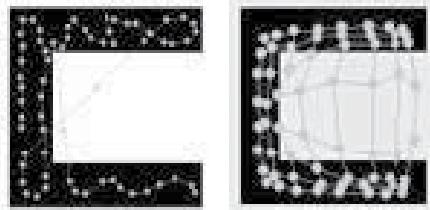


Fig. 5. Converged SOM: a) one-dimensional topology; b) two-dimensional topology

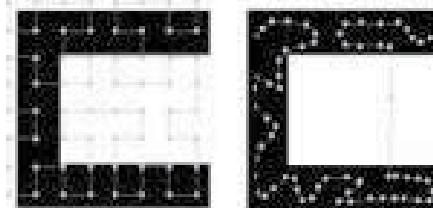


Fig. 6. Hilbert curve initialization approach: a) initialized network; b) resulting map

While the classic SOM features random initialization of the weight vectors, the authors have demonstrated the advantages of one-dimensional SOM initialization based on a Hilbert curve (Buer, 2006; Valova, Beaton, & MacLean, 2008a) with neurons positioned in a chain following the space-filling curve (Fig.6a). Kohonen posited that one-dimensional SOM converge in a shape resembling Peano curves. The authors followed this observation and utilized the idea in the initialization process to speed up convergence and ensure linearity of the network. Fig.6b demonstrates the resulting map. It is obvious that the map is not tangled, and the neurons that are physical neighbors also represent topologically close input vectors unlike the map on Fig.5a, which is tangled and topologically close neighbors do not always share physical proximity.

The algorithm can now be formalized. Each neuron in the network has a weight or reference vector $\xi = [x_1, x_2, \dots, x_m]$ where x_i is an individual attribute of ξ . The neurons are gradually organized over an n-dimensional input space V^n . Each input $\vartheta = [\mu_1, \mu_2, \dots, \mu_n] \in V^n$, where μ_i is an attribute in ϑ , has the same number of attributes as the weight vector ξ of each neuron in the network.

Once the network is initialized, the input is presented sequentially to the network. The best-matching unit in the network is determined by comparing each neuron to the current input based on Euclidean distance, with the winner being the neuron closest to the input.

$$c = \arg_i \min \{\xi_i - \mu_i\} \quad (1)$$

where c is the best-matching unit.

The winning neuron and all neurons falling within the neighborhood radius for an iteration i update their reference vectors to reflect the attraction to θ . To change the reference vector of a neuron, the following equation (which is a very common SOM learning equation) is used:

$$\xi(t+1) = \xi(t) + h(t)\alpha(t)(\theta - \xi) \quad (2)$$

where t is the current iteration and $\alpha(t)$ is a monotonically decreasing learning rate, and $h(t)$ is the neighborhood function.

2.3 Visualizing a SOM

In low dimensional patterns, such as one- or two-dimensional, the input and the SOM can be visualized by using positions of pixels. However, when scaling to three, or five dimensions, pixels can be used for dimensions that represent two-dimensional space, but the remaining one or three attributes in this case would be represented by gray scale or RGB, respectively. This, however, implies that the visualization of the data, and the SOM can be expressed in x , y and color values.

When dealing with complex data that is not directly visualized, or even very high dimensional data (e.g. greater than 10) visualization becomes an issue. One of the methods used for showing the lattice structure style network for high dimensional data, is to use a dimensionality reduction or dimensionality mapping technique. One of the simplest, and most well known is Sammon's mapping (Eq.3) (Sammon 1969).

$$\left(\sum_{(i,j) \in A^2}^{i \neq j} d_A(i,j) \right)^{-1} \sum_{(i,j) \in A^2}^{i \neq j} \frac{(d_A(i,j) - d_V(w_i, w_j))^2}{d_A(i,j)} \quad (3)$$

This mapping measure effectively takes the distances between high dimensional objects, e.g. neurons, and allows them to be plotted in two-dimensions, so the researcher may visually inspect the geometric relations between neurons. Often, when the number of inputs is very high, or patterns are non-distinct and complex, the visualization does not include input data, rather, just the neurons.

To form the lattice structure correctly, the connections between known neighbors should be illustrated. The neurons are often recorded in one- or two-dimensional arrays, to allow the physical neighbors of each neuron to be recorded.

3. Growing Self-organizing Algorithms

3.1 Fritzke's growing SOM variants

Self-organizing Maps, as introduced by Kohonen, are static sized network. The obvious disadvantage to the predetermined number of neurons is that number is either not high enough to adequately map the input space or too high, thus leaving many neurons underutilized. SOM being trained without supervision, is not expected to know the input space characteristics apriori. Hence, a topology which allows the addition/removal of neurons, is a logical step in the development of self-organization. Fritzke introduced three architectures in the 1990's - growing grid (GG), growing cells (GC), and growing neural gas (GNG) (Fritzke, 1992, 1993a, b, c, 1994, 1995). All three start with minimal number of neurons and add neurons as needed. The need is based on an age parameter, while an error parameter determines which neuron will be joined by a new neighbor (GC, GNG) or a new set of neighbors (GG).

In the case of GC, once a neuron with the highest error value is selected, its longest connection (or edge) is replaced by two edges and a neuron is added to facilitate their connection (Fig.7).

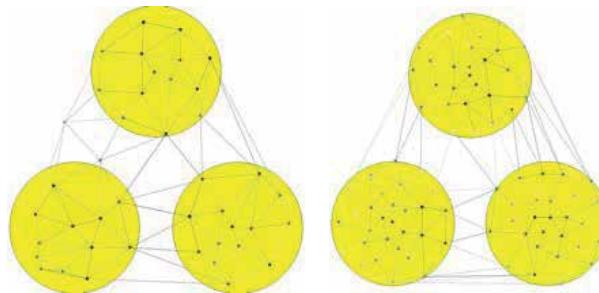


Fig. 7. Growing cells at: a) 3333 iterations; b) 15000 iterations

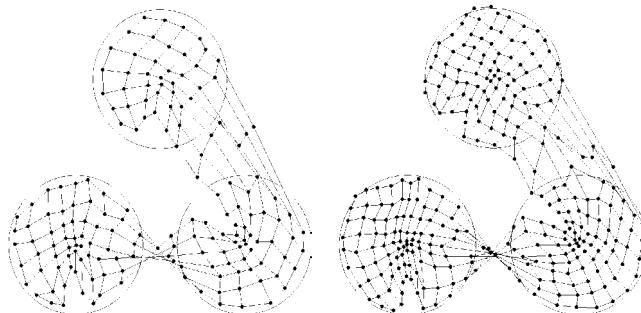


Fig. 8. Growing grid at: a) 33333 iterations; b) 100000 iterations

GG also utilizes an age parameter for the neurons. Every time a neuron wins, its age is incremented. At intervals, the neuron with the most wins is slated to receive new neighbors and the furthest direct topological neighbor from the selected neuron is found. GG utilizes a rectangular grid topology, which is maintained during growth. If these two neurons are in the same row of neurons, a column will be added between the two – thus affecting neurons

in other rows. If the selected neurons are in the same column, then a new row will be added between them – thus affecting neurons in other columns (Fig. 8).

In the third Fritzke contribution – the GNG – a similar concept to GC is observed. However, the connections between the neurons are assigned the age parameter. Therefore, GNG adds and removes connections, based on neuron selection. GNG is also capable of attaining configurations with multiple networks (Fig. 9).

One of the major drawbacks to GNG, as well as other growing methods, is that the number of nodes are ever increasing. This is, in part, compensated for by an extension to GNG, Growing neural gas with utility (GNG-U), which features removal of neurons based on the criterion of low probability density in the underlying input space “beneath” the neuron. The downfalls to these methods are than they do not exhibit incremental learning – a problem that is discussed in a later section on ESOINN.

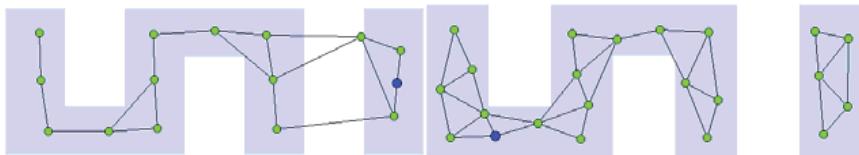


Fig. 9. Provided by (DemoGNG); At 7500 iterations (left) the map is starting to take the form of the input; At 12000 iterations (right) the map is a much better topological representation of the given input.

Fritzke’s growing methods can be explored by the reader interactively at the DemoGNG website (DemoGNG), provided by the Institut für Neuroinformatik.

3.2 ParaSOM

The ParaSOM is architecture developed by the authors (Valova, Szer, Gueorguieva, & Buer 2005), which shares several similar attributes with the classic SOM and the growing architectures developed by Fritzke. However, it improves the effectiveness of the network and allows for different approaches to cluster analysis and classification, which will be demonstrated in later sections. The unique characteristics of the architecture include: parallel input processing, the adoption of cover region to manage the influence area of a neuron, and network growth, which is inspired by the GC architecture.

The ParaSOM is designed to process the entire input space in parallel. The classic SOM presents the network with one input at a time and determines the winner to move closer to that input. With the ParaSOM, however, the entire input space is presented to the network at the same time. Therefore, multiple neurons can adapt to nearby inputs independently of other neurons. This trait helps the network in recognizing patterns that it has already learned. For instance, imagine the network adapted itself to a particular pattern A, and another pattern B, that is very similar to A, is presented to the network. Because the neurons process the input space independently of each other, the ones that are already covering the pattern (most of them, since A is very similar to B) will not move. As a result, adapting to B is much faster, as the network only needs to learn the small differences between A and B.

ParaSOM features the following parameters that are unique to the architecture. The network produces a cover matrix, which represents the approximation of an input pattern made by

the network in its current state. The cover matrix calculation is based on the cover region, which is an attribute of every neuron. This is the area surrounding the point in space where the neuron exists and is considered to be the region of space that the neuron is covering. The move vector is another neuron attribute, which indicates the amount a neuron should move and in which direction is calculated, and added to the neuron's reference (weight) vector. The age parameter, which represents for how long a neuron has been well or poorly positioned is closely related to the inertness of a neuron. The inertness is the measure of how effectively a neuron covers the input space. Both, age and inertness determine when and where the network should grow or contract.

The ParaSOM in action is illustrated in Fig.10. The network is initialized with a minimal number of neurons, which have large cover regions (denoted by the large circles in Fig.10a). As the input space is shown to the network, the move vectors are calculated along with the other parameters gauging the performance of individual neurons (i.e. age, inertness, cover region) and the neurons are moved, new ones are added or some are removed in order to achieve the final comprehensive coverage illustrated in Fig.10c.



Fig. 10. ParaSOM in action: a) randomly initialized with a predetermined number of neurons; b) the network position at iteration 25; c) the network at iteration 500

The formalization of the algorithm will begin with the cover matrix which consists of subtracting a set of cover regions from the input space. Each such region is associated with a single neuron, and represents the contribution of the neuron to the total approximation. The job of the cover region is to weaken the signals of the inputs that are being covered, where the weaker the signal, the better the neuron is covering that region. Each cover region also maintains a radius which decreases in each epoch. The cover matrix is formulated as

$$C = V^m - \sum_{i=1}^s \theta_i \quad (4)$$

with the cover region calculated as

$$\theta_i(x_j) = \begin{cases} f_{m_i}(x), & \text{if } \|x_j - m_i\| < \text{threshold} \\ 0, & \text{else} \end{cases}, \quad \{1 \leq j \leq k\} \quad (5)$$

where the modified Gaussian cover function is defined as

$$f_{m_i}(x) = \exp \left[-\frac{\left[(\mu_1 - x_1)^2 + \dots + (\mu_n - x_n)^2 \right]^2}{\lambda} \right] \quad (6)$$

for radius λ and μ being an attribute of the input vector.

The cover function is utilized in the computation of a cover value, which indicates how well the neuron is covering the inputs and is calculated as

$$c_i = C_i \cdot f_{m_i} \quad (7)$$

where the local cover matrix is represented by

$$C_i = (C + \theta_i) \quad (8)$$

The inertness, is an indicator as to whether the neuron should be moved, removed, or is in a place where new neurons ought to be added nearby. The lower the inertness the better the coverage and hence position of the neuron. The inertness is given by

$$\varphi_i = c_i / c_{\max} \quad (9)$$

where the cover value max is calculated by

$$c_{\max} = \int_{V^m} f_{m_i}(x) dx \quad (10)$$

The network utilizes the inertness to determine whether and where to grow/shrink. A high inertness indicates that the neuron is well positioned and should not move (or move very little), while a low inertness indicates poor positioning and greater neuron movement. Inertness is one of two components that dictate network growth, with age being the second. Each neuron has an age attribute that. When a neuron is well positioned, as determined by a high-inertness threshold, its age is incremented. The same is true with a poorly positioned neuron having its age decremented based on a low-inertness threshold. When a neuron is well (or poorly) positioned for a sufficient period of time, it becomes a candidate to have a neuron added as a neighbor, or to be removed from the network, respectively.

Finally, the move vector, which indicates the amount a neuron should move and in which direction is calculated, and added to the neuron's reference vector. The attractions also affect immediate neighbors of the neuron but to a lesser degree where the amount of movement is proportional to the distance between the neuron and neighbor. The move vector $v_i = (v_{i1}, v_{i2}, \dots, v_{in})$ consists of components $v_{ik} = \int_{V^m} C^i \cdot f_{m_i}(x_k) dx$.

The authors have explored the effect a Hilbert initialization has on ParaSOM. As with the classic SOM, this network is also positively influenced by this mode of initialization. Fig.11 shows some results. Fig.11a features the network at iteration 0, using the same input topology as Fig.10. Fig.11b illustrates the intermediate result at iteration 25, and Fig.11c illustrates the final converged state of ParaSOM at iteration 500, same as the iteration in

Fig.10c. The last point is made to focus the reader attention to the tangled state of the randomly initialized network in Fig.10c. The Hilbert initialization, as the same iteration, features untangled, well-organized network.



Fig. 11. ParaSOM in action: a) Hilbert initialized with a predetermined number of neurons; b) the network position at iteration 25; c) the network at iteration 500

Other investigations with ParaSOM include parallelization (Hammond, MacLean, & Valova, 2006) via message-passing interface (MPI) among 4 worker and 1 director machines (Valova et al., 2009), controlling the parameters of ParaSOM with genetic algorithms (MacLean & Valova, 2007), and more recently, testing and network adjustment for multidimensional input.

3.3 ESOINN

The Enhanced Self-Organizing Incremental Neural Network (ESOINN) (Furao, Ogura, & Hasegawa 2007) represent growing architectures, which are partially inspired by GNG and GNG-U. According to the authors of ESOINN, it addresses the stability-plasticity dilemma (Carpenter & Grossberg 1988), by providing the ability to retain knowledge of patterns it has already learned (stability), while still being able to adapt to, and learn, new patterns that it is yet to be exposed to (plasticity). ESOINN identifies clusters during execution by joining together subclusters that form within a larger cluster. Thus, overlap in multiple clusters can be identified and effectively separated.

Typically with growing architectures, the network grows by adding neurons in sufficiently dense area of the input space. In ESOINN, neurons are added when the current input is adequately distant from the closest neuron. A new neuron is added to the network containing the same (not similar) reference vector as the input.

ESOINN decides on adding a neuron based on similarity threshold. It is basically a dynamic distance measure calculated by the distance of the neuron's neighbors, or, if no neighbors are available, all other neurons in the network. When an input is presented to the network, the first and second winners, or best matching unit (BMU) and second matching unit (2BMU), are determined. The network then decides if a connection between the winner and second winner should be created, if one does not already exist.

In ESOINN, knowledge of the neuron density in a given area of the input space is critical to performing tasks such as creating connections between neurons and detecting overlap in clusters. By being able to measure the density, the network can better determine whether a particular section of the input space is part of a single cluster or of an overlapped section. After detection of overlapped areas, connections between neurons of different subclasses are

removed. This separates the subclasses belonging to different composite classes. This process is performed at regular intervals, where the number of inputs presented to the network is evenly divisible by predetermined integer value.

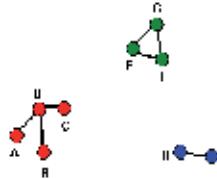


Fig. 12. Neurons in {A, B, C, D} are all connected by paths and therefore are in the same cluster. The same is true with {E, F, G}, and {H, I}. Conversely, A and E have no path to each other, and therefore are not in the same class

Connections in ESOINN are used to identify subclasses. To aide in this identification, they are created and removed from neurons as new data is presented to the network. When a connection is removed between two neurons, a boundary is identified between the different classes that each neuron is a part of. The paths created by connections are also the way that neurons are classified at the end of ESOINN execution. Any given neuron i , and all other neurons that are connected to i by a path, are considered to be in the same class. Neurons that cannot be connected by a path are said to be in different classes (Fig.12).

Connections in ESOINN are created when there is no existing connection between a winner and second winner. In this case, the newly created connection has an age attribute that is set to zero. If a connection already exists between the winner and second winner, the age of that connection is reset to zero. In either scenario, the ages of all existing connections between the winner and its neighbors are increased by one (except the connection between it and the second winner). Deletion of connections occurs when the ESOINN algorithm determines that the current winner and second winner are in different subclasses and those subclasses should not be merged.

ESOINN adds neurons because they represent noisy input which is likely to be distant from relevant patterns. As a result, the input will be outside of the similarity threshold of the winner and second winner, and a new neuron is created. These neurons are undesirable because they are generally placed in low-density areas and can skew the cluster identification. ESOINN removes neurons with two or fewer neighbors utilizing average density value. When a neuron is deleted, all connections associated with it are also removed. This process also occurs after predetermined number of inputs has been presented.

The connection removal and addition features of ESOINN make it very powerful at finding distinct patterns in a wide array of problem domains. ESOINN is a major step forward in unsupervised learning. Since ESOINN addresses the stability-plasticity dilemma (continued learning with no forgetting), it is an algorithm that can be used for varying types of data sets, including overlapping Gaussian distributions.

3.4 TurSOM

TurSOM (the amalgamation of Turing and SOM) is a new variant of the Kohonen Self-organizing Map, introduced by the authors (Beaton, 2008; Beaton, Valova, & MacLean, 2009a, b, c). TurSOM's primary contribution is the elimination of post-processing techniques

for clustering neurons. Its features are inspired in part by Turing's work on unorganized machines (Turing, 1948). Turing's unorganized machines (TUM) represent early connectionist networks, meant to model the (re)organization capability of the human cortex. In Kohonen's SOM algorithm, the neurons are the components of self-organization, whereas with Turing's idea, the connections also fulfil that role. In TurSOM, we capitalize on both methods of self-organization.

While the neurons of TurSOM adhere to the same learning rules and criteria of the standard SOM, the major differentiating feature of TurSOM is the ability to reorganize connections between neurons. Reorganization includes the removal, addition, or exchanging of connections between neurons. These novelties make TurSOM capable of identifying unique regions of input space (clustering) during execution (on-the-fly), as demonstrated in Fig.13. The clustering behavior is achieved by allowing separate networks to simultaneously execute in a single input space. As TurSOM progresses, connections may be removed, or exchanged – causing a network to split into two networks, and two into three or four, and so on. Additionally, when separate networks get closer to one another they may join to form a single network.

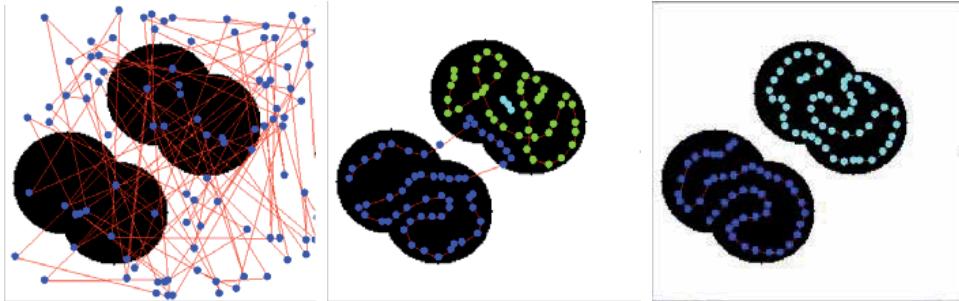


Fig. 13. TurSOM on connection reorganization: a) TurSOM at initialization; b) TurSOM at 250 iterations – exemplary of TurSOM reorganizing connections; c) TurSOM at 350 iterations – exemplary of TurSOM identifying unique patterns

In order for TurSOM to achieve the behavior it exhibits, several new mechanisms are introduced to the Kohonen SOM.

In SOM algorithms, there is a neuron learning rate. The point of the neuron learning rate is to decrease the movement of winning neurons (and subsequently their neighbors) as time progresses. As a SOM adapts to input, it should require less drastic organization, i.e., smaller movements.

Similarly, TurSOM introduces a connection learning rate (CLR), which regulates the reorganization of connections as TurSOM progresses. The CLR is a global value controlling the maximum allowable distance between two neurons. If the distance between any two neurons exceeds the CLR, they must disconnect. CLR is computed as follows:

$$CLR = Q_3 + (i \times (Q_3 - Q_1)) \quad (11)$$

The CLR formula is derived from the upper outlier formula from box-plots (a statistical technique of measuring distribution by analyzing four quartiles). In CLR, the x in Q_x represents which quartile it is, and i , is an incrementing value as time progresses. The data

being measured (for the quartiles), is the length of all connections available in the current input space. The CLR is instrumental to the reorganization process in TurSOM as it effectively decides which connections are unnecessary.

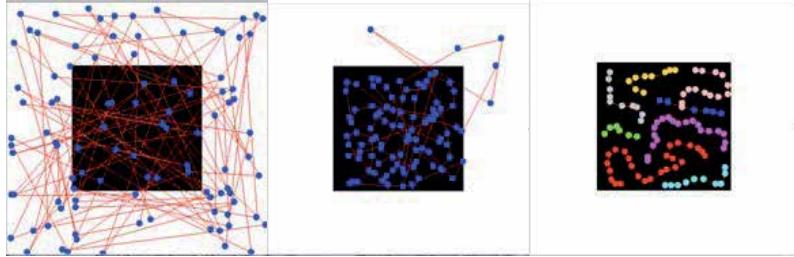


Fig. 14. CLR in TurSOM: a) the square pattern with random initialization; b) the first 50 iterations of TurSOM, where its behavior is the same as a SOM; c) CLR has been active for 100 iterations and a rapid, and sudden reorganization of connections is evident

Fig.14a, b and c demonstrates a simple solid pattern for the first 150 iterations of TurSOM. The CLR determines which connections will be eliminated. The connections that are not considered optimal are removed and as evident by the figure, the removed connections were negatively impacting the network.

So far, we have described how TurSOM may separate into different networks, but we have not addressed how two networks can rejoin into one. The neuron responsibility radius (NRR), inspired by ParaSOM's neuron cover region (addressed in section 3.2), becomes active in TurSOM when two neurons disconnect from one another. However, there is one requirement for networks that disconnect – they must be of size three or greater. Empirically, it has been shown (in TurSOM) that networks smaller than three (i.e. 2 or a single neuron) become “pushed aside” for other neurons that are active in a network. A neuron with an active radius still has one neighbor.

The neuron responsibility radius, is effectively a “feeler”, actively searching for other “free” neurons with similar features. To calculate the NRR, the following formulae are used when the dimensionality of input space is even:

$$r_e = \left[\frac{\rho}{e} \right]^{\frac{1}{\delta}} \quad (12)$$

$$e = \left(\frac{\delta}{2} \right)^{-1} \times \pi^{\frac{\delta}{2}} \quad (13)$$

If the dimensionality is odd:

$$r_o = \left[\frac{\rho}{o} \right]^{\frac{1}{\delta}} \quad (14)$$

$$o = \left(\frac{2^\delta \times \frac{\delta-1}{2}!}{\delta!} \right) \times \pi^{\frac{\delta-1}{2}} \quad (15)$$

where δ represents the number of dimensions, and ρ represents the number of inputs a neuron is responsible for. ρ is calculated by dividing the number of neurons, by the number of inputs. To follow along with the example provided in the previous section on connection learning rate, the following Fig.15) demonstrate the remaining iterations of TurSOM, where the effects of the NRR are seen.

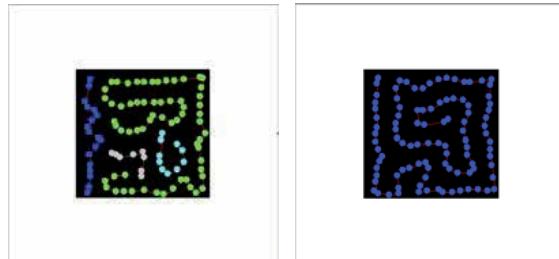


Fig. 15. The effects of NRR: a) demonstrates the reconnection process, which is governed by the NRR; b) The single pattern in an optimal mapping, where the Peano-like curve of the network must be noted

As demonstrated in Fig.15, the NRR determines the reconnection process, and works in cooperation with the CLR (the disconnection process). TurSOM also provides for a neuron to sense nearby neurons that are *better suited* to be connected neighbors than current neighbors. Simply stated, this is a check that neurons perform by knowing the distance to their neighbors, and knowing of other neurons in the network that are nearby. This process is considered to be a part of the reorganization process.

Similar to Frtizke's growing grid algorithm, TurSOM has a growth mechanism. TurSOM begins as a one-dimensional chain, which upon *convergence*, will spontaneously grow (SG) to two-dimensional grids. The term *convergence* is used loosely here to mean a network reaching a fairly stable representation of the input where further computation would not benefit the network significantly. During the spontaneous growth phase, connection reorganization (which implies network splitting and rejoining) is turned off. Presumably, at this point, the one-dimensional networks have settled to satisfactory positions, and do not require further adaptation. The growing process is demonstrated in Fig.16. Fig.16a illustrates the input pattern. The converged one-dimensional SOM is shown in Fig.16b. Finally, the SG is demonstrated in Fig.16c, where it is evident that each one-dimensional network grows independently.

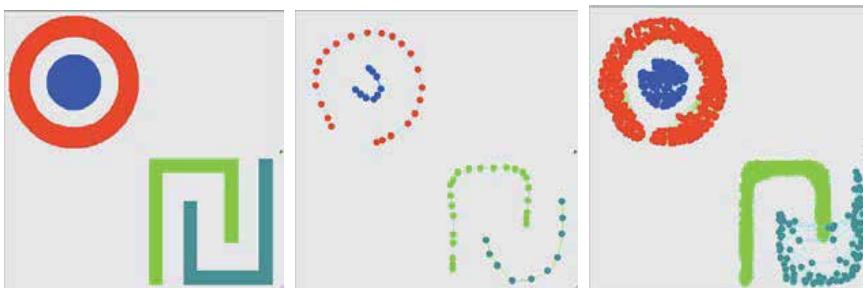


Fig. 16. TurSOM in action: a) Input space with 4 distinct patterns, which are five-dimensional data (X, Y, R, G, B); b) TurSOM in one-dimensional form mapping each of the distinct patterns; c) TurSOM with SG for a better representation of the patterns

TurSOM's growing capabilities are an instrumental part facilitating the performance of the network. Often times, one-dimensional networks do not represent the underlying data well enough. Two-dimensional networks have a better representation, or, a better *resolution*.

Finally, the TurSOM algorithm can be summarized in the following steps:

- a) Select Input
- b) Find best-matching unit (BMU)
- c) Update BMU and BMU's neighbors
 - 1) Record the distances between all connected neighbors
- d) Check lengths of all connections (Step c.1)
 - 1) If connection is too large
 - Disconnect neurons
 - Update Connection Learning Rate
 - Activate Neuron Responsibility Radius
- e) Check neuron physical locations
 - 1) If neuron A is a neighbor of B, but not C (which is a neighbor of B), but A is closer to C than B, switch connections - thereby changing neighbors
- f) Check neuron responsibility radius for proximity to other neurons
 - 1) Reconnect neurons that have overlapping NRR
- g) If TurSOM has reached convergence
 - 1) Spontaneous Growth

3.5 Modular Network Self-organizing Map

While not a growing architecture, a very recent SOM architecture called the modular network Self-Organizing Map (mnSOM) (Tokunagaa & Furukawa, 2009) is mentioned here. This architecture is a hybrid approach to neural network computing. The mnSOM architecture consists of a lattice structure like that of a typical SOM. Additionally, the neurons in the SOM behave in a similar self-organizing fashion. However, each neuron is composed of or "filled with" a feed-forward network, such as a multi-layer perceptron (MLP).

The major difference between SOMs and feed-forward networks, is that SOMs learn the *topology* or structure of data. Feed-forward architectures learn *functions* about input.

The effective outcome of this network is that it *self-organizes function space*. That is to say, when presented with various types of input patterns where functional knowledge might be very important, mnSOM is able to topologically order functions based on similarity.

4. Methods for SOM analysis

Self-organizing maps are powerful analytical tools. Visualization is often employed to analyze the resulting topological map. However, sometimes networks do not represent optimal mappings. This can skew the understanding, or even representation of the data that is supposed to be visualized. In this section we provide methods of analyzing the *quality* of a SOM network. Commonly, these techniques are used *post-execution*, in order to analyze how well the SOM *converged* to the given data. A survey of SOM quality measures can be found in (Pöhlbauer 2004).

4.1 Quantization Error

Quantization error (QE) is a simple measure used in other fields, including clustering and vector quantization as a technique for verifying that inputs are with their proper (or best suited) clusters. As SOMs perform clustering, QE can be utilized. However, one major draw back is that QE does not address the *quality of organization* of the network. Rather, it measures neuron placement to inputs.

Quantization error is measured by computing the average distance from inputs to their appropriate outputs. One point to note about this measure, is that when the number of neurons is decreased or increased for the same input space, the value acquired by quantization error is increased or decreased respectively. Effectively, more neurons mean a smaller error, and vice versa for less neurons. The QE is calculated by computing the average distance from inputs to their associated neuron. Short pseudocode is given below:

```

uniquely number all neurons
for each input
    find best-matching unit (BMU); aka neuron
    array[BMU#][1] = array[BMU#][1] + distance from input to BMU
    array[BMU#][2] = array[BMU#][2] + 1;
end
for each neuron as x
    error[x] = array[x][2] / array[x][1]
end

```

4.2 Topographic Error

Topographic error (TE) measures the *quality of organization* of SOMs, and provides information of *how well organized neurons are with respect to other neurons*. This measure is used to see if neurons are correctly identified as topological neighbors, with respect to inputs.

Conceptually, TE is attempting to give an idea as to how *twisted*, or *tangled* a SOM network is. An example of a 2-dimensional pattern, with a 1-dimensional map is shown in Fig.17. Topographic error is represented as a value between 0 and 1, where 0 indicates no topographic error, *therefore, no tangling*, and 1 would indicate maximum topographic error or *complete tangling*.

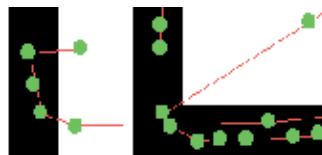


Fig. 17. This pattern shows (in the bottom right) a 1-dimensional network that intersects, or crosses connections. Effectively, this network is tangled, as there are *more appropriate neighbors* for some of the neurons.

Topographic error is computed as follows:

```

error = 0
for each data sample
    find best-matching unit (BMU)
    find second best-matching unit (2BMU)

```

```

if BMU is not a lattice neighbor of 2BMU
    error = error + 1;
end
error = error / number of neurons;

```

4.3 Topographic Product

Topographic product (TP), introduced by (Bauer & Pawelzik, 1992), is a measure to indicate if a SOM is too large or too small for the input it is representing. TP measures the *suitability* and *size appropriateness* of a map, for a given set of input. Two primary variables are utilized for the computation of TP, Q_x and P_x . Q_x is a ratio of distances found in input and output spaces, and P_x is a multiplicative normalization of its respective Q_x value. Below are the initial steps for TP:

Step 1: For the weight (in input space) (\mathbf{w}) of neuron j (\mathbf{w}_j), find the k^{th} :

- a. Closest data in input space, as distance d_1^V
- b. Closest neuron in output space, as distance d_2^V

Step 2: For the neuron j , find the k^{th}

- a. Closest neuron in output space, as distance d_1^A
- b. Closest data in input space, as distance d_2^A

Step 3: Create two ratios:

$$Q_1(j,k) = d_1^V / d_2^V$$

$$Q_2(j,k) = d_1^A / d_2^A, \text{ where } k \text{ represents an iterative value.}$$

When using k in the SOM, the iteration occurs through all other neurons besides j (steps 1a and 2a). Similarly, when calculating Q_1 , the iteration occurs through all inputs, excluding \mathbf{w}_j if \mathbf{w}_j is equal to one of the inputs (steps 1b and 2b).

These two values, Q_1 and Q_2 , optimally would be equal to 1, if and only if neighbors are correctly preserved and suitably organized. However, Bauer and Pawelzik point out that this is far too sensitive. A normalization of Q_1 and Q_2 is required, via Π function (pseudo code provided):

```

Px(j,k) = 1;
for each neighbor of a neuron j, represented by k
    Px(j,k) = Px(j,k) * Qx(j,k)
end
Px(j,k) = power(Px(j,k), (1/k))

```

x of P_x , and Q_x are either 1 or 2, defined from the previous steps. At this point, $P_1 \geq 1$, and $P_2 \leq 1$, as P_1 is a value created from all the data in input space, based on the weights of all neurons. If there is a 1-to-1 mapping of neurons to input, then this value should be 1. Additionally, P_2 will be less than or equal to one, because it is a representation of neighboring neurons in the output space. This occurs because the denominator of Q_2 comes from input space distances and the numerator comes from neurons distances.

However, having two numbers to explain a mapping is not desirable, so Bauer and Pawelzik introduce P_3 (provided below in pseudo code):

$P_3(j,k) = 1;$

for each neighbor of a neuron j , as k
 $P_3(j,k) = P_3(j,k) * (P_1(j,k) * P_2(j,k))$
end
 $P_3(j,k) = \text{power}(P_3(j,k), (1/2k))$

P_3 is normalized. The relationship of P_1 and P_2 is *inverse*, thereby giving way to these rules:

- 1: $P_3 > 1$ means the map is too large, $P_1 > (1/P_2)$
- 2: $P_3 < 1$ means the map is too small, $P_1 < (1/P_2)$

The final step in topographic product is computing an average of the values already obtained, when using all neurons in a SOM:

```

 $P = 0;$ 
for each neuron, as  $j$ 
    for each neighbor as  $k$ 
         $P = P + \log(P_3(j,k))$ 
    end
end
 $P = P/(N*(N-1)) // \text{where } N \text{ is the number of neurons}$ 
```

All values of P that deviate from 1, should be of concern. The same rules apply to P as do P_3 , concerning deviation from 1. The formulaic view of P_3 is provided by Eqs. (16) and (17).

$$P_3 = \left(\sum_{l=1}^k \frac{d^V(w_j, w_{n_l^A(j)})}{d^V(w_j, w_{n_l^V(j)})} \cdot \frac{d^A(j, n_l^A(j))}{d^A(j, n_l^V(j))} \right)^{1/2k} \quad (16)$$

$$P = \frac{1}{N(N-1)} \sum_{j=1}^N \sum_{k=1}^{N-1} \log(P_3(j,k)) \quad (17)$$

4.4 Other Measures

We have presented three measures of SOM that evaluate fundamental aspects of SOM quality, namely, correct neuron to input positioning (QE), network organization quality (TE), and suitability of map size (TP).

However, there are several measures beyond these that attempt to combine these fundamental aspects, or measure other characteristics of SOMs. Some of these measures include Trustworthiness and Neighborhood Preservation (Venna & Kaski, 2001), which aim to measure data projection relations, by comparing input data relations to output data relations; and Topographic Function (Villmann et al., 2007), a measure which accounts for network structure, and neuron placement.

5. Pattern identification and clustering

Over the years many methods of analyzing the patterns of the neurons of SOMs have been introduced. One of the simplest methods is the gray scale clustering presented by Kohonen in his book, on the poverty map data set (Kohonen, 1995). Kohonen's example colors distances between nodes a shade of light gray, if the nodes are close, or dark gray, if the nodes are far. However, visual methods leave interpretation up to the reader. In this section

we present two methods of analyzing and identifying patterns exhibited by the neuron configurations of SOMs. These are methods for post-convergence analysis. When a SOM converges, it is not always necessary to perform any post-processing techniques, especially in lower dimensionality. At the time of convergence, what we do know is that each neuron has found a suitable region of space, where it is representing a given amount of inputs. Exactly what inputs is not always clear unless another technique is used (one technique to map inputs to neurons is QE). Additionally, there may be a relationship that exists between neurons. This section will explain methods of measuring similarity and the relationships between neurons.

5.1 PCA

Principal components analysis (PCA) is a well-established statistical technique, used in a variety of fields on high-dimensional data. The primary goals of PCA are dimensionality reduction and explanation of covariance (or correlation) in variables. Effectively, PCA provides linearly separable groups, or clusters within high-dimensional data along a given dimension (variable). Additionally, the principal components computed by PCA can identify which variables to focus on, i.e. which variables account for the most variance. Variables are determined to be unnecessary when they do not explain much variance. For a detailed explanation on PCA and how to implement it, please see (Smith 2002, Martinez, & Martinez 2005).

PCA can be used as a pre- (Kirt, Vainik & Võhandu, 2007; Sommer & Golz, 2001) and post- (Kumar, Rai & Kumar 2005; Lee & Singh, 2004) processor for SOM. Additionally, a SOM has been created to combine the capabilities of both PCA and SOM (López-Rubio, Muñoz-Pérez, Gómez-Ruiz, 2004).

When analyzing a SOM for potential clusters, understanding the relationship among neurons usually presents great challenge. This analysis can become difficult when analyzing a converged map when there are very few (small network) or very many (large network) neurons. Additionally, it may be more useful to ignore certain variables prior to executing a SOM on a data set. This is where PCA becomes a very useful tool.

It is important to note that PCA is a linearly separable unsupervised technique. Effectively, a vector is drawn from the origin to a point in space and it is determined that the groups to one side and the other are significantly distinct (based on a given variable or dimension). SOM on the other hand, is non-linear, and each neuron can be thought of as a centroid in the k-means clustering algorithm (MacQueen, 1967). Neurons become responsible for the input that they are closest to, which may be a spheroid, or even a non-uniform shape.

In the case PCA is performed *prior to* executing a SOM on a data set, it will be determined which variables, or dimensions, are most important for a SOM, and now the neurons in a SOM will have less weights than the original data set. In case PCA is performed *after* a SOM has executed, the method will determine which variables in the weights of the SOMs are most important. This will help explain which neurons are more similar than others, by contrast to other methods like distance measures and coloring schemes. In summary, PCA helps eliminate attributes that are largely unnecessary.

5.2 ParaSOM modifications

The ParaSOM architecture takes a unique approach to performing cluster identification. It relies heavily on the features of the network and the behavior it exhibits because of those features (Valova, MacLean & Beaton, 2008b).

When the network is well-adapted and near the end of execution, the cover regions of the neurons are generally small and covering their respective sections of the input space precisely. Therefore, in dense regions the neurons should be plentiful and in very close proximity. A key property of the network at convergence is that the distances between intra-cluster neurons will likely be much smaller than the distance between inter-cluster neurons. This is the central concept of the cluster identification algorithm that ParaSOM takes advantage of.

Once convergence takes place, the network will perform clustering in two phases. The first phase utilizes statistical analysis to initially identify clusters. The second phase employs single and complete linkage to combine any clusters that may have been separated, but are in close enough proximity to be considered a single cluster.

In order to determine the minimal distance between neurons in different clusters we make use of the mean of the distances between neighboring neurons, \bar{x} , as well as their standard deviation, σ . The standard deviation is used to determine how far away from the mean is considered acceptable in order for a neighbor to be labeled in the same cluster as the neuron in question.

The overwhelming majority of the connections between neighbors will be within clusters. Therefore, the number of standard deviations away from the mean connection distance that a certain majority of these connections is within will be a good indicator of an adequate distance threshold.

To discover the initial clusters, the mean of the distances between neighbors is determined through iteration on all neurons. Following that, the standard deviation of the distances between neighboring neurons is computed via Eq. (18).

$$\sigma = \sqrt{\sum_{i=1}^n (\bar{x} - d_i)^2} \quad (18)$$

where d_i is the distance between a pair of neighboring neurons.

Further, determine how many neuron pairs lie within $\bar{x} + m\sigma$ standard deviations, for each $m \in \{1\dots n\}$. Based on some threshold α , where $0.0 \leq \alpha \leq 1.0$, determine the minimum distance $\bar{x} + m\sigma$ that the percentage of neurons specified by α are within. This distance will become the initial cutoff threshold κ . Finally, iterate through the neurons one final time to determine where new clusters are formed. When a neuron's distance to a neighbor exceeds κ , add the neighbor to new cluster.

The method used to determine κ is based on Chebyshev's theorem (Sternstein, 1994), which states that at least $1 - \frac{1}{p^2}$ of the values in a set of data lie between p standard deviations of the mean. This theorem is applicable to generalizations that are valid for any set of data. However, the behavior of the ParaSOM requires heuristic modifications to the theorem. We modify the original theorem to obey the following principle:

$$\kappa = \bar{x} + p \cdot \sigma \quad (19)$$

where p satisfies the minimum $p \cdot \sigma$ from \bar{x} that the percentage of neurons specified by α lie within. Since the ParaSOM provides a narrow normal distribution of distances between neighbors, the determination of κ for it to be fine-tuned provides effective adaptation to any input space distribution.

There are situations where separating clusters based on distances between neighbors leads to undesired results. Ideally, the cutoff threshold κ should be adequate to accurately determine cluster membership. However, a number of factors that influence the location of neurons can cause erroneous decisions by the method described above. There are cases where the distance between neighboring neurons is outside the cutoff threshold because of the insertion of neurons between them (Fig.18).

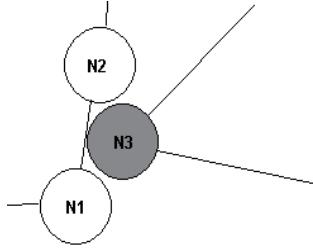


Fig. 18. Neurons N1 and N2 are neighbors, but their distance exceeds the cutoff threshold because N3 is between them.

The cluster identification algorithm takes this into account and will recombine clusters that are originally determined to be disjoint, but have sufficient similarity to merge. This cluster recombination is facilitated by two techniques used in tandem: single and complete linkage. Single linkage is a technique that links together clusters based on the minimal distance between any one element in one cluster and any one element in a second cluster. Single linkage recombination sometimes has a tendency to chain together clusters and produce potentially undesirable results. Being aware of this tendency, ParaSOM's recombination algorithm uses a threshold value β to make sure the linkage distance between clusters does not exceed this value. The β threshold determination also relies on the ParaSOMs tendency to create an abundance of tightly packed neurons in dense input regions.

Looking again at Fig.18, let us assume that the distance separating N1 and N2 is greater than κ . Therefore, according to the initial clustering, N1 and N2 are in different clusters. Let us also assume that N2 and N3 are in the same cluster. Because N1, N2, and N3 are all in such close proximity, it makes sense that they should be in the same cluster. When single linkage occurs, the distance between cluster containing N1 and the cluster containing N2 (or N3) will be well within the β threshold, and these clusters will be combined.

Since the ParaSOM covers space as effectively as possible, in a mature network adaptation of the input space the distance between neighboring clusters that should be combined will be very close to $2r$, where r is the cover region radius. As a result, any reasonable β value will recombine these two clusters. For the ParaSOM, β is set to $\bar{x} + \sigma$.

The second aspect of the cluster recombination process is using complete linkage to rejoin stray clusters that were mishandled by the initial clustering and not accounted for by single linkage. Complete linkage is utilized by taking the maximum distance, or dissimilarity, between elements of two clusters. Complete linkage is employed to join small clusters and clusters containing only a single neuron. These are cases where the clusters may have been too distant for single linkage to recombine them. Generally, when complete linkage is used, there is a chance that the quality of clusters may degrade if the farthest neighbors that join two clusters together happen to be closer to other clusters than they are to their own. However, such an event will not be the case. Since complete linkage measures cluster distance based on the furthest neighbors, the threshold that determines if clusters are recombined is more lenient than β . For complete linkage, we use a threshold of κ , which was introduced previously.

Following the methodology description, we provide two examples of the pattern identification capabilities of ParaSOM. Figs.19 and 20 take the network through its progress in adjusting to the input topology and proceeding to evaluate the number of distinct regions the input space features. The pattern in Fig.19 is clearly requiring one class, however due to the two very distant shoulders, presents problems for most unsupervised clustering methods. Not so for ParaSOM, which recognizes and configures the final result into one cluster. The pattern in Fig.20 presents the difficulty of the two clusters being very close together. Within 600 iterations and the above described algorithm, ParaSOM clearly identifies the correct number of clusters.



Fig. 19. Identification of patterns with ParaSOM: a) network at 100 iterations; b) network state at 400 iterations; c) 600 iterations; d) inspite of the input distribution, the single object is identified

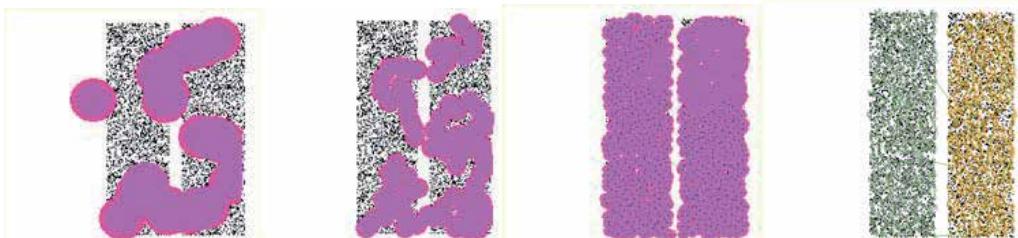


Fig. 20. Identification of patterns with ParaSOM: a) network at 100 iterations; b) network state at 300 iterations; c) 600 iterations; d) the two distinct regions are recognized

6. Conclusions

In summary, we present a number of SOM algorithms, with a primary focus on growing architectures. The goal of this chapter is to introduce, in chronological order, the

development of growing SOM-based techniques, featuring various variants based on Kohonen and Fritzke algorithms. The authors present their two major contributions - ParaSOM and TurSOM along with the ESOINN network which features incremental learning and cluster identification. We also present methods for SOM analysis which we have used in our research to facilitate the comparison between the methods we develop and the state-of-the-art algorithms presented by other researchers.

As a point of future work, we are continuing the development of both ParaSOM and TurSOM for various applications and further improve the algorithms in terms of complexity of performance and implementation.

7. References

- Bauer, H. U. and Pawelzik, K. R., (1992) Quantifying the neighborhood preservation of self-organizing feature maps, *IEEE Trans. on Neural Networks*, vol. 3, no. 4, July 1992.
- Beaton, D. (2008). *Bridging Turing unorganized machines and self-organizing maps for cognitive replication*. Master's Thesis, University of Massachusetts Dartmouth.
- Beaton, D., Valova, I., & MacLean, D. (2009a). TurSOM: a Turing inspired self-organizing map, *International Joint Conference on Neural Networks*.
- Beaton, D., Valova, I., & MacLean, D. (2009b). Growing Mechanisms and Cluster Identification with TurSOM, *International Joint Conference on Neural Networks*.
- Beaton D., Valova, I., & MacLean, D. (2009c). The Use of TurSOM for Color Image Segmentation. *IEEE Conference on Systems, Man and Cybernetics*.
- Buer, A. (2006). *Initialization of self-organizing maps with self-similar curves*. Master's Thesis, University of Massachusetts Dartmouth.
- Carpenter, G. A., & Grossberg, S., (1998) The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network, *Computer*, vol. 21, no. 3, pp. 77-88.
- Carpenter, G.A., & Grossberg, S. (1990) ART 3: hierarchical search using chemical transmitters in self-organizing pattern recognition architecture, *Neural Networks*, 3:129-152.
- Chien-Sing, Lee & Singh, Y.P., (2004) Student modeling using principal component analysis of SOM clusters, *IEEE International Conference on Advanced Learning Technologies*, 30:480-484.
- Fritzke, B. (1992). Growing cell structures -- a self-organizing network in k dimensions. In I. Aleksander, & J. Taylor (Eds.), *Artificial neural networks II*. North-Holland, Amsterdam.
- Fritzke, B. (1993a). Growing cell structures – a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7, 1441-1460.
- Fritzke, B. (1993b). Supervised learning with growing cell structures. In *Proceedings of Neural Information Processing Systems*, pp. 255-262.
- Fritzke, B. (1993c). Kohonen feature maps and growing cell structures – a performance comparison. In *Proceedings of Neural Information Processing Systems*, pp. 123-130.
- Fritzke, B. (1994). A growing neural gas network learns topologies. In *Proceedings of Neural Information Processing Systems*, pp.625-632.
- Fritzke, B. (1995). Growing grid – a self-organizing network with constant neighborhood range and adaptation strength. *Neural Processing Letters*, 2, 5: 9-13.

- Furao, S., Ogura, T., & Hasegawa, O. (2007). An enhanced self-organizing incremental neural network for online unsupervised learning. *Neural Networks*, 20, 8:893-903.
- Furao, S., Hasegawa, O. (2006). An incremental network for on-line unsupervised classification and topology learning. *Neural Networks*, 19, 1:90-106.
- Hammond, J., MacLean, D., & Valova, I., (2006) A Parallel Implementation of a Growing SOM Promoting Independent Neural Networks over Distributed Input Space, *International Joint Conference on Neural Networks (IJCNN)*, 1937 - 1944.
- Institut für Neuroinformatik from DemoGNG website: <http://www.neuroinformatik.ruhr-uni-bochum.de/VDM/research/gsn/DemoGNG/GNG.html>
- Kirt, T., Vainik, E., Võhandu, L., (2007). A method for comparing self-organizing maps: case studies of banking and linguistic data. In: *Local Proceedings of the 11th East-European Conference on Advances in Databases and Information Systems*, 107 - 115.
- Kohonen, T. (1995) *Self-Organizing Maps*, Springer, Berlin, Heidelberg, New York.
- Kumar, D., Rai, C.S., Kumar, S., (2005) Face Recognition using Self-Organizing Map and Principal Component Analysis, *Neural Networks and Brain*, 2005. *ICNN&B '05. International Conference on*, pp.1469-1473.
- López-Rubio, E., Muñoz-Pérez, J. & Gómez-Ruiz, J.A., (2004) A principal components analysis self-organizing map, *Neural Networks*, 17:261-270.
- MacLean, D., (2007) Clustering and classification for a parallel self-organizing map. Master's Thesis, University of Massachusetts Dartmouth.
- MacLean, D., & Valova, I. (2007) Parallel Growing SOM Monitored by Genetic Algorithm, *International Joint Conference on Neural Networks (IJCNN)*, 1697-1702.
- MacQueen, J., (1967) Some methods for classification and analysis of multivariate observations, *Proc. Fifth Berkeley Symp. on Math. Statist. and Prob.*, 281-297.
- Martinez, W.L., & Martinez, A.R., (2005). *Exploratory Data Analysis with MATLAB*. Chapman & Hall/CRC Press, Boca Raton.
- Polani, D. (2002). Measures for the organization of self-organizing maps. In U. Seiffert, & L. C. Jain (Eds.), *Self-organizing neural networks*, Physica Verlag.
- Pöhlbauer, G.,(2004) Survey and comparison of quality measures for self-organizing maps. In Ján Paralic, Georg Pöhlbauer, and Andreas Rauber, editors, *Proceedings of the Fifth Workshop on Data Analysis*, 67-82, Elfa Academic Press.
- Sammon, J.W. Jr., (1969) *A nonlinear mapping for data structure analysis*, IEEE Transactions on Computation, C-18, 401-409.
- Smith, L (2002) A tutorial on Principal Components Analysis. Website: www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf
- Sommer, D., & Golz, M. (2001). Clustering of EEG-Segments Using Hierarchical Agglomerative Methods and Self-Organizing Maps. In *Proceedings of the international Conference on Artificial Neural Networks*, 642-649.
- Sternstein, M., (1994). *Statistics*. Barron's Educational Series, Inc., Hauppauge, N.Y.
- Tokunaga, K. & Furukawa, T., (2009) Modular network SOM, *Neural Networks*, 22:82-90.
- Turing, A.M. (1948) 'Intelligent Machinery'. National Physical Laboratory Report. *Collected Works of A.M. Turing: Mechanical Intelligence*. Ince, D.C. (ed.) (1992), North Holland, Amsterdam.
- Valova, I., Szer, D., Gueorguieva, N., & Buer, A. (2005). A parallel growing architecture for self-organizing maps with unsupervised learning. *Neurocomputing*, 68C, 177-195.

- Valova, I., Beaton, D., & MacLean, D. (2008a) Role of initialization in SOM networks - study of self-similar curve topologies, In *Proceedings of International Conference on Artificial Neural Networks in Engineering* (pp. 681-688).
- Valova, I., MacLean, D., & Beaton, D. (2008b) Identification of Patterns via Region-Growing Parallel SOM Neural Network, *International Conference on Machine Learning and Applications (ICMLA)*, 853 - 858.
- Valova, I., Beaton, D., MacLean, D., Hammond, J., & Allen, J. (2009) NIPSOM: Parallel Architecture and Implementation of a Growing SOM, *The Computer Journal*, Oxford.
- Venna, J., Kaski, S., (2001) Neighborhood preservation in nonlinear projection methods: An experimental study. *Lecture Notes in Computer Science*, 2130:485–491.
- Villmann, T. et al., (1997) Topology Preservation in Self-Organizing Feature Maps: Exact Definition and Measurement; *IEEE Transactions on Neural Networks*, vol 8. no. 2, pp 256-266.

Relational Analysis for Clustering Consensus

Mustapha Lebbah, Younès Bennani and Nistor Grozavu

LIPN - UMR 7030 CNRS, Université Paris 13,

99, avenue Jean-Baptiste Clément

93430 Villejuif.

e-mail:firstname.lastname@lipn.univ-paris13.fr

France

Hamid Benhadda

Thales Land & Joint 160, Bld de Valmy - BP 82

92700 - Colombes cedex.

e-mail:Hamid.BENHADDA@fr.thalesgroup.com

France

1. Introduction

One of the most used techniques among many others in the data mining field is the clustering. The aim of this technique is to synthetize and summarize huge amounts of data by splitting it into small and homogenous clusters such that the data (observations) inside the same cluster are more similar to each other than to the observations inside the other clusters. This definition assumes that there exists a well defined clustering quality measure that quantifies how much homogeneous are the obtained clusters. The aim of this chapter is to expose an original approach to merge different partitions, related to the same data set, which are obtained either by applying different clustering techniques either by the same clustering technique with different parameters. Fusing partitions has been broadly studied and has been given several names, depending on different scientific fields, like machine learning or bioinformatics (Dudoit & Fridlyand, 2003; Kim & Lee, 2007; Monti et al., 2003). Among these names we can quote: consensus clustering, clustering aggregation, clustering combination, fusion of clustering, ...etc. Several studies (Frossiniotis et al., 2002; Minaei-Bidgoli et al., 2004; Strehl & Ghosh, 2002; Topchy et al., 2004; 2005) have pioneered clustering data sets as a new branch of the conventional clustering methodology. In (Topchy et al., 2004) the authors propose a probabilistic formalism of clustering consensus using a finite mixture of multinomial distributions in a space of clustering. The approach proposed in (Frossiniotis et al., 2002) is designed for combining runs of clustering algorithms with the same number of clusters. In (Strehl & Ghosh, 2002) the authors proposed combiners based on a hyper-graph model to solve the cluster fusion problem. The authors discuss two manners of consensus clustering: (1) Feature Distributed Clustering (FDC): a set of clustering are obtained from partial view of variables using all observations (2) Object-Distributed Clustering (ODC): with this technique the ensemble clustering has limited to subset of observation with access to all variables. The

authors provide three techniques (CSPA¹, HGPA², MCLA³), but indicate that HGPA delivers poor scores for the both data sets used in this chapter. Our work is in FDC category. In (Azimi et al., 2007) authors propose a modification of k-means for clustering a multiple runs of k-means. It's named intelligent k-means, which is especially defined for clustering ensembles. All these models assume that the correct number of clusters is given as parameter of model. In (Gionis et al., 2007) the authors give a formulation of ensemble clustering titled clustering aggregation, which does not require a number of clusters. The authors give a nice review of algorithm dedicated to ensemble clustering.

In this chapter, we offer a representation of consensus clustering as a set of new variables characterizing the observations. This leads to a formulation of the fusion problem as categorical clustering problem. We propose to use *Relational Analysis (RA)* as consensus method for unsupervised learning. The consensus clustering is provided as solution of the minimization of the objective function for a given consensus clustering. The main idea, shared with other algorithm is : If many clustering algorithms assign two observations in the same cluster, it will not benefit to consensus clustering to split these observations.

There are several advantages of RA consensus function: first we have low computational complexity, and second ability to deal with huge data set. Another purpose of our algorithm is not to neglect the weak clustering result. Often in the ensemble/aggregation/fusion clustering we combine only the best results. Given observations and m clustering result proposed with categorical variables, the purpose is to produce a single clustering that agree as much as possible with all results of clustering algorithms. The algorithm we propose for the problem of consensus clustering takes advantage of statistical formulation, (Benhadda & Marcotorchino, 2007).

Relational Analysis as clustering fusion can be applied in various settings. Multiple runs of clustering algorithm, like self-organizing map, generate a new variable space, which is significantly better than pure or normalized variable space. Therefore, running a simple clustering algorithm on generated variable space can provide the consensus cluster significantly better than pure observations. In this chapter we present another features of our framework:

- *Clustering categorical variable*: Consensus clustering provides a natural method for clustering categorical data.
- *Determining the correct number of clusters*: The formulation we propose don't require as parameter the number of clusters. The only parameter needed by RA is the similarity threshold.
- *Clustering mixed data* : the clustering fusion method can be particularly effective in the cases where data are defined over heterogeneous variables that contain incomparable values. We consider in this chapter a particular case, that we deal with continuous and categorical variables. In such cases the data set can be divided vertically into sets of homogeneous variables. Thus we apply an appropriate clustering algorithm and then combine the individual clustering into single clustering using categorical data clustering method.

¹ Cluster-based Similarity Partitioning Algorithm

² HyperGraph Partitioning Algorithm

³ Meta-Clustering Algorithm

The rest of the chapter is structured as follows: In section 2 we describe in detail the proposed model for consensus clustering. In section 3 we present a special case of global fusion based on self-organizing map. In section 4 we present experiments on public data set.

2. Relational analysis framework

Relational analysis theory is a mathematical data analysis approach with a broad application field. It was initiated and developed by (Marcotorchino & Michaud, 1978) at the IBM's European Center of Applied Mathematics (ECAM) by the end of the seventies. This technique uses the concept of "pairwise comparisons" which has been introduced in the statistical literature by the end of the thirties, through the work of (Kendall & Smith, 1940). Nevertheless the concept which has inspired the previous authors, dates of 1785 based upon some works of the "marquis de Condorcet" (Condorcet, 1785), related to "voting theory". In a general way, *Relational Analysis* makes it possible to model and solve problems whose general formulation can be stated as : *Seeking a particular relation \mathcal{R} which fits "as well as possible" single (or several) given relations $(\mathcal{R}^1, \mathcal{R}^2, \dots, \mathcal{R}^m)$* .

Unlike the existing clustering techniques, RA methodology does not need necessarily, neither to do sampling to be able to get results in a reasonable computing time, nor to fix arbitrarily the number of clusters that could be hidden in the data.

The principle of "pairwise comparisons" consists in transforming, each variable V measured on N objects into a $N \times N$ squared matrix C representing the similarity, with regards to variable V , between the N^2 couples of objects. An illustration of the "pairwise comparison principle" can be found in (Benhadda et al., 2007).

2.1 Relational analysis clustering methodology

To cluster a data set \mathcal{P} composed of n observations (O_1, O_2, \dots, O_n) described by m variables (V^1, V^2, \dots, V^m) , we firstly start by transforming each column V^k into a relational matrix C^k with general term $c_{ii'}^k$ defined by:

$$c_{ii'}^k = \begin{cases} 1 & \text{if } O_i \text{ and } O_{i'} \text{ have the same modality of variable } V^k \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

This term representing the similarity between the observations O_i and $O_{i'}$, with respect to variable V^k . Once all the m matrices C^k had been built up, we construct a global relational matrix C called "Condorcet's matrix" of general term $c_{ii'}$ representing the global similarity of O_i and $O_{i'}$ with respect to the whole set of the m variables: $c_{ii'} = \sum_{k=1}^m c_{ii'}^k$. This global similarity

has the so called "self maximal similarity property defined by: $c_{ii'} \leq M_{ii'} \forall O_i, O_{i'}$, where $M_{ii'} = \text{Min}(c_{ii}, c_{i'i'})$ is the "maximum possible similarity" between the two observations O_i and $O_{i'}$.

Using the global similarity $c_{ii'}$ and the "maximum possible similarity" $M_{ii'}$ between O_i and $O_{i'}$, we define their dissimilarity $\bar{c}_{ii'}$ as the complement of their global similarity to their "maximum possible similarity":

$$\bar{c}_{ii'} = M_{ii'} - c_{ii'} \quad (2)$$

Two observations will be, a priori, in the same cluster of the final expected partition as soon as their similarity will be greater than their dissimilarity i.e.: $c_{ii'} \geq \bar{c}_{ii'}$. The required final partition will be represented by a $N \times N$ binary squared matrix X with general term $x_{ii'}$ defined as follows:

$$x_{ii'} = \begin{cases} 1 & \text{if } O_i \text{ and } O_{i'} \text{ are in the same cluster} \\ & \text{of the final partition} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

This partition will be obtained by maximizing the Condorcet's criterion $\mathcal{C}(X)$ defined hereafter:

$$\mathcal{C}(X) = \sum_{i=1}^n \sum_{i'=1}^n (c_{ii'} x_{ii'} + \bar{c}_{ii'} \bar{x}_{ii'})$$

where:

$$\bar{x}_{ii'} = 1 - x_{ii'} \quad (4)$$

Using the expressions (2) and (4), the criterion $\mathcal{C}(X)$ can be rewritten as:

$$\mathcal{C}(X) = \sum_{i=1}^n \sum_{i'=1}^n (2c_{ii'} - \mathcal{M}_{ii'}) x_{ii'} + \sum_{i=1}^n \sum_{i'=1}^n \bar{c}_{ii'} \quad (5)$$

As the second member of the sum of expression (5) is a constant, we deduce that maximizing the Condorcet's criterion is equivalent to maximizing the following criterion $\mathcal{C}'(X)$

$$\mathcal{C}'(X) = \sum_{i=1}^n \sum_{i'=1}^n \left(c_{ii'} - \frac{\mathcal{M}_{ii'}}{2} \right) x_{ii'}$$

The cost function of the criterion $\mathcal{C}'(X)$ will be positive when the similarity $c_{ii'}$ between two observations O_i and $O_{i'}$ is greater or equal to half of their "possible maximal similarity". This condition is sometimes very difficult to reach, especially when the number of variables (or descriptors) is very high compared to the number of observations i.e. $m \gg n$, this is usually the case when the data set to be clustered is a set of documents. In that case, the number of clusters of the final partition will be so high that it could deprive the clustering task of its interest for practical purpose. As the goal of the clustering task is to summarize the amount of data into simpler structures, to avoid this problem, a solution consists in relaxing the cost function related to the clustering criterion. To reach that goal it is sufficient to replace the coefficient $1/2$ of $\mathcal{M}_{ii'}$ by a parameter α such that $0 < \alpha < 1/2$. The new formulation of the criterion $\mathcal{C}'(X)$ will be then:

$$\mathcal{C}'(X) = \sum_{i=1}^n \sum_{i'=1}^n (c_{ii'} - \alpha \times \mathcal{M}_{ii'}) x_{ii'}$$

Thus, the mathematical formulation of the relational analysis clustering problem is:

$$\max_X \mathcal{C}'(X)$$

under the constraints:

$$\left\{ \begin{array}{lll} x_{ii'} \in \{0, 1\} & \forall (O_i, O_{i'}) \in \mathcal{P}^2 & \text{(binarity)} \\ x_{ii} = 1 & \forall O_i \in \mathcal{P} & \text{(reflexivity)} \\ x_{ii'} - x_{i'i} = 0 & \forall (O_i, O_{i'}) \in \mathcal{P}^2 & \text{(symmetry)} \\ x_{ii'} + x_{i'i''} - x_{ii''} \leq 1 & \forall (O_i, O_{i'}, O_{i''}) \in \mathcal{P}^3 & \text{(transitivity)} \end{array} \right.$$

2.2 The RA heuristic

The exact solution of the problem above can be obtained by linear programming techniques when the studied population is relatively small (few hundreds). But, in practice, the data set size can often exceed hundreds of thousands or millions of observations. This situation leads to use heuristics, to get the "best" and closest partition to the exact one, in reasonable time processing. We give below the description of the heuristic which was used by the relational analysis methodology in the eighties.

Phase 1

This step consists in initializing the clustering process by building a first partition. To build up this first partition we construct progressively its clusters according the operations described bellow:

1. Initialization: we take randomly a first observation which constitutes the first cluster of the unknown partition
2. We take an observation $O_i \in \mathcal{P}$, and compute its link $\mathcal{L}_{i\mathcal{V}}$ (expression 6) with all the existing clusters \mathcal{V} .

$$\mathcal{L}_{i\mathcal{V}} = \sum_{i' \in \mathcal{V}} \mathcal{L}_{ii'} \quad (6)$$

where the link $\mathcal{L}_{ii'}$ between O_i and $O_{i'}$:

$$\mathcal{L}_{ii'} = c_{ii'} - \alpha \times \mathcal{M}_{ii'} \quad (7)$$

This observation is assigned to the cluster which has the biggest strictly positive link with. If all the links are negative, then we create a new cluster to put in this new observation.

3. **Repeat** this process until all observations of population \mathcal{P} had been assigned to a cluster.

Phase 2

At the end of the first step, we obtain a partition with a number of clusters⁴.

1. **Merging two clusters:** We take, now, the clusters one after another and we compute the link $\mathcal{L}_{\mathcal{V}\mathcal{V}'}$ (expression 8) of each cluster \mathcal{V} with all the others \mathcal{V}' .

$$\mathcal{L}_{\mathcal{V}\mathcal{V}'} = \mathcal{A}_{\mathcal{V}\mathcal{V}'} - \alpha \times \mathcal{M}_{\mathcal{V}\mathcal{V}'} \quad (8)$$

where the agreement $\mathcal{A}_{\mathcal{V}\mathcal{V}'}$ between the two clusters:

$$\mathcal{A}_{\mathcal{V}\mathcal{V}'} = \sum_{i \in \mathcal{V}} \sum_{i' \in \mathcal{V}'} c_{ii'}.$$

The disagreement $\bar{\mathcal{A}}_{\mathcal{V}\mathcal{V}'}$ between the two clusters is:

$$\bar{\mathcal{A}}_{\mathcal{V}\mathcal{V}'} = \sum_{i \in \mathcal{V}} \sum_{i' \in \mathcal{V}'} \bar{c}_{ii'},$$

and the possible maximal agreement $\mathcal{M}_{\mathcal{V}\mathcal{V}'}$ between the two clusters:

$$\mathcal{M}_{\mathcal{V}\mathcal{V}'} = \sum_{i \in \mathcal{V}} \sum_{i' \in \mathcal{V}'} \mathcal{M}_{ii'}.$$

⁴ This number is not fixed a priori, but will be discovered automatically during the first process

We will, then, merge the clusters, which have the best link (higher strict positive value). This must be carried out as long as there is a possibility to improve the criterion $\mathcal{C}'(X)$.

2. **Transferring an observation from a cluster to another one.** When no cluster's merging is possible, we take the observations of each cluster and compute the link $\mathcal{L}_{i\mathcal{V}}$ (expression 6) of each observation O_i with the other clusters \mathcal{V} . If an observation has a better link with another cluster than its own, then this observation is transferred from its own cluster to this new cluster. This will be carried out, as long as improvement of the criterion $\mathcal{C}'(X)$ occurs.

When, no observation's transfer is possible, we turn back to the merging step to see whether it is possible to improve the Condorcet's criterion by merging other clusters. These four steps will be applied, until no more improvements of the criterion occurred.

2.2.1 Illustrative example

Let us suppose that the studied population \mathcal{P} is composed of seven observations (O_1, O_2, \dots, O_7) which have three qualitative variables (V^1, V^2, V^3) were measured. The data set is presented in table 1.

	V^1	V^2	V^3
O_1	1	1	1
O_2	1	1	1
O_3	1	2	2
O_4	2	2	2
O_5	2	2	2
O_6	3	2	3
O_7	3	3	3

Table 1. Data set

After transformation of the three qualitative variables into their relational matrix representations, and after summing up those matrices, we obtain the Condorcet's global matrix C represented in table 2

	O_1	O_2	O_3	O_4	O_5	O_6	O_7
O_1	3	3	1	0	0	0	0
O_2	3	3	1	0	0	0	0
O_3	1	1	3	2	2	1	0
O_4	0	0	2	3	3	1	0
O_5	0	0	2	3	3	1	0
O_6	0	0	1	1	1	3	2
O_7	0	0	0	0	0	2	3

Table 2. Condorcet's global matrix C .

As the number of variables measured on this population is equal to three, it represents also the "maximum possible similarity" that can occur between two observations O_i and $O_{i'}$. We

can then deduce, that the global dissimilarity between those observations is $\bar{c}_{ii'} = 3 - c_{ii'}$. The binary squared matrix X , representing the obtained final partition of population \mathcal{P} has the following general term:

$$x_{ii'} = \begin{cases} 1 & \text{if } c_{ii'} \geq \bar{c}_{ii'} \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

Due to the transitivity constraints, the solution is not so trivial⁵ because of the so called "Condorcet's effect" cf. (Marcotorchino & Michaud, 1978; 1982), but the proposed heuristic is able to take into account some of those constraints limitations and avoid getting untransitive solutions. Applying the heuristic to the example (see Table 3), one gets the following optimal solution:

- Cluster 1: O_1, O_2
- Cluster 2: O_3, O_4, O_5
- Cluster 3: O_6, O_7

The relational representation X of this partition is then:

	O_1	O_2	O_3	O_4	O_5	O_6	O_7
O_1	1	1	0	0	0	0	0
O_2	1	1	0	0	0	0	0
O_3	0	0	1	1	1	0	0
O_4	0	0	1	1	1	0	0
O_5	0	0	1	1	1	0	0
O_6	0	0	0	0	0	1	1
O_7	0	0	0	0	0	1	1

Table 3. Binary matrix representation X of the final partition .

The corresponding Condorcet's criterion value is: $\mathcal{C}(X) = 131$.

3. Special case of clustering mixed data: Global Fusion

A specific SOM (Self-Organizing Map) model has been developed for mixed data using the similar cost function as the model presented in Kohonen (2001); Lebbah et al. (2005). The model dedicated to binary and continuous data is called MTM (Mixed Topological Map). As with a traditional self-organizing map, we assume that the lattice \mathcal{C} (map) has a discrete topology defined by an indirect graph. Usually, this graph is a regular grid in one or two dimensions. For each pair of cells (c,r) on the map, the distance $\delta(c,r)$ is defined as the length of the shortest chain linking cells r and c . Let $\mathcal{P} = \{O_i, i = 1..n\}$ the learning data set where each observation $O_i = (O_i^1, O_i^2, \dots, O_i^k, \dots, O_i^m)$ is made of two parts: continuous part $O_i^{r[.]}$ = $(O_i^{r[1]}, O_i^{r[2]}, \dots, O_i^{r[d_r]})$ ($O_i^{r[.]} \in \mathcal{R}^{d_r}$) and binary part $O_i^{b[.]}$ = $(O_i^{b[1]}, O_i^{b[2]}, \dots, O_i^{b[k]}, \dots, O_i^{b[d_b]})$ where the k th component $O_i^{b[k]}$ is binary variable ($O_i^{b[k]} \in \beta = \{0,1\}$) such as each observation O_i is thus, a realization of a random variable which belongs to $\mathcal{R}^{d_r} \times \beta^{d_b}$. With these notations a particular observation $O_i = (O_i^{r[.]}, O_i^{b[.]})$ is

⁵ Just applying the rule (9) could yield to untransitive solution.

a mixed of subvectors (continuous and binary variables) of dimension $m = d_r + d_b$.

Since for binary vectors the Euclidean distance is no more than the Hamming distance \mathcal{H} , then the Euclidean distance can be rewritten by:

$$\|O - \mathbf{w}_c\|^2 = \|O^{r[\cdot]} - \mathbf{w}_c^{r[\cdot]}\|^2 + \mathcal{H}(O^{b[\cdot]}, \mathbf{w}_c^{b[\cdot]})$$

where $\mathcal{H}(O^{b[\cdot]}, \mathbf{w}_c^{b[\cdot]})$ the complement of global similarity between a binary part of an observation O and referent $\mathbf{w}_c^{b[\cdot]}$.

Using this expression, the cost function of the traditional SOM algorithm, which is dedicated to mixed data can be expressed as:

$$\begin{aligned} \mathcal{G}(\phi, \mathcal{W}) &= \sum_{O_i \in \mathcal{P}} \sum_{r \in \mathcal{C}} \mathcal{K}(\delta(r, \phi(O_i))) \|O_i^{r[\cdot]} - \mathbf{w}_r^{r[\cdot]}\|^2 \\ &\quad + \sum_{O_i \in \mathcal{P}} \sum_{r \in \mathcal{C}} \mathcal{K}(\delta(r, \phi(O_i))) \mathcal{H}(O_i^{b[\cdot]}, \mathbf{w}_r^{b[\cdot]}) \end{aligned} \quad (10)$$

Where ϕ assigns each observation O_i to a single cell in \mathcal{C} . \mathcal{K} is a particular kernel function which is positive and symmetric ($\lim_{|y| \rightarrow \infty} \mathcal{K}(y) = 0$).

The first term is the classical cost function used by the Kohonen Batch algorithm Kohonen (2001), and the second term is the cost function used in BinBatch model Lebbah et al. (2000). The cost function (10), is minimized using an iterative process with two steps.

1. Assignment step, which leads to the use of the following assignment function:

$$\forall O, \phi(O) = \arg \min_c \left(\|O^{r[\cdot]} - \mathbf{w}_c^{r[\cdot]}\|^2 + \mathcal{H}(O^{b[\cdot]}, \mathbf{w}_c^{b[\cdot]}) \right)$$

2. Optimization step: It is easy to see that this two minimizations of both terms allow to define:

- The continuous part $\mathbf{w}_c^{r[\cdot]}$ of the referent vector \mathbf{w}_c as the mean vector as:

$$\mathbf{w}_c^{r[\cdot]} = \frac{\sum_{O_i \in \mathcal{P}} \mathcal{K}(\delta(c, \phi(O_i))) O_i^{r[\cdot]}}{\sum_{O_i \in \mathcal{P}} \mathcal{K}(\delta(c, \phi(O_i)))},$$

- The binary part $\mathbf{w}_c^{b[\cdot]}$ of the referent vector \mathbf{w}_c as the median center of the binary part of the observations $O_i^{b[\cdot]} \in \mathcal{P}$ weighted by $\mathcal{K}(\delta(c, \phi(O_i)))$. Each component $\mathbf{w}_c^{b[\cdot]} = (w_c^{b[1]}, \dots, w_c^{b[k]}, \dots, w_c^{b[d_b]})$ is then computed as follows:

$$w_c^{b[k]} = \begin{cases} 0 & \text{if } \left[\sum_{O_i \in \mathcal{P}} \mathcal{K}(\delta(c, \phi(O_i))) (1 - O_i^{b[k]}) \right] \geq \\ & \quad \left[\sum_{O_i \in \mathcal{P}} \mathcal{K}(\delta(c, \phi(O_i))) O_i^{b[k]} \right] \\ 1 & \text{otherwise} \end{cases},$$

4. Experimental evaluation

In the following, the RA is used as the clustering consensus/fusion based algorithm for categorical and mixed data. First, the original data set is divided into two sub-data sets: pure categorical data set and pure continuous data set. Next, existing well established clustering algorithms designed for different data types are employed to provide corresponding clusters. We can run many algorithms or the same with different parameter using the same data. Finally the clustering results are combined as categorical data set to provide a consensus single clustering.

As quality evaluation criterion we use purity index. However, when class labels are available for each observation, we can use purity measure to indicate the match between cluster labels and class labels. The purity assess clustering quality from 0 (worst) to 1 (best).

5. Relational analysis for clustering categorical dataset

We used our RA clustering technique to cluster textual database "20 Newsgroups", which is a reference, for benchmarks for the data analysis scientific and technical community. This database is composed of 19997 documents, stemming from 20 different forums and described by 145980 descriptors (or variables). A major characteristic of this database is its heterogeneity both in terms of size of the documents and in terms of their themes and styles citelemoine.

At the end of the clustering process, we obtain 330 clusters. These clusters were sorted out in decreasing of magnitude (their size) order. As an example, we give here the list of the 7 first biggest clusters. Each cluster is described by the words or expressions (descriptors) participating the most into its constitution

Cluster	Descriptors	Cardinal
1	game, team, player, hockey, season, playoff, fan, baseball, league, coach	1325
2	file, directory, program, window, FTP, archive, DOS, disk, server	1144
3	Government, right, law, constitution, weapon, citizen, president, gun, policy	1095
4	Car, engine, mile, tire, mileage, brake, dealer, wheel, auto, clutch	755
5	Clipper, encryption, key, chip, escrow, crypto, wire tap, algorithm, privacy, government	673
6	Drive, SCSI, IDE, disk, controller, ram, floppy, CD-ROM, jumper, software	628
7	Card, video, driver, ISA, monitor, bus, VGA, VLB, SVGA, graphics	579

Table 4. The first seven clusters of the final partition.

Interpretation attempt

We can observe, in view of the descriptors characterizing the clusters that:

- the cluster 1 is compound of documents which generally deal, with "sport",
- the cluster 2 is compound of documents which are mainly related to "software" in general,
- the cluster 3 is built up with documents which are concerned with "politics"("policy"),
- the cluster 4 gathers documents which deal, in general, with "motorcar",
- the cluster 5 is made up of documents dedicated to "encoding and data protection",
- the cluster 6 is compound of documents which deal, generally, with "computer hardware" and more particularly with the choice between IDE or SCSI, and finally,
- the cluster 7 gathers documents which are also concerned with "computer hardware" and more particularly with video material.

5.1 Artificial data sets for fusion

We illustrate the cluster consensus applications on two artificial data sets downloaded from <http://strehl.com/> and used by (Strehl & Ghosh, 2002). The first data set (2D2K) was artificially generated and contains 500 observations each of two 2-dimensional (2D) Gaussian clusters. The second data set (8D5K) contains 1000 observations from multivariate Gaussian distributions (200 observations each) in 8D space.

For this experiment we take several clustering results provided by Strehl in his website <http://strehl.com/>. The authors provide two simulations of clustering ensemble: (FDC, Exp1) Feature-distributed Clustering (ODC, Exp2): Object-distributed Clustering. Table 5 indicates different results provided by Strehl and Ghosh adding the result obtained with our consensus clustering technique RA in both experimentations. Our purpose through this comparison, is not to assert that our method is the best, but to show that RA method can obtain quite the same good results as the two previous ones, without making any arbitrary assumptions about the number of clusters to be found. Indeed, as shown in the table below, we can see that RA method give similar results and quite comparable to the ones obtained by both proposed techniques (FDC, ODC). The main difference between these three methods is that, unlike the two other methods, RA does not require a priori knowledge of the number of clusters.

	8D5K	RA
FDC (Exp1)	0.9970	0.9930
ODC (Exp2)	0.9480	0.9330

	2D2K	RA
FDC (Exp1)	0.9440	0.9440
ODC(Exp2)	0.9680	0.9700

Table 5. Comparison of consensus clustering. FDC: Feature-Distributed Clustering; ODC: Object-Distributed Clustering; RA: Relational Analysis; Exp: Experimentation

5.2 Real data sets and fusion

We will use three data sets coming from UCI repository (Asuncion & Newman, 2007). These data are mixed, in the sense that they contain both numerical and categorical data. These data are described below.

Heart disease data set: this data set, which is D. Detrano's heart disease data set, was generated by the Cleveland Clinic. It consists in 303 observations, described by 6 numerical and 8 categorical variables. The observations are also classified into two classes: healthy class (buff) and with heart-disease class (sick).

Credit data set : The data set has 690 instances, each being described by 6 continuous and 9 categorical variables. The observations were classified into two classes, approved class and rejected class.

Handwritten data: this data set consists of the handwritten numerals ("0"–"9") extracted from a collection of Dutch utility maps. There are 200 samples of each digit such that there is a total of 2000 samples. Each sample is a 15×16 binary pixel image. The data set is represented as a 2000×240 binary data matrix. Each categorical variable is a pixel with two possible values "On=1" and "Off=0".

In the first experiment we simulate such clustering result by running several clustering algorithms, each one having access to only a restricted categorical or continuous variables. Thus, each cluster has a partial view of the observations. The clusters are found using subspaces and adapted clustering technique. In the consensus clustering, cluster labels are clustered using RA technique. In order to compare our result, we cluster the data using a dedicated Self-organizing map for mixed categorical and continuous data. This technique is titled Mixed Topological Map (MTM), and provide a small cluster organized as map (see section 3). Often we use hierarchical clustering to reduce the number of the clusters (Vesanto & Alhoniemi, 2000). The combining method is indicated by MTM+HC and the number of clusters between bracket.

The figures 1 and 2 show the comparative results in term of number of clusters and the purity index. As can be seen, the both figures indicate that RA provides the high scores when compared for the same number of clusters. Note in this case for the both data set we have, a priori, two classes, and the RA (2) provides high purity for this case. We note also that RA don't require two steps of clustering, comparing to the MTM and other clustering ensemble algorithms found in the literature which needs an agglomerative clustering technique to reduce the number of clusters. The only parameter needed by RA is the similarity threshold.

In this second experiment we use Handwritten data set. The purpose is to use RA as consensus clustering of several runs of the same clustering algorithm. In this case we simulate 16 cluster results obtained with Self-organizing map dedicated to categorical data and hierarchical clustering, using different parameters (Lebbah et al., 2005; Vesanto & Alhoniemi, 2000). We use 5 cluster results with purity score lower than 0.4, and four results lower than 0.72, and the rest results are between 0.74 and 0.76. Thus the RA consensus clustering provide a stable purity with 0.76.

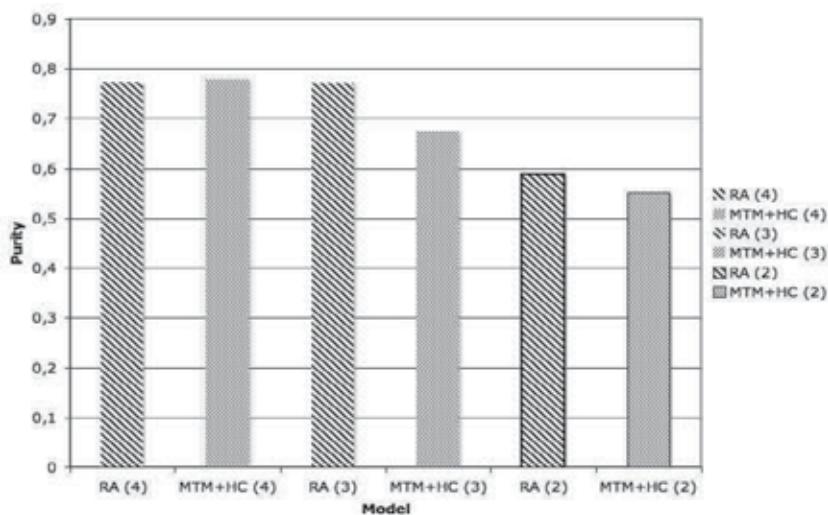


Fig. 1. Credit data set. Purity scores for consensus clustering. RA : Relational Analysis; MTM: Mixed Topological Map. HC: Hierarchical Clustering. The number between brackets indicates the number of clusters provided automatically

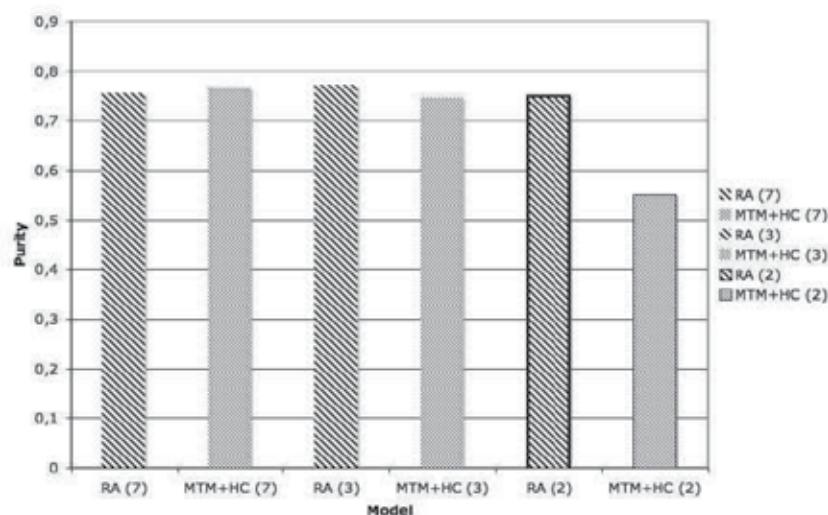


Fig. 2. Heart disease data set. Purity scores for consensus clustering. RA : Relational Analysis; MTM: Mixed Topological Map. HC: Hierarchical Clustering. The number between brackets indicates the number of clusters provided automatically

The figure 3 shows the distribution of each class of digit in all 15 consensus clusters. The figure 4 shows the best map obtained among the 16 maps used for consensus clustering. We visualize this figure in order to interpret the results of consensus. We note that RA grouped in a cluster numbered 12, 13, 15, the mix of digit 7, 9 and digit 5. It is clear to see on the map (Fig.4) that some figures such as "9" are written in the same way as "5" and "7". The same analysis could be done with the other clusters.

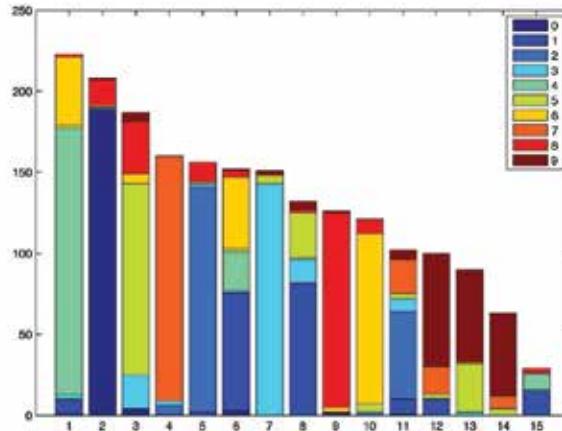


Fig. 3. Consensus clustering with RA. Each bar shows the distribution of each cluster.

6. Conclusions

In this chapter, we formally defined the problem of clustering and we presented an original and new approach of fusion/ensemble/consensus/aggregation clustering. The main idea was to find a clustering (or partition) of observations that represents the best consensus between several other clustering related to the same data set. The goal of the proposed algorithm is the improvement of confidence in cluster assignments by evaluating a history of cluster assignments for each observation. If we compare our algorithm (or method) to some recent clustering algorithm, we can assert that, unlike these new algorithms, our method is scalable, linear, in memory use and computational time and can handle data represented as observations cross attributes or as similarity matrix. Our clustering method handles missing values without replacing them by values that could be very far away from the true ones. It also contains a preprocessing module that, among other processings, can compute how discriminant are the attributes measured on the observations to be clustered. Finally we verified the intuitive appeal of the proposed approach and we studied the behavior of our algorithm on real and synthetic heterogeneous data sets. We observed that the proposed method increases performance as more as iterations of the process are performed. Another advantage of our method is that, neither do we need to re-process the data; nor do we need to fix the same cluster numbers for each application or clustering algorithm. In the future, we would like to perform a more detailed analysis involving huger data set and investigating the collaborative clustering.

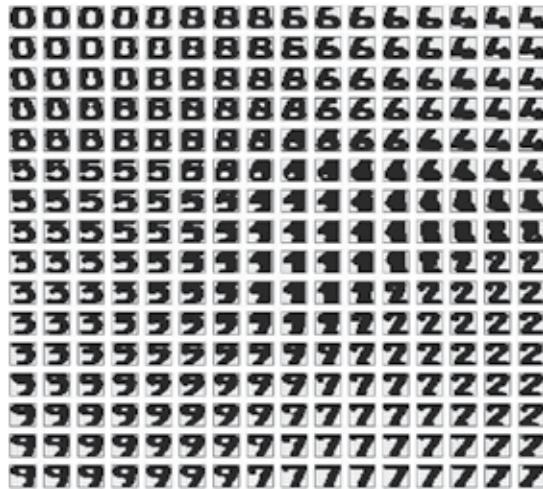


Fig. 4. 16×16 map using MTM with only categorical data

7. References

- Asuncion, A. & Newman, D. (2007). UCI machine learning repository, <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
- Azimi, J., Abdoos, M. & Analoui, M. (2007). A new efficient approach in clustering ensembles, *IDEAL, International Conference on Intelligent Data Engineering and Automated Learning*.
- Benhadda, H. & Marcotorchino, F. (2007). L'analyse relationnelle pour la fouille de grandes bases de données., *Revue des Nouvelles Technologies de l'Information*, RNTI-A-2, Cépaduès, pp. 149–167.
- Benhadda, H., Patino, J., Corvee, E., Bremond, F. & Thonnat, M. (2007). Data mining on large video recordings, *Colloque V.S.S.T.2007 : Veille Stratégique Scientifique & Technologique (21-25 Octobre) Marrakech*.
- Condorcet, M. N. D. (1785). Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix, *De l'imprimerie royale, Paris*.
- Dudoit, S. & Fridlyand, J. (2003). Bagging to improve the accuracy of a clustering procedure, *Bioinformatics* **19**.
- Frossyniotis, D. S., Pertsakis, M. & Stafylopatis, A. (2002). A multi-clustering fusion algorithm, *SETN '02: Proceedings of the Second Hellenic Conference on AI*, Springer-Verlag, London, UK, pp. 225–236.
- Gionis, A., Mannila, H. & Tsaparas, P. (2007). Clustering aggregation, *ACM Trans. Knowl. Discov. Data* **1**(1): 4.
- Kendall, M. G. & Smith, B. B. (1940). On the method of paired comparisons, *Biometrika* **31**: 324–345.
- Kim, S. Y. & Lee, W. (2007). Ensemble clustering method based on the resampling similarity measure for gene expression, *Stat Methods Med Res* **16**: 539–564.
- Kohonen, T. (2001). *Self-organizing Maps*, Springer Berlin.
- Lebbah, M., Chazottes, A., Badran, F. & Thiria, S. (2005). Mixed topological map., *ESANN*, pp. 357–362.

- Lebbah, M., Thiria, S. & Badran, F. (2000). Topological map for binary data, *Proceedings European Symposium on Artificial Neural Networks-ESANN 2000, Bruges, April 26-27-28*, pp. 267–272.
- Marcotorchino, F. & Michaud, P. (1978). Optimisation en analyse ordinale des données, *Biometrika* **65**: 324–345.
- Marcotorchino, F. & Michaud, P. (1982). Agrégation des similarités en classification automatique, *Revue de statistique appliquée* **30**(2): 21–44.
- Minaei-Bidgoli, B., Topchy, A. & Punch, W. F. (2004). Ensembles of partitions via data resampling, *ITCC '04: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 2*, IEEE Computer Society, Washington, DC, USA, p. 188.
- Monti, S., Tamayo, P., Mesirov, J. & Golub, T. (2003). Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data, *Mach. Learn.* **52**(1-2): 91–118.
- Strehl, A. & Ghosh, J. (2002). Cluster ensembles – a knowledge reuse framework for combining multiple partitions, *Journal on Machine Learning Research (JMLR)* **3**: 583–617.
- Topchy, A. P., Jain, A. K. & Punch, W. F. (2004). A mixture model for clustering ensembles, *SDM,proceedings of the Fourth SIAM International Conference on Data Mining, Lake Buena Vista, Florida, USA, April 22-24*.
- Topchy, M.-A., Jain, F.-A. K. & Punch, W. (2005). Clustering ensembles: Models of consensus and weak partitions, *IEEE Trans. Pattern Anal. Mach. Intell.* **27**(12): 1866–1881.
- Vesanto, J. & Altoniemi, E. (2000). Clustering of the self-organizing map, *Neural Networks, IEEE Transactions on* **11**(3): 586–600.

A Classifier Fusion System with Verification Module for Improving Recognition Reliability

Ping Zhang

*Department of Mathematics and Computer Science
Department of Advanced Technologies
Alcorn State University
USA*

1. Introduction

Recognition reliability is a vital and sensitive issue in the pattern recognition applications. Recognition with a proper rejection option provides a means to reduce the error rate and to increase recognition reliability (Chow, 1970). It deals with how research projects can be developed into real applications. Relatively few research papers on this topic have been reported in literature (Frelicot & Mascarilla, 2002). In order to clearly explain the recognition reliability problem, the Recognition Rate (RR), Mis-Recognition Rate (MR), Rejection Rate (RJ), and Reliability (Re) are defined and their relationships are analyzed as follows:

The Recognition Rate (RR) is defined as:

$$RR = \frac{\text{Number of correctly recognized objects}}{\text{Total number of testing objects}} \quad (1)$$

The Misrecognition Rate (MR) is presented as:

$$MR = \frac{\text{Number of misrecognized objects}}{\text{Total number of testing objects}} \quad (2)$$

The Rejection Rate (RJ) is written as:

$$RJ = \frac{\text{Number of rejected objects}}{\text{Total number of testing objects}} \quad (3)$$

The Reliability (RE) can be denoted as:

$$RE = \frac{\text{Total number of testing objects} - \text{Number of misrecognized objects}}{\text{Total number of testing objects}} \quad (4)$$

The reliability can also be deduced as:

$$RE = 1 - MR = RJ + RR \quad (5)$$

Technically speaking, there is a tradeoff for a recognition system to pursue a very high recognition rate and a very low misrecognition rate at the same time given a specific set of feature set and a classifier or a combination of classifiers (Chow, 1970). It is common sense that setting a relatively low threshold for a recognition system can achieve a high recognition rate; however, it also introduces more misrecognitions.

In many pattern recognition systems, the goal is to seek the highest reliability and the highest recognition rate as possible at the same time. In other words, the misrecognition rate must be suppressed. When designing a sensitive object recognition system, it is preferred rejecting objects with low confidences over mistakenly recognizing the objects (Zimmermann, Bertolami & Bunke, 2002). For example, in an automatic bank check processing system, a very high recognition reliability is a vital criterion. The misrecognition is absolutely forbidden and a small percentage of rejection is allowed. The rejected checks can be sent for manually handling.

An automatic bank check processing system can be divided into the following aspects: 1) magnetic ink character recognition (MICR); 2) handwritten legal amount recognition (English character recognition, or other language character recognition); 3) handwritten courtesy amount recognition (handwritten digital recognition); 4) payer's signature verification or recognition; 5) the recognition of name and address of a payer, etc.

Among the above mentioned recognitions, MICR plays an important role in the automatic bank check processing system based on the following reasons:

- a) Information in the MICR area includes the account number of a payer and bank identification number; both of which need to be firstly recognized in order to verify the payer's identification and the payer's bank number while transition is processed;
- b) Individual character recognition rate in the MICR area is very high (over 99%), it is possible to use an automatic process.

However, some errors have been reported in the bank applications due to the following reasons: the mechanical deficiency of MICR scanners; the distortion of the printed characters in the MICR area, and others.

In this paper, we will propose a novel classifier fusion system with a verification module to improve system's reliability. The arrangement of the paper is as follows: In Section 2, the concept of MICR is briefly introduced, which includes one dimensional MICR waveform process and two dimensional image process. The flowchart of the new classifier fusion system with a verification module is proposed in the section 3; In Section 4, the basic concepts of classifiers: Artificial Neural Networks (ANNs), modified K-Nearest Neighbor (KNN) classifier and Support Vector Machines (SVMs) are reviewed. A gating network for congregating the outputs of ANN and KNN is applied to the classifier fusion system. Three

experiments conducted on the MICR character recognition are reported, and the recognition reliability is analyzed in the section 5. Finally, the conclusion ends this paper.

2. MICR Information Processing

Fig. 1 shows a blank bank check image. The MICR area is located at the bottom of the check. In North America, E13B font has been officially used as the printed fonts in the MICR area for almost commercial banks. E13B symbols include ten numerals and four symbols: On-Us symbol, Transit symbol, Amount symbol, and Dash symbol.

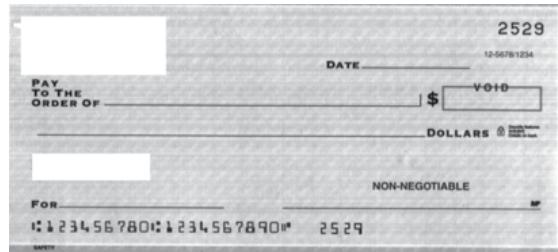


Fig. 1. MICR area at the bottom of check

MICR characters are printed with a magnetic ink or toner. A specially designed MICR scanner can read not only the E13B character waveforms (one dimensional signals), but also the characters' images with different resolutions (two dimensional images). The standard font images and their waveforms of the fourteen characters are shown in Fig. 2.

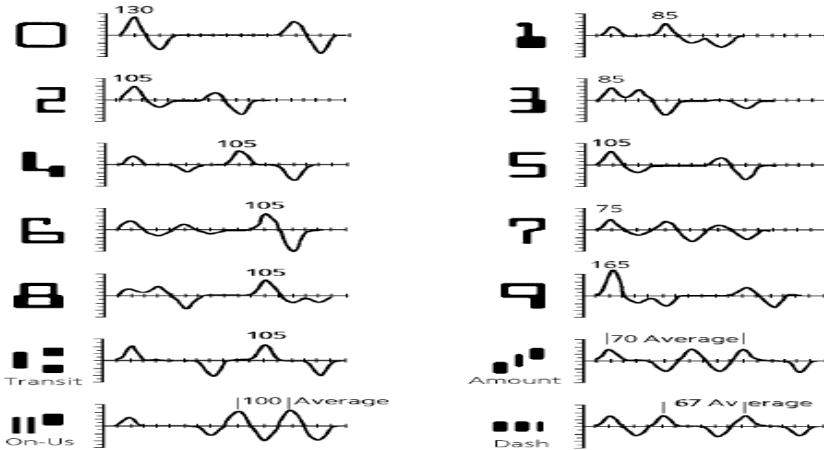


Fig. 2. E13B MICR fonts and their waveforms

2.1 One Dimensional MICR Waveform

One dimensional MICR waveform processing and recognition has been extensively researched for a long time. The very high recognition rate was reported under an ideal condition. However, there are still some recognition errors and rejections reported in the commercial applications due to unstable paper feeding mechanism of scanners, the

distortion of printed MICR characters on the checks and other factors, which lead to waveform distortions and noise on MICR images. Fig. 3 shows two MICR waveforms scanned from two personal bank checks: one has two sub-MICR areas; another has four sub-MICR areas.

A waveform segmentation algorithm can segment each sub-MICR waveform into individual waveforms, each representing one character. The detail algorithm for waveform segmentation is beyond the scope of this paper.

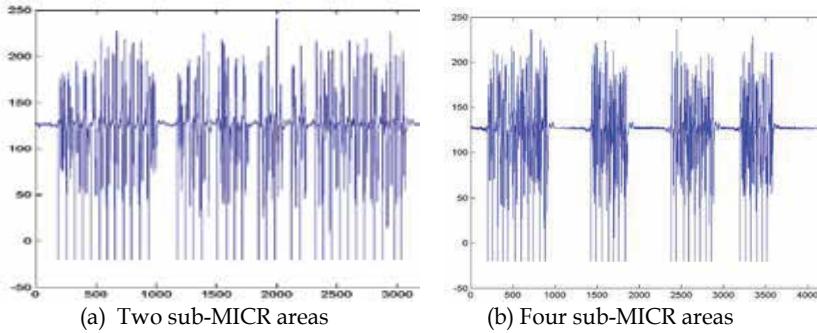


Fig. 3. MICR waveform

2.2 Image-based Character Segmentation

For the recognition of image-based MICR characters, the key issue is how to deal with image segmentation, noise removal, image enhancement, feature extraction, and the design of classifiers for recognition. The image-based character image segmentation in the MICR area can be divided into two steps: 1) locating top and bottom lines of the characters; 2) segmenting each character from the vertical direction.

In order to accurately locate the top and the bottom lines of the MICR characters in the check images, a fast algorithm is proposed as follows: scanning each line in the horizontal direction, counting the number of zero-crossing points in each horizontal line as NC ; If the number of characters in the MICR area is N , then following conditions apply to locate the top or the bottom lines of MICR characters:

Condition 1: If $NC \geq 2*N$, then the line likely belongs to MICR character area.

Condition 2: Beginning at the L^{th} line of the MICR image, then moving downwards, if there are a few consecutive lines satisfy condition 1, then the L^{th} line is the top line of MICR character area. The same method is applied to locate the bottom line of MICR character area.

Condition 3: if a check is scanned with certain angle w , the line seeking algorithm presented in Condition 2 is also traced with this angle.

As soon as the top line and the bottom line of the MICR character area are located, the characters are segmented based on character's connectivity and vertical profiles. As to four special symbols, each symbol consists of three black blocks. The following criteria are used to combine three black blocks as a symbol:

- a) The distance between two adjacent black blocks is shorter than the distance between two characters;
- b) The length and the width of any black block of the four symbols are shorter than that of the ten numerals.

Fig. 4 (a) shows an MICR character image; Fig. 4 (b) shows the segmentation result of Fig. 4 (a). MICR area image with background is shown in Fig. 5 (a) and the segmented image is shown in Fig. 5 (b)

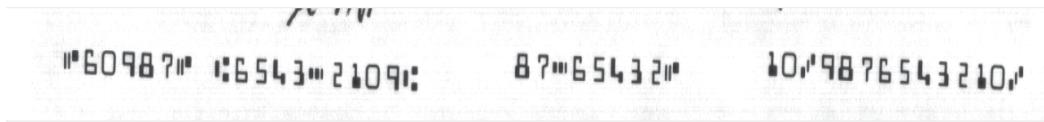


Fig. 4 (a) Original MICR character image



Fig. 4. (b) Segmentation result of Fig. 4 (a)

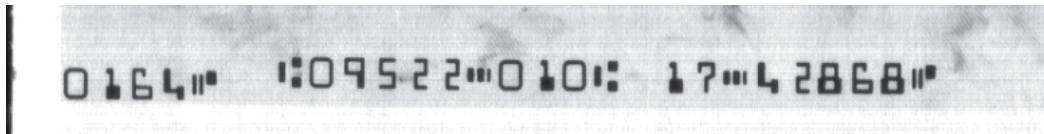


Fig. 5. (a) MICR character image with background



Fig. 5. (b) Segmented image of Fig. 5 (a)

3. Classifier Fusion System with SVM Verification Module

It is difficult for a single classifier to obtain a very high reliability and recognition rate at the same time for a complex pattern recognition system. Some theoretical advancements have been proposed in literature (Kuncheva, 2002; Kittler, Hatef, Duin & Matas, 1998). There are a few possible solutions to help reduce the number of errors. One solution is to employ a verification module. Another solution is to use a combination of multiple classifiers (Suen & Tan, 2005). The different features extracted by different means, which are inputted to different ensemble classifiers for classification, have different merits for recognition because some of the features are complementary (Zhang, Bui. & Suen, 2007). It is reasonable to combine two classifiers to produce a higher reliability and at the same time to seek the lowest misrecognition rate. A classifier fusion system with SVM verification module is proposed and it is shown in Fig. 6.

In the proposed recognition and verification scheme, a classifier fusion system consists of an ANN classifier and a KNN classifier, which are trained by two sets of feature vectors respectively. As the two sets of feature vectors may be complementary, the trained ANN and KNN as recognizers have their merits on character recognition. Experiments will prove that the combination of two classifiers can achieve a higher recognition rate.

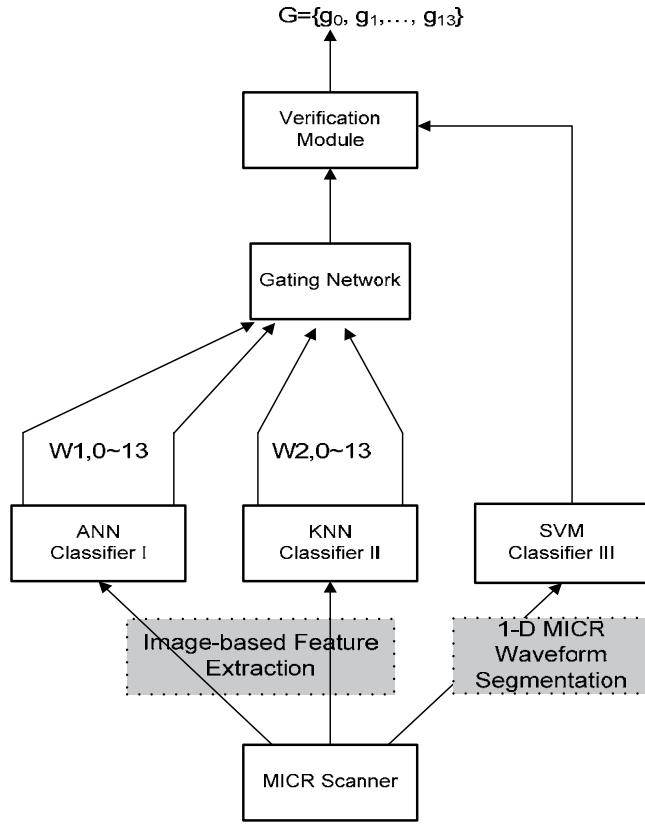


Fig. 6. Classifier Fusion system with SVM verification module

There are fourteen characters in the E13B character set. For the verification purpose, fourteen two-class SVMs are applied to classify the MICR waveforms. the result of SVMs is used to verify the image-based recognition result. The detail recognition and verification process will be elaborated in Section 5.

4. Classifier Design and Feature Extraction

4.1 Artificial Neural Network Classifier

An ANN is an interconnected group of artificial neurons (Duda, Hart & Stork, 2000). ANN refers to electrical, mechanical, or computational simulations or models of biological neural networks. One of the most popular methods for training a multilayer network is based on the gradient descent principle using the back-propagation algorithm or generalized delta rule. The principle is a natural extension of the Least Mean Squares (LMS) algorithm because it is powerful, useful, and relatively easy to understand and implement.

An ANN classifier consists of input units, hidden units, and output units. In terms of classifying fourteen numerals and symbols in this research, the ANN will have fourteen output units. The signal from each output unit is the discriminant function $g_k(x)$. The discriminant function can be expressed as:

$$g_k(x) \equiv z_k = f\left(\sum_{j=1}^r w_{kj} f\left(\sum_{i=1}^d w_{ji} x_i + w_{j0}\right) + w_{k0}\right) \quad (6)$$

where x_i is a feature component; w_{ji} is a weight between the input layer and the hidden layer; w_{kj} is a weight between the hidden layer and the output layer; $i=1, \dots, d$, and d is the number of nodes in the input node; $j=1, \dots, r$, and r is the number of nodes in the hidden layer; $k=0, 1, 2, \dots, m$, which represents the number of nodes in the outputs layer. For example, fourteen nodes of outputs represent ten digits and four special symbols used in this paper. Thus, the discriminant function can be implemented by a three-layer neural network. The configuration of the three-layer neural network for the recognition is drawn in Fig. 7.

We now turn to the crucial problem of setting the weights based on training patterns and the desired output. Backpropagation algorithm is used to train the classifier. Some of considerations in the training and testing procedures are listed as follows:

Target Values: The target value (the desired output) of the output category is chosen as +1, while others are set equal to 0.0.

Number of Nodes in the ANN: According to a convenient rule of thumb, the total number of weights in the net is roughly chosen as $n/10 \sim n/4$. Here n is the number of training samples.

Initializing Weights: Random data are generated for all weights in the range of $-1.0 < \text{all weights} < +1.0$.

Learning Rates: In general, the learning rate is small enough to ensure convergence. A learning rate of (0.1-0.4) is often adequate as a first choice.

Training Different Patterns: We used the following strategies to train the classifiers: our training procedure concentrates on the “difficult” patterns. Firstly, an ANN classifier is trained on all training samples, then the same set of training samples are fed into the ANN for testing. Those “difficult” patterns, which are not correctly recognized, are copied several times and randomly put into the training set for training again. As more “difficult” patterns are in the training set, the ANN can adaptively learn how to correctly recognize those “difficult” patterns without losing its generality.

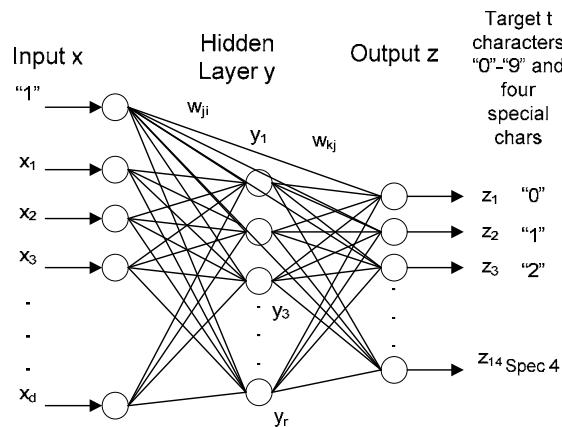


Fig. 7. Configuration of three-layer neural networks

4.2 KNN Classifier

In a KNN classifier, for each testing sample, the Euclidean distance between the testing sample and all the training samples are calculated. Let the testing sample x_i be represented by the feature vector $[x_1^i, x_2^i, x_3^i, \dots, x_N^i]^T$, where x_k^i denotes the value of the k th feature component in the i th sample. The distance between x_i and x_j can be calculated by

$$d(x_i - x_j) = \sqrt{\sum_{k=1}^N (x_k^i - x_k^j)^2} \quad (7)$$

If the number of training data is N , then N distances will be identified as neighbors. If $K=1$, then the class label of the testing sample is equal to the closest training data. If $K>1$, then the class label of the testing sample is equal to the class label that most of the neighbors belong. The output of the KNN algorithm can be interpreted as a posteriori probability. Hence, instead of labeling the output class label equal to the class label that most of the neighbors have, we assign the following class confidence values of x :

$$p_c(x) = (\text{no. of neighbors with class label } c)/K \quad (8)$$

Here, p_c is the a posteriori probability that x belongs to the class c ; K denotes the number of nearest neighbors. We can assign class label j to the testing sample x when

$$p_j(x) = \max_{1,2,\dots,M} \{p_c(x)\} \quad (9)$$

Here M is the total number of classes.

One improvement to the KNN algorithm is to weigh the contribution of each of the K neighbors based on its distance to the testing sample. The closest neighbor should receive the highest weight. It can be represented by modifying the equation into following:

$$p_c(x) = \sum_{i=1}^K \left(\frac{1}{\sum_{j=1}^K \frac{1}{d(x, x_j)^2}} \right) \quad (10)$$

The equation can be normalized as:

$$\sum_{c=1}^M p_c(x) = 1$$

The KNN algorithm with this refinement is also known as the fuzzy K-nearest neighbor algorithm (Keller, Gray and Givens, 1985). The normalized $p_c(x)$ can be used as input of gating network indicated in Fig. 6.

4.3 SVM Classifier

SVMs rely on the preprocessing the data to represent patterns in a higher dimension by an appropriate nonlinear mapping $\phi(\cdot)$. Data from two categories can always be separated by a

hyperplane. The detail theory can be referred to the references (Decoste & Scholkopf, 2002; Heisele, Serre, Prentice & Poggio, 2003).

In this research, the kernel function is a Gaussian radial basis kernel:

$$K(x, z) = \exp(-\|x - z\|^2 / \sigma^2) \quad (11)$$

Training a support vector machine for the pattern recognition problem leads to the following quadratic optimization problem:

$$W(\alpha) = -\sum_{i=1}^l \alpha_i + \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l y_i y_j \alpha_i \alpha_j k(x_i, x_j) \quad (12)$$

and subject to:

$$\begin{aligned} \sum_{i=1}^l y_i \alpha_i &= 0 \\ \forall i : 0 \leq \alpha_i &\leq C \end{aligned} \quad (13)$$

The number of training examples is denoted by l , α is a vector of l variables. Each component α_i corresponds to a training examples (x_i, y_i) . The solution is the vector α for which (12) is minimized and the constraints (13) are fulfilled.

SVM is a two-class classifier. For each testing sample, we compare the coefficients of fourteen classifiers and assign a class with maximum coefficient as overall recognition output. If the output matches the character's label, it means that the testing character is correctly recognized. Otherwise, this character is misrecognized. For example, SVM0 classifier can be employed to distinguish character 0 from all other characters. The overall recognition is congregated from all SVM classifiers.

Fig. 8 shows the flowchart of MICR waveform recognition using fourteen SVM classifiers.

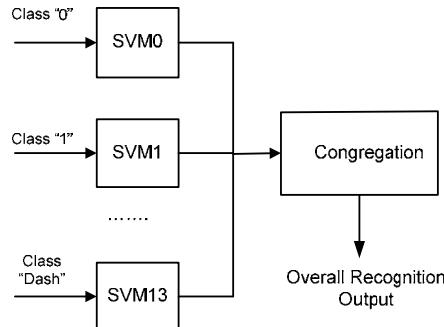


Fig. 8. Flowchart of MICR waveform recognition using SVMs

4.4 Feature Extraction

Feature extraction is a very important step for the image-based character recognition in the MICR area. Three feature extraction methods (Zhang, Bui & Suen, 2005) are used to construct two hybrid feature sets. The feature sets include: Directional-based Wavelet

Feature Set, Medial Axial Transformation (MAT) Gradient Feature Set, and Geometrical Feature Set.

4.5 Genetic Algorithm Used to Envolve Gating Network

A new combination scheme of classifiers is proposed in order to achieve the lowest error rate while pursuing the highest recognition rate for the recognition of E13B characters. The schematic diagram is shown in Fig. 6. The output confidence values of the ANN are weighted by $w_{1,0} \sim w_{1,13}$ and the output confidence values of the KNN classifier are weighted by $w_{2,0} \sim w_{2,13}$.

A genetic algorithm is used to evolve the optimal weights for the gating network from the confidence values of ANN classifier and KNN classifier.

Suppose the outputs of two classifiers are represented as: $\{c_{1,0}, c_{1,1}, \dots, c_{1,13}\}, \{c_{2,0}, c_{2,1}, \dots, c_{2,13}\}$.

The weighted outputs of the two classifiers' confidence values can be calculated as follows:

$$X_i = W_i^T \cdot C_i \quad (14)$$

where $W_i = [w_{i,0}, w_{i,1}, \dots, w_{i,13}]$, $C_i = [c_{i,0}, c_{i,1}, \dots, c_{i,13}] \quad i=1,2$.

Add two weighted confidence values into a Y vector:

$$Y = \sum_{i=1}^2 X_i \quad (15)$$

$$Y = [y_0, y_1, \dots, y_{13}]$$

In order to generalize the output, the j -th output g_j of the gating network is the "softmax" function of y_j as follows (Friedman, 1997):

$$g_j = \frac{e^{y_j}}{\sum_k e^{y_k}} \quad (16)$$

$$G = [g_0, g_1, \dots, g_{13}]^T$$

G is the output of the gating network.

Our goal is to pursue a lowest misrecognition rate and at the same time to seek the highest recognition performance. We can create a vector O_{target} with fourteen elements, which represent ten numerals and four symbols of the E13B fonts. In the vector, the value of the corresponding label is set equal to 1.0, while others are set equal to 0.0. A fitness function f is chosen to minimize the difference between the output G and the corresponding training sample vector O_{target} as follows:

$$f = |G - O_{target}|^2 \quad (17)$$

By minimizing the equation (17) through a genetic evolution, the weights tend to be optimal. Then, the recognition criterion is set as follows:

A recognition result is accepted if one of the following three conditions is satisfied:

- 1) ANN classifier and KNN classifier vote for the same character at the same time, where the sum of the confidence values is equal to or larger than 1.6;
- 2) The gating network votes for a character, where the confidence value of the gating network is larger than 0.85;
- 3) The confidence value of any classifier is larger than 0.95 and the gating network votes for the same character;

Otherwise, the character is rejected.

4.6 Genetic Algorithms for Training Gating Network

Genetic Algorithms have been developed based on Darwinian evolution and natural selection for solving optimization problems. GA applies evolution-based optimization techniques of selection, mutation, and crossover to a population for computing an optimal solution (Siedlecki & Sklansky, 1989). The problem of the weight selection in the gating network is well suited to the evolution by GAs.

In the ANN training procedure, the most difficult problem is to find a reasonable fitness function for a large set of training samples. Ideally, the recognition rate can be used as a fitness criterion for training a classifier. However, using the recognition rate this way is unfeasible for some pattern recognition problems because it requires huge computations for each generation of learning.

In this paper, we use GAs to train the gating network. When equation (17) is used as the fitness function, the GAs pursue the smallest difference between the gating network's outputs and the target label vector O_{target} . The following is a description of the steps used for our genetic algorithms.

Chromosome Representation

There are an ANN classifier, a KNN classifier in the system. Each classifier's outputs have fourteen nodes. A chromosome is a vector consisting of 28 weights. A chromosome is presented as:

$$\begin{bmatrix} w_{1,0} & w_{1,1} & \dots & w_{1,13} & w_{1,14} & w_{1,15} & \dots & w_{1,27} & w_{1,28} \end{bmatrix}$$

| --14 weights for ANN-- | | --14 weights for KNN-- |

Population Initialization

The initial chromosomes P (48 populations), are randomly created (0.0~1.0).

Selection

The best 24 chromosomes with minimum fitness values, taken from 48 populations in each generation, are chosen to go into the mating pool.

Fitness Computation

Equations (17) are used to calculate the fitness function.

Crossover

Crossover occurs when information is exchanged between two parent chromosomes and the new information is introduced to child chromosomes. A single offspring parameter value, w_{new} , comes from a combination of the two corresponding parent parameter values. The

crossover begins by randomly selecting a parameter a in a pair of parents, which is a crossover point. The crossover is calculated as follows:

$$\begin{aligned} a &= \text{roundup} \{ \text{random}(M - 1) \} \\ \text{parent 1(mother)} &= [w_{m0}, w_{m1}, w_{m2}, \dots, w_{ma}, \dots, w_{mM-1}] \\ \text{parent 2(father)} &= [w_{d0}, w_{d1}, w_{d2}, \dots, w_{da}, \dots, w_{dM-1}] \end{aligned} \quad (18)$$

where M is the length of the weight vector. The subscripts m and d in the weight parameters (w_{mi}, w_{di}) represent the mother and the father in the mating pool. Then, the selected parameters are combined to form new parameters. Two new weights are calculated as follows:

$$\begin{aligned} w_{new1} &= w_{ma} - \beta [w_{ma} - w_{da}] \\ w_{new2} &= w_{da} + \beta [w_{ma} - w_{da}] \end{aligned} \quad (19)$$

where β is a random value between 0.0 and 1.0. The next step is to exchange the right parts of two parents, consisting of the crossover point to the end for each parent.

$$\begin{aligned} \text{offspring 1} &= [w_{m0}, w_{m1}, w_{m2}, \dots, w_{new1}, \dots, w_{dM-1}] \\ \text{offspring 2} &= [w_{d0}, w_{d1}, w_{d2}, \dots, w_{new2}, \dots, w_{mM-1}] \end{aligned} \quad (20)$$

Mutation

In our experiments, the mutation rate is set at 0.01. According to the mutation rate, we randomly replace w_{mi} (w_{di}) with a new weight element, which is produced by multiplying the old weight value with a new uniform random number (0.0-1.0).

Termination Criteria

Termination occurs when either the number of iterations reaches its defined number or the fitness value converges so that the weights in the chromosome pool are stable.

5. Experiments

In order to see how the proposed system can improve system's reliability, we conducted the following three experiments. In all of the experiments, the E13B characters extracted from 250 personal checks (6250 characters and symbols) are used as training samples; another set of 6250 characters and symbols is used as testing samples. The training samples and testing samples are separated.

Experiment One

In this experiment, two classifiers: ANN classifier and KNN classifier are individually used to test the recognition performance on E13B image-based characters. Two hybrid feature sets are used to train the two classifiers, respectively. Table I lists the rejection rate, recognition rates, and reliability results conducted on ANN classifier.

Feature Set	Rejection Rate (%)	Recognition Rate (%)	Reliability (%)
Hybrid Feature Set I	0.00	98.50	98.50
Hybrid Feature Set II	0.00	98.69	98.69

Table 1. Recognition performance of ANN classifier trained by two hybrid feature sets

Note: Hybrid Feature Set I: Directional-based Wavelet Feature+20 Geometrical Features

Hybrid Feature Set II: MAT-based Gradient Feature + 20 Geometrical Features

We use the same training samples and testing samples for KNN classifier. The test result is shown in Table II.

Feature Set	Rejection rate (%)	Recognition Rate (%)	Reliability (%)
Hybrid Feature Set I	0.00	98.40	98.40
Hybrid Feature Set II	0.00	98.59	98.59

Table 2. Recognition performance of KNN classifier trained by two hybrid feature sets

From above tests, it can be concluded as follows:

- 1) There is no rejection rate since an individual classifier sets a threshold to either correctly recognize a testing character or mistakenly recognize it;
- 2) Two classifiers have a similar recognition rate trained by two feature sets;
- 3) Reliability is relatively low.

Experiment Two

In Experiment Two, a classifier fusion scheme, which consists of an ANN classifier, a KNN classifier, and a gating network to congregate the two classifiers, is tested without SVM verification module. Different feature sets are applied to train two classifiers. There are four options:

Combo I: ANN trained by Hybrid Feature Set I+KNN trained by Hybrid Feature Set I+gating network

Combo II: ANN trained by Hybrid Feature Set I+KNN trained by Hybrid Feature Set II+gating network

Combo III: ANN trained by Hybrid Feature Set II+KNN trained by Hybrid Feature Set I+gating network

Combo IV: ANN trained by Hybrid Feature Set II+KNN trained by Hybrid Feature Set II+gating network

The recognition rate, rejection rate, and reliability are conducted on the four fusion schemes. The test results are listed in Table III. Since the classifier fusion system introduces a rejection option, some characters with a relative low confidence value are rejected. The rule to reject characters in the classifier fusion system was described in the last part of Section 4.5.

Classifier Fusion Scheme	Rejection Rate (%)	Recognition Rate (%)	Reliability (%)
Combo I	0.69	98.60	99.29
Combo II	0.72	98.70	99.42
Combo III	0.80	98.67	99.47
Combo IV	0.72	98.64	99.36

Table 3. Recognition performance of classifier fusion system, consisting of ANN, KNN and gating network, trained by different hybrid feature sets

It is observed that some checks have been severely folded. As a result, character images are deteriorated and noises are added on the check images, which affected recognition rate.

Although a rejection strategy is introduced in the tests and the reliability is increased; however, some recognition errors still remain.

For example, in the Combo III experiment, the recognition rate is 98.67%. The rejection rate is 0.80% and the reliability is as high as 99.47%. As such, there are still 33 misrecognized characters, which is unacceptable in an automatic bank check processing system.

Experiment Three

In order to pursue an excellent reliability in the system, we propose a verification module. Firstly, SVM classifiers are used to recognize the segmented one dimensional MICR waveforms. Then, the recognition results are used to verify the recognition results of the classifier fusion system. The recognition rate of waveform-based MICR is 99.52%, which means that 30 characters out of 6,250 testing samples were misrecognized. The reliability of the SVMs is also 99.52%.

The verification rule is explained as follows: if the classifier fusion system votes a character and SVMs also vote the same character, the recognition is confirmed; otherwise, the testing sample is rejected.

Since two classifications use entirely different input signals (the classifier fusion system uses image-based OCR method, whereas the SVM classifier uses one dimensional MICR waveform), the overall reliability is significantly increased. Table VI shows the overall recognition rate, rejection rate and reliability of the classifier fusion system with SVM verification module.

Classifier Fusion Scheme + SVM Verification Module	Rejection Rate (%)	Recognition Rate (%)	Reliability (%)
Combo I+ SVM Module	1.80	98.19	99.99
Combo II+ SVM Module	1.66	98.34	100
Combo III+ SVM Module	1.70	98.30	100
Combo IV+ SVM Module	1.80	98.15	99.95

Table 4. Recognition performance of classifier fusion system with SVMs verification module

Comparing Table IV with Table III, it can be concluded that the system's reliability has been improved significantly. Both Combo II+SVM Module and Combo III +SVM Module achieve 100% reliability and have recognition rates over 98.30%. The remaining characters will be rejected and will be processed manually.

There are a few reasons behind the better recognition performance:

- 1) ANN classifier and KNN classifier are trained using different feature sets, which makes the two classifiers in the fusion system complementary;
- 2) Gating network can enhance recognition rate and reliability;
- 3) SVM verification module is trained by different signal input, which ensures that the overall system reliability will increase.

Fig. 9 shows the reliability comparison between the classifier fusion system and the classifier fusion system with SVM verification module.

Fig. 10 shows the reliability improvement from individual classifier to classifier fusion system (including an ANN, a KNN, and a gating network), and to the fusion classifier system with

a SVM verification module. Experiments demonstrated that the reliability increases from 98.5% to nearly 100%.

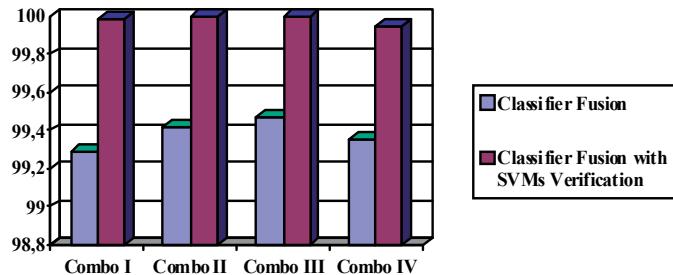


Fig. 9. Reliability comparsion of classifier fusion system and the system with SVM verification module

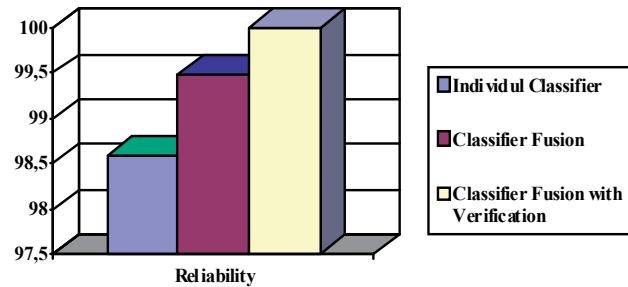


Fig. 10. Reliability improvement from individul classifier to classifier fusion system with SVM verification module

6. Conclusions

In this paper, we proposed a novel classifier fusion system to congregate the recognition results of an ANN classifier and a modified KNN classifier. The recognition results are verified by the recognition results of SVM. As two entirely different classification techniques (image-based OCR and 1-D digital signal SVM classification) are applied to the system, experiments have demonstrated that the proposed classifier fusion system with SVM verification module can significantly increase the system's recognition reliability and can suppress misrecognition rate at the same time.

In the future, the theory foundation of classifier fusion with rejection strategies will be further investigated. It is expected that the theory will be employed to solve more complex pattern recognition problems.

Acknowledgements: Part of gating network and genetic algorithm research was conducted at Centre for Pattern Recognition and Machine Intelligence (CENPARMI), Concordia University, Canada. Author wishes to thank Professors and colleagues of CENPARMI for their help.

7. References

- Chow, C. K. (1970), On Optimum Recognition Error and Reject Tradeoff. *IEEE Transactions on Information Theory*, Vol-16, No. 1, pp. 40-46.
- Decoste, D. & Scholkopf, B. (2002), Training Invariant Support Vector Machines. *Machine Learning*, Vol. 46, No. 1-3, pp.160-190.
- Duda, R. O.; Hart, P. E. & Stork, D. G. (2000). *Pattern Classification*, John Wiley & Sons, Inc., Wiley-Interscience, Second Edition.
- Frelicot, C. & Mascarilla, L. (2002). Reject Strategies Driven Combination of Pattern Classifiers, *Pattern Analysis and Applications*, Vol. 5, No. 2, pp. 234-243.
- Friedman, J. H. (1997). On Bias, Variance, 0/1-loss and the Curse-of-dimensionality, *Data Mining and Knowledge Discovery*, Vol. 1, No. 1, pp.55-77.
- Giusti, N.; Masuli, F. & Sperduti, A. (2002). Theoretical and Experimental Analysis of A Two-stage System for Classification, *IEEE Transactions on PAMI*, Vol-24, No. 7, pp. 893-904.
- Heisele, B.; Serre, T.; Prentice, S. & Poggio, T. (2003). Hierarchical Classification and Feature Reduction for Fast Detection with Support Vector Machines, *Pattern Recognition*, Vol. 36, No. 9, pp.2007-2017.
- http://en.wikipedia.org/wiki/Magnetic_ink_character_recognition
- Keller, J. M., Gray, M. R. & Givens Jr.; J. A. (1985). A fuzzy K-Nearest Neighbor Algorithm, *IEEE Trans. on SMC*, Vol.SMC-15, No.4, pp. 580-585.
- Kittler, J; Hatef, J; Duin, R. P. & Matas, J. (1998). On Combining Classifier, *IEEE Transactions on PAMI*, Vol. 20, No. 3, pp. 226-239.
- Kuncheva, L. I. (2002). A Theoretical Study on Six Classifier Fusion Strategies, *IEEE Transactions on PAMI*, Vol. 24, No. 2, pp. 281-286.
- Liu, C. L.; Nakashima, K.; Sako, H. & Fujisawa, H. (2004). Handwritten Digit Recognition: Investigation of Normalization and Feature Extraction Techniques, *Pattern Recognition*, Vol. 37, No. 2, pp.265-279.
- Siedlecki, W. & Sklansky, J. (1989). A Note on Genetic Algorithm for Large-scale Feature Selection, *Pattern Recognition Letters*, Vol. 10, No. 5, pp.335-34.
- Suen, C. Y. & Tan, J. (2005). Analysis of Errors of Handwritten Digits Made by A Multitude of Classifiers, *Pattern Recognition Letters*, Vol. 26, No. 1, pp. 369-379.
- Zhang, P. ; Bui, T. D. & Suen, C.Y. (2007). A Novel Cascade Ensemble Classifier System with A High Recognition Performance on Handwritten Digits, *Pattern Recognition*, Vol. 27, No. 12, pp. 3415-3429.
- Zhang, P. ; Bui, T. D. & Suen, C.Y. (2005). Hybrid Feature Extraction and Forest Feature Selection for Increasing Recognition Accuracy of Handwritten Numerals, *in the Proceedings of 8th International Conference on Document Analysis and Recognition (ICDAR)*,
- Zimmermann, M.; Bertolami, R. & Bunke, H. (2002). Rejection Strategies for Offline Handwritten Sentence Recognition, *in the Proceedings of the 17th International Conference on Pattern Recognition (ICPR2002)*, Vol. 2, Quebec, Canada, pp. 550-553.

Watermarking Representation for Adaptive Image Classification with Radial Basis Function Network

Chi-Man Pun
*University of Macau
Macau SAR
China*

1. Introduction

Rapid continual advances in computer and network technologies coupled with the availability of relatively cheap high-volume data storage devices have effected the production of thousands of digital images everyday. Therefore, many content-based image retrieval (CBIR) systems have been proposed to cope with such huge image archives. To facilitate image retrieval from the huge volume image repositories, there is a great need to search for effective content-based image features. Traditionally, the most straightforward way to implement image database management systems is to make use of the conventional database-management systems (DBMS) such as relational databases or object-oriented databases. Such systems are usually keyword-based, in which the image attributes, usually in the form of text annotations, are extracted manually or partially computed and managed within the framework of a conventional DBMS, such as Chabot (Ogle and Stonebraker 1995) Piction(Srihari 1995), Photobook(Pentland, Picard et al. 1996), WebSeer(Swain, Freankel et al. 1997),etc.. However, the keyword-based approach provides limited capacity for retrieving visual information. In most cases, the associated image attributes cannot fully describe the contents of the imagery by themselves. Since the image attributes are annotated manually or semi-automatically, the process of feature extraction is extremely time-consuming and labor-intensive. Current researches on CBIR systems (Belongie, Carson et al. 1998; Gupta 1995; Smith and Chang 1996; Tao, Tang et al. 2006) mostly focus on the capability of visual search, i.e., images are retrieved based on a certain similarity criterion for a user provided sample images or sketch. These systems employ visual information indexing scheme and approximate matching instead of the exact matching used in conventional DBMS. However, most of these methods involve a high computational complexity for its feature extraction. On the other hand, with the rapid development of digital multimedia technology, different digital watermarking schemes have been proposed to address the issue of multimedia copyright protection. Many of robust watermarking schemes are using the frequency domain approach. Most of these approaches are based on discrete Fourier transform (DFT) (Pereira, Ruanaidh et al. 1999), cosine transform (DCT)

(Cox, Kilian et al. 1997; Hernandez, Amado et al. 2000; Piva, Barni et al. 1997) or wavelet transform (DWT) (Hsieh, Tseng et al. 2001 ; Pun and Kong 2007; Wang and Kuo 1998; Wang and Lin 2004), and usually have fast watermarking detection.

In this chapter, a novel approach using watermarking representation for adaptive image classification with Radial Basis Function (RBF) network is proposed. The original image is decomposed into wavelet coefficients using discrete wavelet packet transform. The energy signatures of most dominant sub-bands are extracted adaptively to form a reduced feature vector which is to be encoded as a binary watermark. The watermark is embedded by quantization into the wavelet coefficients with highest magnitudes except for those in the lowest frequency channel. Then the image features can be extracted from the watermarked image by a fast discrete wavelet packet transform and de-quantization. The extracted image features are fed to the trained RBF network for image classification. The outline of this chapter is organized as follows. In next section, we briefly introduce and review the standard 2-D discrete wavelet packets transform techniques. In section III, we present our proposed algorithm for embedding image features by watermarking and the algorithm for extracting the image features from the watermarked image. In section IV, the algorithm for adaptive image classification with RBF network is proposed. The experiment results for robustness and classification accuracy of our proposed method to various attacks, and the efficiency comparison results with other image classification method are presented in Section V. Finally, conclusions are drawn in Section VI.

2. Discrete Wavelet Packet Transform

The 2-D discrete wavelet packet transform (DWPT) is a generalization of 2D discrete wavelet transform (DWT) that offers a richer range of possibilities for image analysis. In 2D-DWT analysis, an image is split into an approximation and three detail images. The approximation image is then itself split into a second-level approximation and detail images, and the process is recursively repeated. So there are $n+1$ possible ways to decompose or encode the image for an n -level decomposition. In 2D-DWPT analysis, the three details images as well as the approximation image can also be split. So there are $4n$ different ways to encode the image, which provide a better tool for image analysis. The standard 2D-DWPT can be described by a pair of quadrature mirror filters (QMF) H and G (Mallat 1989). The filter H is a low-pass filter with a finite impulse response denoted by $h(n)$. And the high-pass G with a finite impulse response is defined by:

$$g(n) = (-1)^n h(1-n), \text{ for all } n \quad (1)$$

The low-pass filter is assumed to satisfy the following conditions for orthonormal representation:

$$\sum_n h(n)h(n+2j) = 0, \text{ for all } j \neq 0 \quad (2)$$

$$\sum_n h(n)^2 = 1 \quad (3)$$

$$\sum_n h(n)g(n+2j) = 0, \text{ for all } j \quad (4)$$

The 2D discrete wavelet packet decomposition of an $M \times N$ discrete image x up to level $p+1$ ($0 \leq p \leq \min(\log_2(N), \log_2(M))$) is recursively defined in terms of the coefficients of level p as follows:

$$C_{4k,(i,j)}^{p+1} = \sum_m \sum_n h(m)h(n)C_{k,(m+2i,n+2j)}^p \quad (5)$$

$$C_{4k+1,(i,j)}^{p+1} = \sum_m \sum_n h(m)g(n)C_{k,(m+2i,n+2j)}^p \quad (6)$$

$$C_{4k+2,(i,j)}^{p+1} = \sum_m \sum_n g(m)h(n)C_{k,(m+2i,n+2j)}^p \quad (7)$$

$$C_{4k+3,(i,j)}^{p+1} = \sum_m \sum_n g(m)g(n)C_{k,(m+2i,n+2j)}^p \quad (8)$$

where $C_{0,(i,j)}^0 = x_{(i,j)}$ is given by the intensity levels of the image x .

Since the image x has only a finite number of pixels, different methods such as symmetric, periodic or zero padding should be used for the boundary handling. At each step, we decompose the image C_k^p into four quarter-size images C_{4k}^{p+1} , C_{4k+1}^{p+1} , C_{4k+2}^{p+1} and C_{4k+3}^{p+1} . The inverse wavelet packet transform of a discrete image x from wavelet coefficients at level $p+1$ can be achieved by applying recursively the following formulae until $C_{0,(i,j)}^0$ is obtained:

$$\begin{aligned} C_{k,(i,j)}^p = & \sum_m \sum_n h(m)h(n)C_{4k,(m+2i,n+2j)}^{p+1} + \\ & \sum_m \sum_n h(m)g(n)C_{4k+1,(m+2i,n+2j)}^{p+1} + \\ & \sum_m \sum_n g(m)h(n)C_{4k+2,(m+2i,n+2j)}^{p+1} + \\ & \sum_m \sum_n g(m)g(n)C_{4k+3,(m+2i,n+2j)}^{p+1} \end{aligned} \quad (9)$$

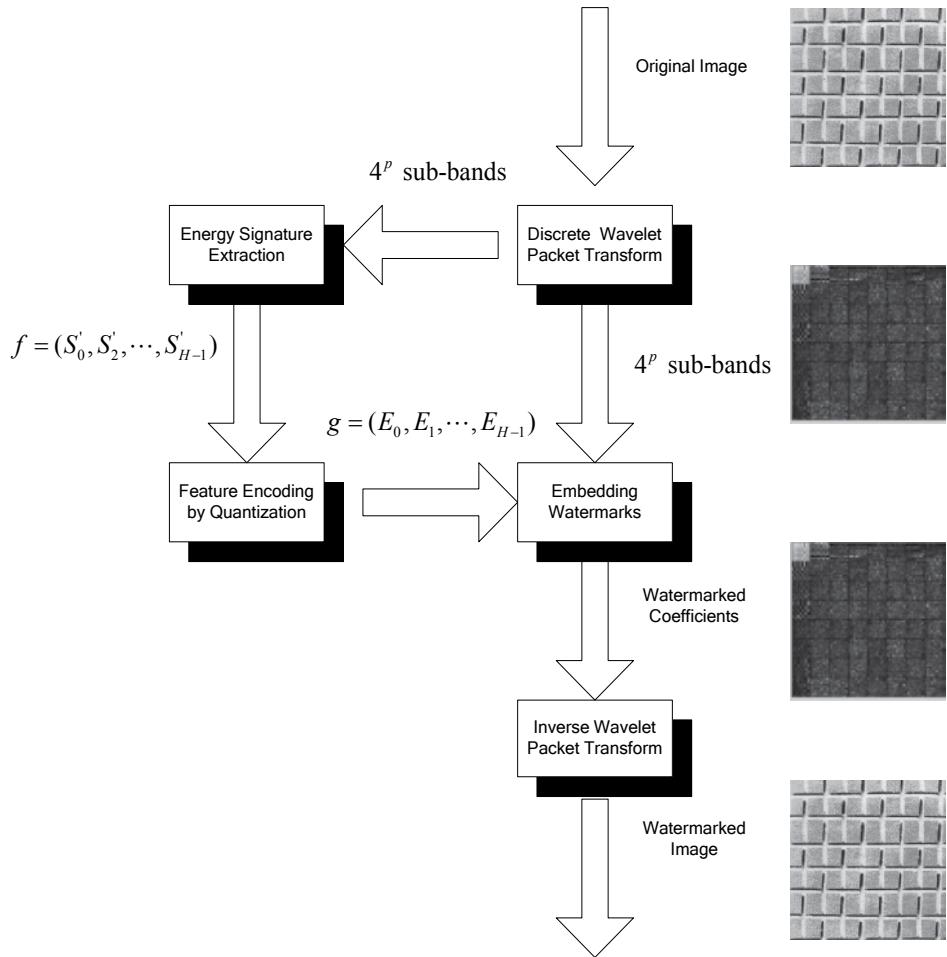


Fig.1. Procedure of embedding image features as digital watermark into the original image for image analysis.

3. Watermarking Representation of Image Features

3.1 Embedding Image Features as Digital Watermark

The procedure of embedding image features as digital watermark into the original image for image analysis or retrieval is depicted in Fig. 1. The $M \times N$ original image is decomposed into wavelet coefficients by a 2D discrete wavelet packet transform up to level p . An energy signature is computed for each sub-band of wavelet coefficients. However, the number of energy signatures for texture classification can be still very large. As suggested by Chang and Kou (Chang and Kuo 1993) the most dominant frequency sub-band provide very useful information for discriminating images. Therefore, we sort all energy signatures and choose only H most dominant energy signatures (with highest energy values) as feature vector. This feature vector is then encoded in binary feature vector, which are embedded back to the wavelet sub-bands. In order to have better perceptual invisibility, the feature vector is

embedded into the largest wavelet coefficients in each sub-band except the lowest frequency sub-band. To improve the robustness to various attacks, the same feature vector is embedded several times in remaining unused sub-bands. Finally, the inverse discrete wavelet packet transform is applied to obtain the watermarked image. The details of the algorithm are as follows:

Algorithm I: Embedding image features

- Step 1.** For a given $M \times N$ image, apply the p-level discrete wavelet packet transform (as described in section 2) to generate 4^p sub-bands of wavelet coefficients $C_{k,(i,j)}^p$, where $p \leq \log_2(N)$, $k \in \{0, \dots, 4^p - 1\}$ and $i, j = 0, 1, \dots, 2^{\log N - p} - 1$.
- Step 2.** Compute an energy signature

$$S_k = \frac{1}{M \times N} \sum_{i=1}^M \sum_{j=1}^N |C_{k,(i,j)}^p|^2 \quad (10)$$

for each sub-band of wavelet coefficients $C_{k,(i,j)}^p$, where $k \in \{0, \dots, 4^p - 1\}$.

- Step 3.** Arrange all energy signatures in descending order according to their values $S'_0, S'_1, \dots, S'_{4^p-1}$, and choose first H most dominant energy signatures (with highest energy values) as feature vector, $f = (S'_0, S'_1, \dots, S'_{H-1})$, where $H = \left\lfloor \frac{4^p}{\alpha} \right\rfloor$.

- Step 4.** Encode the feature vector f to a binary feature vector $g = (E_0, E_1, \dots, E_{H-1})$ by quantization, where each energy signature S'_n is represented by E_n with β bits, where $n = 0, \dots, H-1$.

- Step 5.** For first H sub-bands of wavelet coefficients C_k^p excluding C_0^p , embed the encoded feature E_n of binary feature vector $g = (E_0, E_1, \dots, E_{H-1})$ to the largest b coefficients $C_{k,q}^p$ in the sub-band by:

$$C_{n,q}^p = \begin{cases} C_{n,q}^p, & \text{if } \left\lceil \frac{C_{n,q}^p}{\Delta} \right\rceil \bmod 2 = (E_n)_q \\ C_{n,q}^p = C_{n,q}^p + \Delta, & \text{if } \left\lceil \frac{C_{n,q}^p}{\Delta} \right\rceil \bmod 2 \neq (E_n)_q \end{cases} \quad (11)$$

where $n = 0, \dots, H-1, q = 1, \dots, \beta$.

- Step 6.** Repeat Step 5 for the next H sub-band of wavelet coefficients for $\lfloor \alpha \rfloor$ times.
- Step 7.** Apply the Inverse discrete wavelet packet transform (as described in section 2) to obtain the watermarked image.

3.2 Extracting the Image Features

The procedure of extracting the image features in a watermarked image is depicted in Fig. 2. The watermarked $M \times N$ image is first decomposed into wavelet coefficients by the 2D discrete wavelet packet transform up to level p . The binary feature vector is then extracted from the sub-bands of wavelet coefficients. In order to improve the reliability, several feature vectors are extracted and combined. Finally, the image feature vector can be obtained by de-quantization for content-based image classification. The details of the algorithm are as follows:

Algorithm II: Extracting image features

- Step 1.** For a given $M \times N$ watermarked image, apply the p -level discrete wavelet packet transform (as described in section 2) to generate $4p$ sub-bands of wavelet coefficients $C_{k,(i,j)}^p$, where $p \leq \log_2(N)$, $k \in \{0, \dots, 4^p - 1\}$ and $i, j = 0, 1, \dots, 2^{\log N - p} - 1$.
- Step 2.** Extract the binary feature vector $g = (E_0, E_1, \dots, E_{H-1})$ from the largest b coefficients $C_{n,q}^p$ in the sub-band n at level p by:

$$(E_n)_q = \text{round}\left(\left(\sum_{i=0}^{\lfloor \alpha \rfloor - 1} \left\lceil \frac{C_{1+n+H*i,q}^p}{\Delta} \right\rceil \mod 2\right) / \lfloor \alpha \rfloor\right) \quad (12)$$

where $n = 0, \dots, H-1, q = 1, \dots, \beta$.

- Step 3.** Obtain the feature vector $f = (S'_0, S'_1, \dots, S'_{H-1})$ from a binary feature vector $g = (E_0, E_1, \dots, E_{H-1})$ by de-quantization, where $S'_n = E_n \times \frac{\max(S'_n)}{2^\beta}$, $n = 0, \dots, H-1$.

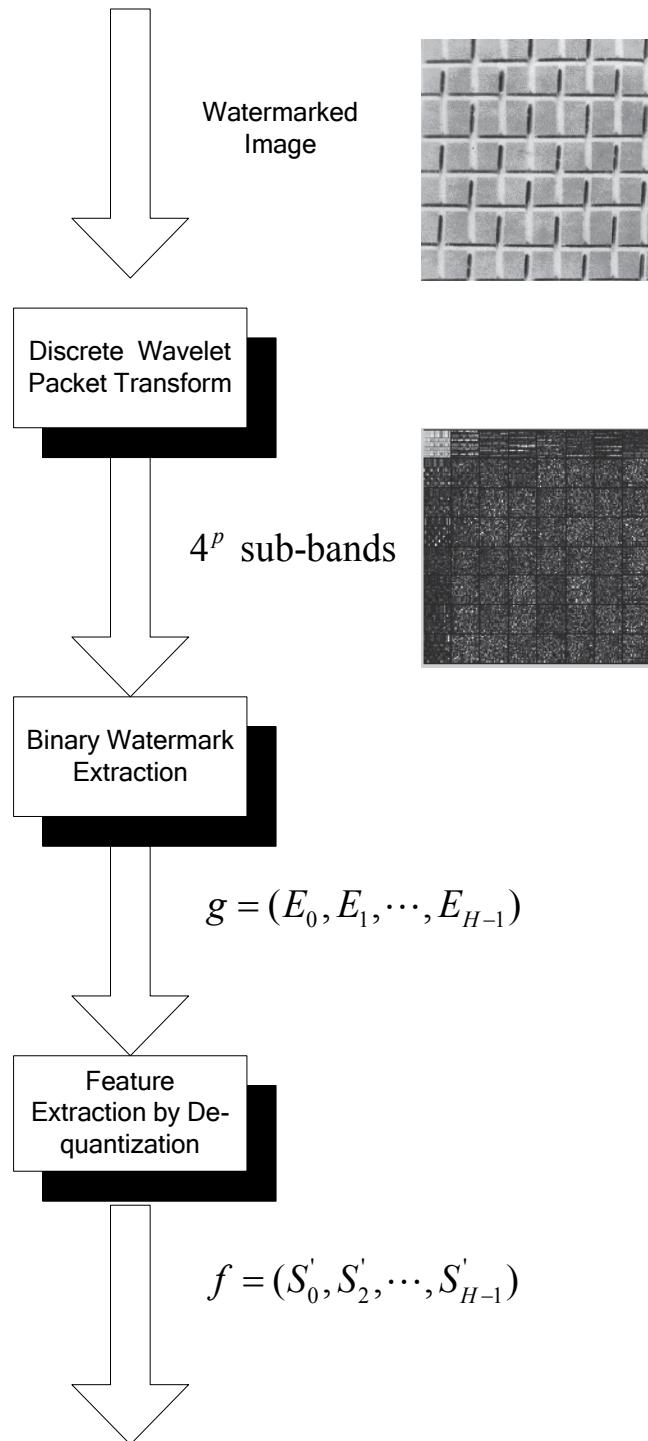


Fig.2. Procedure of extracting the image features in a watermarked image.

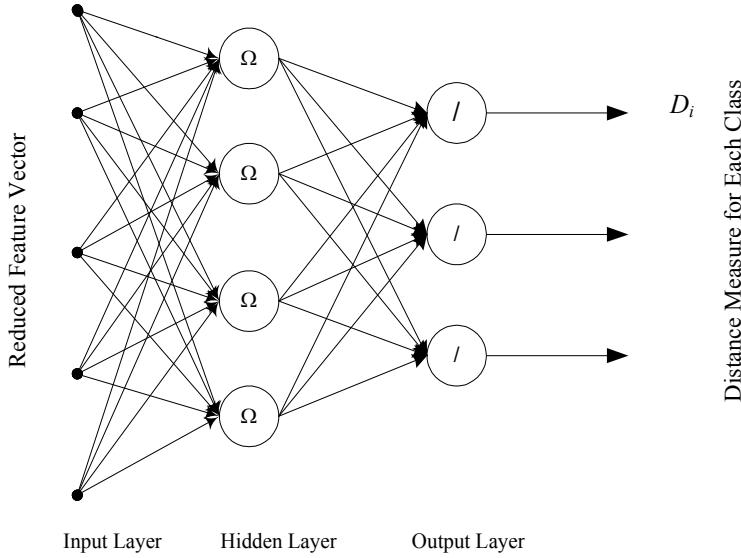


Fig. 3. Radial Basis Function (RBF) architecture.

4. Adaptive Image Classification with Radial Basis Function Network

The extracted image feature vector is used as inputs for the Radial Basis Functions (RBF) network used in the proposed adaptive classification algorithm. The RBF network involves three different layers, namely, input layer, hidden layer, and output layer, as shown in Fig. 3. The input layer is made up of a number of source / input nodes, one node for one energy signature from the reduced feature vector of a given query image. The goal of the hidden layer is to cluster the data and to further reduce its dimensionality. The output layer supplies the responses of the network to the reduced feature vector applied to the input layer during classification. The responses correspond to the distances between the input image and the different database image classes.

The proposed adaptive image classification algorithm can be divided into two stages. The first stage is for training, which is done only once. Its main objective is to construct an RBF network based on the number of features in the feature vectors and the number of classes involved, and to compute the corresponding weights of the hidden layer in the RBF network using a number of training images. The inputs to the RBF network include the feature vectors of the training image samples and their corresponding image classes. The output of the training would be the weights of the hidden layer of the network. The network starts with some initial weights which would be adjusted incrementally by the network as each feature vector and its class data are input. Therefore, the objective of the training is to produce the weights to represent the image classes of the training samples for achieving good classification results. Such weights would be used to classify query images during the classification stage. For efficiency sake, the training can be performed offline and the trained network information, including the weights, be saved for future use. The second stage is for online classification. Its main objective is to find the best match of any given query image to

one of the predefined classes captured in the trained RBF network. The details of the algorithm are as follows:

Algorithm III: Adaptive Image Classification Algorithm

Offline Training (for k training samples):

- Step 1.** For each training image i , compute a feature vector T_i by applying the *Algorithm II: Extracting image features*; where $i = 1, \dots, k$.
- Step 2.** Construct a Radial Basis Function (RBF) network, with m input nodes, $m-1$ hidden nodes, and the number of output nodes being equal to the number of image classes.
- Step 3.** For each training image i , input the feature values of T_i and the class C_j of image i to the RBF network; use the singular value decomposition (SVD) techniques (Bishop 1995) to compute the corresponding weights of the hidden layer of the RBF network by mapping the reduced feature vector T_i to the class C_j , where $i = 1, \dots, k$, and $j = 1, \dots, n$.
- Step 4.** Store the trained RBF network information to secondary storage.

Online Classification:

- Step 1.** Load the trained RBF network information from secondary storage and reconstruct the RBF network.
- Step 2.** Compute a feature vector S for a query image using the *Algorithm II: Extracting image features*.
- Step 3.** Feed the input layer of the RBF network with the reduced feature vector S .
- Step 4.** Compute the outputs of the hidden unit i in the hidden layer by:

$$radbas_i = \Phi_i(\|S - \mu_i\|) = \exp \left[-\sum_{k=1}^N \frac{(s_k - \mu_{ik})^2}{2c_i\sigma_{ik}^2 o} \right] \quad (13)$$

where Φ_i is a radial basis function; c_i is a proportionality constant for the variance σ_{ik}^2 ; s_k is the k th component of the input vector $s = [s_1, s_2, \dots, s_N]$, and μ_{ik} and σ_{ik}^2 are the k th components of the mean and variance vectors defining the Basis Functions (BF) respectively, and o is the overlap factor between BFs.

- Step 5.** Compute and output the feature distance D_j between the query texture image and class texture image j via output node j as follows:

$$D_j = \sum_i w_{ij} radbas_i + w_{0j} \quad (14)$$

where w_{ij} is the weight connecting the i th BF node to the j th output node, and w_{0j} is the threshold of the j th output node.

Step 6. Assign the query texture image to class i if $D_i \leq D_j$ for all $j \neq i$.

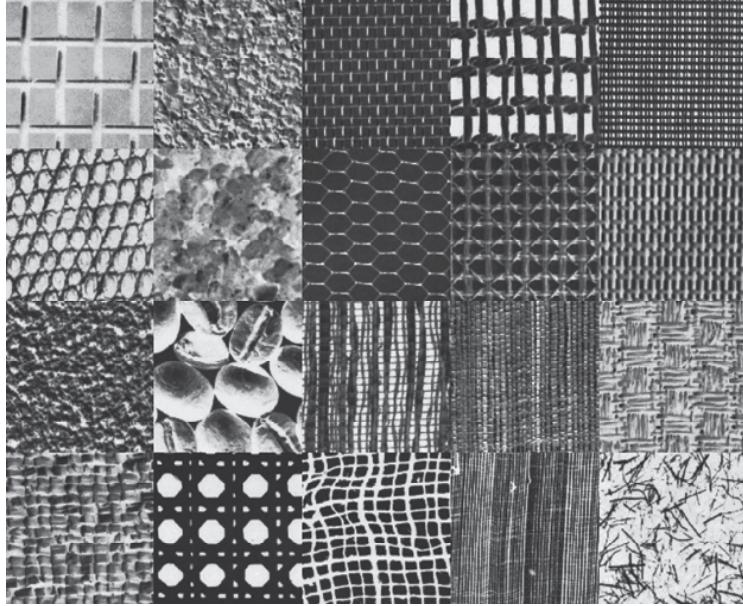


Fig. 3. Twenty class textures from Brodatz album. Row 1: D1, D4, D6, D20, D21. Row 2: D22, D28, D34, D52, D53. Row 3: D57, D74, D76, D78, D82. Row 4: D84, D102, D103, D105, D110

5. Experimental Results

In order to demonstrate the robustness and effectiveness of our proposed method, several experiments have been carried out based on a set of twenty classes of natural texture images as shown in Fig. 5, from the Brodatz's texture album (Brodatz 1996). Each texture is scanned with 150 dpi resolution, and each image, having the size 640×640 pixels and 256 gray levels, is divided into twenty-five 128×128 non-overlapping regions. So, a database of 500 (20×25) images was created for our testing. 200 of the texture images, with 10 images from each class, were used for training the RBF network, and the remaining 300 texture images form another dataset used for different watermarking and classification experiments. For embedding the image features by Algorithm I, a 20-tap Daubechies wavelet (Daubechies 1992) was used for discrete wavelet packet transform up to levels 3. The coefficients of the low-pass filter h of the 20-tap Daubechies wavelet transforms are listed in Table 1. For classification testing, a simple Euclidean classifier was used.

$h(0)$	0.01885858	$h(10)$	-0.02082962
$h(1)$	0.13306109	$h(11)$	0.02348491
$h(2)$	0.37278754	$h(12)$	0.00255022
$h(3)$	0.48681406	$h(13)$	-0.00758950

$h(4)$	0.19881887	$h(14)$	0.00098666
$h(5)$	-0.17666810	$h(15)$	0.00140884
$h(6)$	-0.13855494	$h(16)$	-0.00048497
$h(7)$	0.09006372	$h(17)$	-0.00008235
$h(8)$	0.06580149	$h(18)$	0.00006618
$h(9)$	-0.05048329	$h(19)$	-0.00000938

Table 1. 20-tap Daubechies wavelet transform filter coefficients.

First, we evaluate the perceptual quality of the watermarked images using the images in our database. Fig. 5 shows the original and the watermarked D1 image, which was embedded with 15 image features ($\alpha = 4.27$) encoded in 5 bits ($\beta = 5$). The two images are visually indistinguishable with PSNR is 41.5 dB.

Second, the experiments for verifying the robustness and classification accuracy of our method are carried out. Fig. 5 shows the watermarked D1 image attacked by Gaussian noise, JPEG compression and median filter. Table 2 shows the classification accuracy and robustness of our method for different attacks and number of dominant energy features. From the table, it was shown that the common attacks such as Gaussian noise, JPEG, and median filtering has only little effect on the classification performance. Our method has strong resistance to noise and JPEG compression with very low quality factor. The best performance was obtained using 47 features with 96.8% accuracy. The results also indicate that a higher number of dominant energy features does not imply a higher accuracy rate.

Third, the algorithm efficiency of our method was compared with other image classification method. Table 3 shows that our proposed method achieved the same classification accuracy, while having much lower complexity than other texture classification method such as wavelet packet signature method (Laine and Fan 1993).

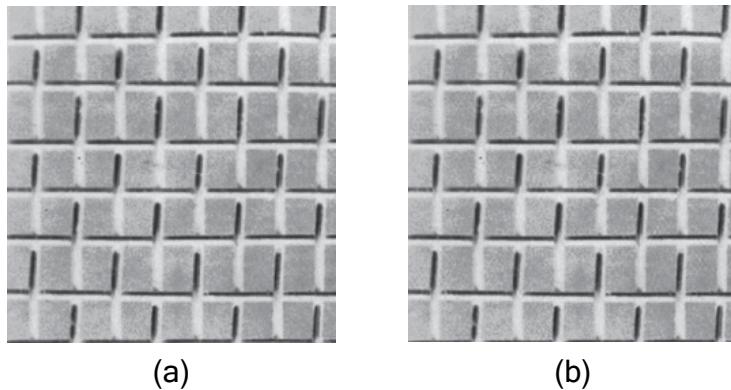


Fig. 4. (a) The original D1 image; (b) Watermarked D1 image with $\Delta = 33$, $\alpha = 4.27$, $\beta = 5$, PSNR = 41.5 dB.

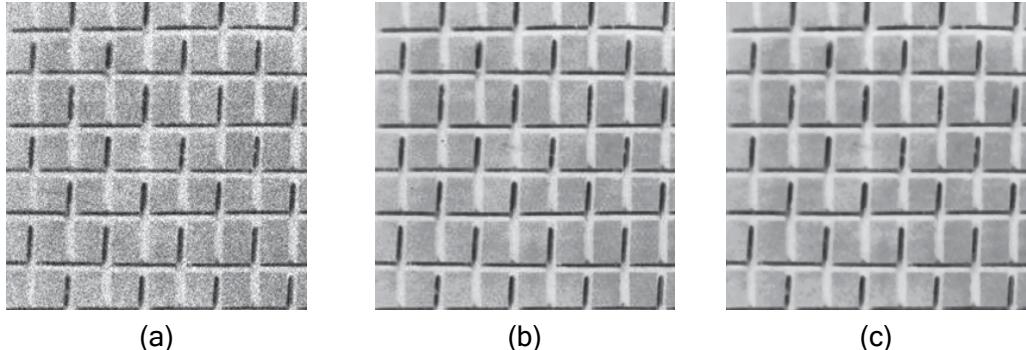


Fig. 5. Watermarked D1 image in Fig 4(b) attacked by (a) Gaussian noise 0.01; (b) JPEG quality factor 50; (c) 3x3 median filter.

Attacks	Number of image features					
	15	23	31	47	55	63
Gaussian Noise (0.001)	86.5	90.5	92.6	94.5	93.2	92.8
JPEG (QF = 50)	85.5	89.6	92.5	94.3	93.3	93.3
JPEG (QF = 30)	82.6	86.4	90.2	92.1	91.8	90.2
3x3 median filter	71.5	75.6	76.3	76.3	76.1	75.8
No attack	89.2	93.8	95.3	96.8	95.6	95.4

Table 2. Classification accuracy (%) with different attacks and number of image features.

	Proposed	WPS
Accuracy (%)	96.8	95.6
Complexity	$O(n)$	$O(n^2)$

Table 3. Performance comparison with the wavelet packet signature method.

6. Conclusion

In this chapter, a novel approach using watermarking representation for adaptive image classification with Radial Basis Function (RBF) network has been proposed. Experimental results show that the proposed method has strong resistance to noise and JPEG compression with very low quality factor, and has much better efficiency than the other image classification method. However, the performance for median filtering attacks still needs to be improved further. For image classification experiments, the best performance was obtained using only 47 features with 96.8% accuracy. Future work may focus on embedding more useful image features such as invariant features for image analysis.

Acknowledgments

This work was supported in part by the Research Committee of the University of Macau.

7. References

- Belongie, S., C. Carson, et al. (1998). Color- and Texture-Based Image Segmentation using EM and Its Application to Content-Based Image Retrieval. *Proceedings of the Sixth International Conference on Computer Vision*, pp. 675
- Bishop, C. (1995). *Neural Networks for Pattern Recognition* Clarendon Press. Oxford
- Brodatz, P. (1996). *Texture: A Photographic Album for Artists and Designers*. Dover
- Chang, T. and C. C. J. Kuo (1993). Texture analysis and classification with tree-structured wavelet transform. *IEEE Trans. Image Processing*, vol. 2, no. 4, pp. 429-441.
- Cox, I. J., J. Kilian, et al. (1997). Secure spread spectrum watermarking for multimedia. *IEEE Trans. Image Processing*, vol. 6, pp. 1673-1687.
- Daubechies, I. (1992). Ten Lectures on Wavelets. *CBMS-NSF Regional Conference Series in Applied Mathematics*, SIAM Press, Philadelphia, Pennsylvania.
- Gupta, G. (1995). *Visual information retrieval technology - a Virage perspective*. White paper, Virage Inc
- Hernandez, J. R., M. Amado, et al. (2000). DCT-domain watermarking techniques for still image: Detector performance analysis and a new structure. *IEEE Trans. Image Processing*, vol. 9, pp. 55-68.
- Hsieh, M.-S., D.-C. Tseng, et al. (2001). Hiding digital watermarks using multiresolution wavelet transform. *IEEE Trans. Industrial Electronics*, vol. 48 no. 5, pp. 875 - 882.
- Laine, A. and J. Fan (1993). Texture classification by wavelet packet signatures. *IEEE Trans PAMI*, vol. 15, no. 11, pp. 1186-1191.
- Mallat, S. (1989). A theory for multiresolution signal decomposition: The wavelet decomposition. *IEEE Trans. PAMI*, vol. 11, no. 7, pp. 674-693.
- Ogle, V. and M. Stonebraker (1995). Chabot: Retrieval from a Relational Database of Images *Computer*, vol. 28, no. 9, pp. 40-48.
- Pentland, A., R. W. Picard, et al. (1996). Photobook: Content-Based Manipulation of Image Databases. *International Journal of Computer Vision*, vol. 18, no. 3, pp. 233-254.
- Pereira, S., J. J. K. Ó. Ruanaidh, et al. (1999). Template based recovery of Fourier-based watermarks using log-polar and log-log maps. *IEEE Int. Conf. Multimedia Computing and Systems*, pp. 870-874.
- Piva, A., M. Barni, et al. (1997). DCT-based watermark recovering without resorting to the uncorrupted original image. *IEEE Int. Conf. Image Processing*, pp. 520-527.
- Pun, C.-M. and I.-K. Kong (2007). Adaptive Quantization of Wavelet Packet Coefficients for Embedding and Extraction of Digital Watermarks. *International Journal of Communications* vol. 1, no. 3, pp. 114-119.
- Smith, J. and S. F. Chang (1996). VisualSEEk: a fully automated content-based image query system. *Proceeding of ACM Multimedia 96*, pp. 87-98.
- Srihari, R. K. (1995). Automatic indexing and content-based retrieval of captioned images. *Computer*, vol. 28, no. 9, pp. 49-56.
- Swain, M. J., C. Freankel , et al. (1997). WebSeer: An Image Search Engine for the World Wide Web. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* pp.
- Tao, D., X. Tang, et al. (2006). Direct Kernel Biased Discriminant Analysis: A New Content-Based Image Retrieval Relevance Feedback Algorithm. *IEEE TRANS. MULTIMEDIA*, vol. 8, no. 4, pp. 716-727.

- Wang, H.-J. and C.-C. J. Kuo (1998). Image protection via watermarking on perceptually significant wavelet coefficient. *IEEE 2nd Workshop Multimedia Signal Processing*, pp. 279-284.
- Wang, S.-H. and Y.-P. Lin (2004). Wavelet tree quantization for copyright protection watermarking. *IEEE Trans. Image Processing*, vol. 13 pp. 154 -165.

Recent advances in Neural Networks Structural Risk Minimization based on multiobjective complexity control algorithms

D.A.G. Vieira, J.A. Vasconcelos and R.R. Saldanha

Department of Electrical Engineering

Federal University of Minas Gerais

Brazil

Nowadays, neural networks (NNs) are widely applied in the solution of several real world problems. They have been successfully used in many fields such as chemistry, physics, engineering, and bio-informatics among others. However, their use often relies on some hand-crafted settings, such as the number of layers and neurons. This chapter will discuss the Structural Risk Minimization (SRM) problem using some multiobjective optimization concepts. Both are closely related to the classical Tikhonov's regularization scheme, and, it is also exploited in this work.

A neural network is a learning machine capable to describe, to the input x , the set of functions $\mathbb{F} = \{f(x, w) : x \in \mathbb{X}, w \in \mathbb{W}\}$, where \mathbb{W} is the space of possible weights. Given a supervisor which defines an output vector $y \in \mathbb{Y}$ (desired output), for a given input x , according to the conditional distribution $F(y|x)$, the ultimate goal in the learning problem is to find $w \in \mathbb{W}$ that best approximates the supervisor answer given some measure. To some loss function $L(\cdot)$, the expected risk (error) can be defined as, Vapnik (1998):

$$R(w) = \int L(y, f(x, w)) dF(x, y). \quad (1)$$

Therefore, the learning problem can be understood as finding $f(x, w_0) : w_0 \in \mathbb{W}$, such that $R(w)$ is minimal. Nevertheless, the function $F(x, y)$ is unknown, thus, it is impossible to direct evaluate $R(w)$. The only available information about the supervisor is contained in the training set $\mathbb{T} = \{(x_1, \tilde{y}_1), \dots, (x_t, \tilde{y}_t)\}$. Where \tilde{y} is y plus some uncertainty, as noise. For instance, for regression and prediction problems $y \in \mathbb{R}^t$, and for binary classification $y \in \{-1, 1\}^t$.

In the early years of NN research, it was believed that decreasing the training error (empirical risk) was a sufficient condition to approximate the supervisor answer. This problem was stated as

$$w_* = \arg \min_{w \in \mathbb{W}} J(\mathbb{T}, w) = \frac{1}{t} \sum_{i=1}^t L(\tilde{y}_i, f(x_i, w)). \quad (2)$$

This approach was considered self-evident for many years and the main milestone was to find better algorithms to solve (2). However, the non self-evident overfitting phenomenon has appeared. This would imply that $w_* \neq w_0$. One way to characterize it is by the bias and

variance dilemma, S. Geman & Doursat (1992). The expected mean-squared error between $f(\cdot)$ and the expected value of y given x , $E[y|x]$, can be written as:

$$E_{\mathbb{T}}[(f(x; \mathbb{T}) - E[y|x])^2] = (E_{\mathbb{T}}[f(x; \mathbb{T})] - E[y|x])^2 + E_{\mathbb{T}}[(f(x; \mathbb{T}) - E_{\mathbb{T}}[f(x; \mathbb{T})])^2], \quad (3)$$

where $E_{\mathbb{T}}[\cdot]$ is the expected value given a set \mathbb{T} . The first term in the right hand side of (3) is known as bias, and the second one as variance. The variance term measures the sensibility of the approximating function given a data set \mathbb{T} . To control the variance, models with less complexity should be generated, i.e., they cannot change too much to a given data \mathbb{T} . On the other hand, some bias is inserted in the problem when the complexity is limited, thus, this should be controlled.

This chapter is organized as follows. First, the regularization theory from Tikhonov (1963), a well-known technique to solve linear ill-posed problems, will be introduced together with the residual method from Phillips (1962) and the quasi-solutions from Ivanov (1962; 1976). It is shown, using Singular Value Decomposition (SVD), the relationship between these methods and the Wiener's filter. After that, the Structural Risk Minimization (SRM), and the multiobjective learning will be discussed. These methods are closely related, and, some of their main aspects will be discussed. Inspired on the Tikhonov's regularization it will be discussed the well-known weight decay (WD) method for NNs, Hinton (1989). However, it will be clarified that this method is not consistent if the functions are not convex, which is usually the case. To overcome that, it is introduced the generalized Tikhonov's regularization based on a Q -norm for Parallel Layers Perceptrons (PLPs). Finally, some results are presented.

1. Linear ill-posed problems

Given the linear mapping $A : \mathbb{W} \mapsto \mathbb{Y}$, the equation

$$Aw = y, \quad A \in \mathbb{R}^{t \times n}, \quad w \in \mathbb{R}^n \text{ and } y \in \mathbb{R}^t, \quad (4)$$

is well-posed provided that: (i) for each $y \in \mathbb{Y}$, $\exists w \in \mathbb{W}$ such that $Aw = y$ (existence); (ii) $Aw_1 = Aw_2 \Leftrightarrow w_1 = w_2$ (uniqueness); (iii) A^{-1} is continuous (stability). Thus, a problem is called well-posed if its solution exists, is unique and stable. Unfortunately, inverse problems, such as the ones to select a model based on the data, are usually ill-posed, i.e., it violates at least one of the aforementioned conditions. In applied sciences and engineering the right-hand side vector y can be contaminated by noise, $\xi \in \mathbb{R}$, thus, instead of y only \tilde{y} is available and

$$\|y - \tilde{y}\|_2 \leq \xi. \quad (5)$$

The problem is said to be stable if small variations in the right-hand side implies small changes in the solution

$$\|w - \tilde{w}\|_2 \leq \delta(\xi). \quad (6)$$

The existence can be imposed by considering the minimal Euclidian norm

$$w_* = \arg \min_{w \in \mathbb{W}} J(w) = \|Aw - \tilde{y}\|_2^2 = (Aw - \tilde{y})^T (Aw - \tilde{y}). \quad (7)$$

Making $\partial J / \partial w = 0^1$,

$$w_* := (A^T A)^{-1} A^T \tilde{y} := A^\dagger \tilde{y}, \quad (8)$$

where A^\dagger is the pseudo inverse. Considering w_0 the desired solution of (4) and w_* the solution of (7), due to the error in y and the ill-conditioning in A , the following relation is usually true

$$\|w_*\| \gg \|w_0\|, \quad (9)$$

which is not a meaningful approximation of w_0 . In the early 60's, Tikhonov proposed the regularization method to solve this problem, Tikhonov (1963). The Tikhonov's method considers the solution of the following auxiliary problem:

$$w_\lambda = \arg \min_{w \in \mathbb{W}} J_\lambda = \|Aw - \tilde{y}\|_2^2 + \lambda \Omega(w), \quad (10)$$

where $\lambda > 0$ is a pre-defined constant known as regularization parameter. The regularization function $\Omega(w)$ is defined as semi-continuous, positive and compact in the space of functions defined by w , i.e., $\Omega(w) \leq c$, $c > 0$. To guarantee the uniqueness of the solution the following properties are required: (i) $\Omega(w)$ is a non-negative convex function; (ii) $\Omega(0) = 0$ holds true; and (iii) the $r(\rho) = \Omega(\rho w)$ is strictly growing function. This method is usually written as

$$\begin{aligned} w_\lambda &= \arg \min_{w \in \mathbb{W}} J_\lambda = \|Aw - \tilde{y}\|_2^2 + \lambda \|w\|_2^2 \\ &= (Aw - \tilde{y})^T (Aw - \tilde{y}) + \lambda x^T I x. \end{aligned} \quad (11)$$

For each positive parameter λ , considering the complexity $\Omega = w^T I w = \|w\|_2^2$, where I is the identity matrix, (10) has a unique solution of the following form:

$$w_\lambda := (A^T A + \lambda I)^{-1} A^T \tilde{y}. \quad (12)$$

This result was fundamental to the popularization of the Tikhonov's technique, since it has a simple closed-form solution. In statistics it is also known as ridge regression. In fact, Tikhonov & Arsenin (1977) proved that w_λ converges to w_0 as $\xi \rightarrow 0$ if

$$\lim_{\xi \rightarrow 0} \lambda(\xi) = 0, \quad (13)$$

$$\lim_{\xi \rightarrow 0} \frac{\xi^2}{\lambda(\xi)} = 0. \quad (14)$$

Consider the set $\mathbb{W}_k = \{w : \Omega(w) \leq c_k\}$, $c_k > 0$. Since Ω defines a compact subset the following holds true

$$\mathbb{W}_1 \subseteq \mathbb{W}_2 \subseteq \dots \subseteq \mathbb{W}_i, \dots \Rightarrow c_1 < c_2 < \dots < c_i, \dots \quad (15)$$

Define w_{k*} as

$$w_{k*} = \arg \min_{w \in \mathbb{W}_k} \|Aw - \tilde{y}\|. \quad (16)$$

¹ Using:

$$\frac{\partial(\tilde{y}^T w)}{\partial w} = \tilde{y}, \quad \frac{\partial(w^T A^T A w)}{\partial w} = (A^T A + A^T A)w$$

For some general conditions, Ivanov (1962; 1976) proved that the sequence w_{1*}, \dots, w_{k*} converges to w_0 , the desired solution. This is called quasi-solutions method and can be written, for some $\epsilon > 0$, as

$$\begin{aligned} w_\epsilon = \arg \min_{w \in \mathbb{W}} \|Aw - \tilde{y}\|_2^2 \\ \text{subject to: } \|w\|_2^2 \leq \epsilon \end{aligned} \quad (17)$$

In the same period Phillips (1962) proposed the residual method

$$\begin{aligned} w_\epsilon = \arg \min_{w \in \mathbb{W}} \|w\|_2^2 \\ \text{subject to: } \|Aw - \tilde{y}\|_2^2 \leq \epsilon \end{aligned} \quad (18)$$

In Vasin (1970) it is shown that the Regularization, Residual and Quasi-solutions methods are equivalent, i.e., they can generate the same set of solutions, given the linear problem stated in (4), and the distance measured using the Euclidian norm. Consider the problem stated in Alavetti & Eichel (2004)

$$\begin{aligned} w_\Delta = \arg \min_{w \in \mathbb{W}} \|Aw - \tilde{y}\|_2^2 \\ \text{subject to: } \|w\|_2^2 = \Delta \end{aligned} \quad (19)$$

Assuming that $\|w_0\| > \Delta$, this constrained minimization problem has a unique solution $w_{\lambda, \Delta}$ of the form (12). The value of λ is positive such that $\|w_\lambda\| = \Delta$ Alavetti & Eichel (2004). Assume that $A^\dagger \tilde{y} \neq 0$, the function

$$\varphi(\lambda) := \|w\|^2, \quad \lambda \geq 0, \quad (20)$$

can be expressed as

$$\varphi(\lambda) := \tilde{y} A (A^T A + \lambda I)^{-2} A^T \tilde{y}, \quad \lambda > 0, \quad (21)$$

which shows that $\varphi(\lambda)$ is strictly decreasing and convex for any $\lambda > 0$, and that, $\varphi(\lambda) = \Delta$ has a unique solution λ , such that $0 < \lambda < \infty$, for any Δ that satisfies $0 < \Delta < \|A^\dagger \tilde{y}\|^2$, Alavetti & Eichel (2004).

Even though all the results considered so far used the Euclidian norm $\|\cdot\|_2$ to define the complexity Ω , the more general p -norm

$$\|w\|_p = \sum_{i=1}^n |w_i|^p, \quad (22)$$

can also be applied. This is the case of the shrinkage method called Lasso, Hastie et al. (2001)

$$\begin{aligned} w_{lasso} = \arg \min_{w \in \mathbb{W}} \|Aw - \tilde{y}\|_2^2 \\ \text{subject to: } \|w\|_1 \leq \epsilon \end{aligned} \quad (23)$$

This chapter will concentrate in the Euclidian norm based formulation due to their simplicity, and the existence of closed form solutions. According to Hastie et al. (2001) it could be used any p besides 1, or 2, and that, indeed, we could try to estimate it from the data, but there is no results in this direction so far.

1.1 Wiener's filter interpretation

Consider the singular value decomposition (SVD) of A as

$$A = USV^T \quad (24)$$

where U and V are unitary matrices, i.e., $U^{-1} = U^T$, and $S = \text{diag}(s_1, s_2, \dots, s_t)$ is a diagonal matrix with $s_1 \geq s_2 \geq \dots \geq s_t \geq 0$, called the singular values of A . Thus, w_λ , given in (12), can be written as:

$$\begin{aligned} w_\lambda &= (VS^T U^T U S V^T + \lambda V I V^T)^{-1} V S^T U^T \tilde{y} \\ &= V(S^T S + \lambda I)^{-1} S^T U^T \tilde{y} \end{aligned}$$

$$w_\lambda = \sum_{i=1}^t \frac{s_i^2}{s_i^2 + \lambda} \frac{u_i^T \tilde{y}}{s_i} v_i, \quad (25)$$

where $\lambda \geq 0 : \frac{s_i^2}{s_i^2 + \lambda} \leq 1$ are the Wiener's filter weights. The SVD of the matrix A is related to the principal component analysis. Therefore, it implies that it shrinks more the directions with smaller variance. Next section will introduce the Weight Decay, the realization of the ideas presented in this section to Neural Networks.

2. Structural Risk Minimization principle

The structural risk minimization (SRM) was introduced by Vapnik and Chervonenkis and a description of it can be found Vapnik (1992), Vapnik (1998). One of the main achievements of the SRM is the introduction of the idea of capacity of a set of functions. It is based on some theoretical results that shows that the upper bound of the learning machine expected risk depends on: (i) the training error and, (ii) the machine capacity, defined as the VC dimension and its variations, Vapnik (2001). This inductive principle is directly applied in learning machines as the Support Vector Machines (SVMs). Following these considerations the SRM principle considers the minimization of two factors: the training error and the VC dimension.

Consider the function $J(\cdot, \cdot) : \mathbb{Z} \times \mathbb{W} \mapsto \mathbb{R}$, in which \mathbb{Z} and \mathbb{W} are arbitrary spaces. Taking its second argument $w \in \mathbb{W}$ as a parameter constrained to a set $\mathbb{W}_k \subset \mathbb{W}$, a set \mathbb{J} of functions $J(\cdot, w) : \mathbb{Z} \mapsto \mathbb{R}$ becomes defined for $w \in \mathbb{W}_k$. This set can be structured as a sequence of nested subsets $\mathbb{J}_k = \{J(\cdot, w), w \in \mathbb{W}_k\}$, such that

$$\mathbb{W}_1 \subset \mathbb{W}_2 \subset \dots \subset \mathbb{W}_i \dots \Rightarrow \mathbb{J}_1 \subseteq \mathbb{J}_2 \subseteq \dots \subseteq \mathbb{J}_i \dots \quad (26)$$

The sequence (26) should fulfill the following conditions: (i) the VC dimension, h_k , of each set \mathbb{J}_k is finite, and (ii)

$$h_1 \leq h_2 \leq \dots \leq h_i \dots \quad (27)$$

For any positive integer k , there is a finite positive scalar B_k such that $J(z, w) \leq B_k, \forall w \in \mathbb{W}_k$ and $z \in \mathbb{Z}$. The principle of SRM is oriented to find the values of w and k such that $w \in \mathbb{W}_k$, making the function $J(\cdot, w)$ minimize the empirical risk, while the set \mathbb{W}_k minimizes the structural risk.

2.1 Multiobjective Learning

The SRM can be interpreted as a bi-objective optimization problem, which considers the minimization of the empirical risk and the machine capacity. Instead of the integer index k , a straightforward generalization is to consider that the set \mathbb{W} is parameterized by a continuous parameter ζ . Given a training set \mathbb{T} , the SRM problem for this set can be written as:

$$(\text{SRM}): \min_{\zeta, w} \left\{ \begin{array}{l} J(\zeta, w) \\ \Omega(\zeta, w) \end{array} \right.. \quad (28)$$

in which J represents some empirical risk function, and Ω the complexity of the learning machine, for instance the fat-shattering dimension, Shawe-Taylor & Bartlett (1998).

Usually, it is not possible to minimize J and Ω simultaneously, because the optimum to one function hardly ever is the optimum to the other one. Thus, there is not a single optimum, but a set of them, when a multiobjective formulation is considered. In order to state the solutions of the SRM, the following definitions are required:

- (i) *Dominance*: A pair (ζ_a, w_a) dominates another pair (ζ_b, w_b) , which is denoted by $(\zeta_a, w_a) \prec (\zeta_b, w_b)$, if $J((\zeta_a, w_a)) \leq J((\zeta_b, w_b))$ and $\Omega((\zeta_a, w_a)) \leq \Omega((\zeta_b, w_b))$, with the strict inequality valid for at least one of the functions.
- (ii) *Pareto optimality*: A pair (ζ_*, w_*) is called Pareto-Optimal (PO) if there is no other feasible pair which dominates it.

By using these definitions, it is possible to generate the set of solutions called PO front, which have the best trade-off between the error and the machine complexity. All such solutions are candidate solutions for the SRM problem.

Examining (26) and (27) from the viewpoint of the Pareto Optimality of (44), it can be seen that in the nested sequence $\mathbb{J}_1 \subset \mathbb{J}_2 \subset \dots \subset \mathbb{J}_i \dots$, the minimal empirical error in the set is ordered as $J_{1*} \geq J_{2*} \geq \dots \geq J_{i*}$, where $J_{k*} := J(w_{k*})$ and

$$\begin{aligned} w_{k*} &= \arg \min_w J(w) \\ \text{subject to: } w &\in \mathbb{W}_k \end{aligned} \quad (29)$$

The solutions w_{k*} are Pareto-Optimal ones, each one associated to the corresponding sequence set \mathbb{J}_k . These are the solutions of the SRM problem. Any other function $J(\cdot, w)$ that is not a solution of any minimization problem of this form must be dominated, and cannot be a solution of the SRM problem. This will be the base of some novel results presented in this chapter. Defining the complexity as $\Omega(\zeta)$, it can be associated to some $\mathbb{W}(\zeta)$ defined by

$$\mathbb{W}(\zeta) = \{w : \|w\| < \zeta\} \quad (30)$$

and

$$\Omega(\zeta) = \zeta. \quad (31)$$

Given $\zeta_1 < \zeta_2$, this choice of $\mathbb{W}(\zeta)$ and $\Omega(\zeta)$ preserves the necessary relations:

- $\mathbb{W}(\zeta_1) \subset \mathbb{W}(\zeta_2)$;
- $J(\cdot, \zeta_1) \geq J(\cdot, \zeta_2)$;
- $\Omega(\zeta_1) < \Omega(\zeta_2)$.

As the minimization of the structural risk $\Omega(\zeta) = \zeta$ is equivalent to the minimization of the norm of w , the structural risk minimization principle becomes, in this case, stated in terms of w only:

$$(\text{SRM}): \min_w \begin{cases} J(\cdot, w) \\ \Omega(w) = \|w\| \end{cases} . \quad (32)$$

3. The Weight Decay for MLPs

The Multi-Layer Perceptron (MLPs) is a popular neural network which considers the neurons (or perceptrons) in cascade. Consider the input vector x , which includes the bias term, i.e., it is added an extra element equal to 1, the vectorial function Φ , and the weight matrix W

$$\begin{aligned} \Phi_1 &= \phi_1(W_1^T x) \\ &\downarrow \\ \Phi_q &= \phi_q(W_q^T \Phi_{q-1}) \end{aligned} , \quad (33)$$

then

$$f(x, w)^{\text{MLP}} = \Phi_q(\Phi_{q-1}(\dots \Phi_1(\cdot))) \quad (34)$$

where q is the number of layers, and ϕ is an activation function as hyperbolic tangent. For a weight matrix W and the vector w_j

$$w_j^T x = \sum_{i=0}^n w_{ji} x_i. \quad (35)$$

Therefore, the MLPs implement a nonlinear function of the sum of nonlinear functions. With one hidden layer, and m neurons, it can be written as:

$$f(x, w)^{\text{MLP}} = \sum_{i=1}^m W_2 i \phi(W_1^T x), \quad (36)$$

where $x \in \mathbb{R}^{n+1}$, $x_0 = 1$ is the bias, W_2 is a vector with m elements and W_1 is a matrix $((n+1) \times m)$. The vector w is defined as a vector which contains all the elements of W_i . Using the ideas from the regularization framework, the Weight Decay (WD) is a direct implementation of the Tikhonov's model to MLPs. The WD consists in writing a weighted sum of the Empirical risk, $J(\cdot)$, and the norm of the weight vector

$$w_\lambda = \arg \min_w J_\lambda^{\text{MLP}} = J(w, x) + \lambda \|w\|_2^2, \quad (37)$$

where $J(\cdot)$

$$J(w, x) = \frac{1}{2} \sum_{i=1}^t (f(x_i, w) - \tilde{y}_i)^2. \quad (38)$$

In Bartlett (1998) it was shown that the fat-shattering dimension, which is a generalization of the VC dimension, can be limited by limiting the weights of a given network. Limiting the fat-shattering dimension leads to a limit in the generalization error, Vapnik (1998; 2001). This gives support to the use of the norm of the weight vector as the complexity constraint. The main difference between the problem stated in (10) and (37) is that in the first one the risk is guaranteed to be convex, while in the second one it can be non-convex and even multi-modal. Next section will show that the weighted sum approach, which is the base of the WD method, is not consistent given non-convex problems.

3.1 The convexity issue

The WD approach is based on the general weighted sum function

$$J_\lambda(w) = \lambda J_1 + (1 - \lambda)J_2, \quad (39)$$

where $\lambda = [0, 1]$ controls the importance of the objectives. Consider the following non-convex unimodal one-variable functions:

$$J_1(w) = ((w - 1)^2 - \tanh(40w - 4))^2, \quad (40)$$

$$J_2(w) = 200w^2, \quad (41)$$

where the factor 200 was used only to simplify presentation. Given, $\lambda = 0.3$ and $\lambda = 0.6$, the following weighted sum functions can be written

$$J_a(w) = 0.3J_1 + (1 - 0.3)J_2, \quad (42)$$

$$J_b(w) = 0.6J_1 + (1 - 0.6)J_2. \quad (43)$$

The functions J_1 and J_2 and two possible weighted solutions, J_a and J_b are shown in Fig. 1. Note that the weighted functions have become multimodal, although the original functions were unimodal. The PO front for this problem is presented in Fig. 2 and it is composed of both convex and non-convex parts.

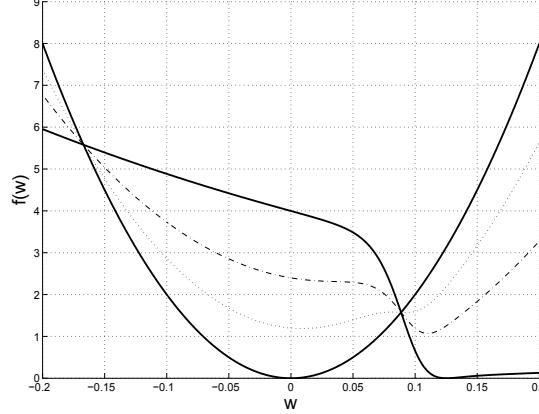


Fig. 1. The original functions are presented in continuous line (—). Two possible weighted solutions for this problem, with $\lambda = 0.3$ and $\lambda = 0.6$ are shown in (—.) and (---), respectively.

The relevant conclusion here is: if J_1 and J_2 are not convex functions (what is the case in most of machine learning problems), the weighted sum approach should not be employed for trying to find the trade-off front, as it may lose some potential solution.

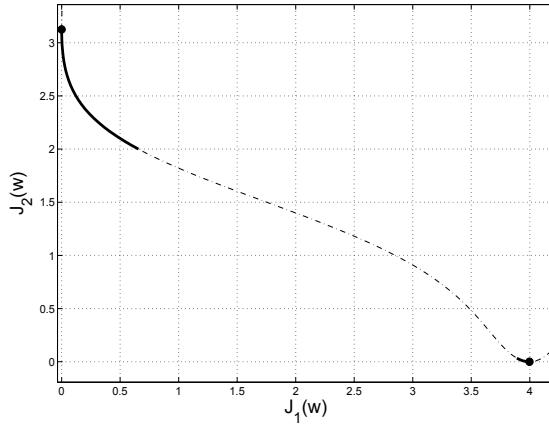


Fig. 2. The minima of J_1 and J_2 are marked with the \bullet , being the PO front everything between them. The convex part of the PO front is marked using continuous lines (-) and the non-convex as (-). The WD method can only generate the networks which belong to the convex part.

4. The Parallel Layer Perceptron

Instead of assembling the layers in cascade, in Caminhas et al. (2003) it was proposed to use them in parallel, given birth to the Parallel Layer Perceptron (PLP). Consider the input vector x , which includes the bias term, the vector function Φ , and the weight matrix W

$$\begin{aligned} \Phi_1 &= \phi_1(W_1^T x) \\ &\quad \downarrow \\ \Phi_q &= \phi_q(W_q^T x) \end{aligned}, \quad (44)$$

then

$$f(x, w)^{PLP} = \phi \left(\prod_{i=1}^q \Phi_i \right), \quad (45)$$

where, \prod represents a point wise product, and w is a vector with all the weights W_i . Hence, the PLP implements a nonlinear function of the product of nonlinear functions. This configuration has some computational advantages as discussed in Caminhas et al. (2003). A particular case of this topology can be written as the sum of the product of a linear layer, $L^T x$, and a nonlinear layer, $\Phi = \phi(N^T x)$, and it is given by

$$f(x, w)^{PLP} = x^T L \Phi^T = \sum_{j=1}^m \left[\sum_{i=0}^n L_{ji} x_i \phi \left(\sum_{i=0}^n N_{ji} x_i \right) \right]. \quad (46)$$

Since $f(x, w)^{PLP}$ is a linear function of the parameters L_{ji} , the PLP output can be written in a matrix form. Thus, consider the vector $l_z = L_{ji}$, where $z = (n+1)(j-1) + i$. This vector is a matrix transformation L to a vector with the same components, where $j = 1, \dots, m$, $i = 0, \dots, n$. By calculating all the outputs of the nonlinear perceptrons, a matrix A , with components $a_{kz} =$

$x_{zk}\phi(N_j^T x_i)$, $k = 1, \dots, t$ can be constructed

$$A = \begin{bmatrix} x_{01}\phi(b_{11}) & \dots & x_{n1}\phi(b_{m1}) \\ \vdots & \dots & \vdots \\ x_{0t}\phi(b_{1t}) & \dots & x_{nt}\phi(b_{mt}) \end{bmatrix}. \quad (47)$$

Therefore, the output of the PLP network can be written as

$$f(x, w)^{PLP} = A(x, N)l. \quad (48)$$

Thus, the empirical risk can be written as:

$$J^{PLP}(\mathbb{T}, w) = (Al - \tilde{y})^T(Al - \tilde{y}). \quad (49)$$

In this case the error is a quadratic function of the control variables - the vector l - while A is a nonlinear function of N . The main idea that will follow is to find a formulation, which resembles the Tikhonov's least squares solution, for this topology. Even though it is clear how to use the vector l , the nonlinear weights N brings an additional complication. To solve this problem it is necessary to find a function $\Omega(l)$ which is capable to consider also the complexity derived from N . For that, a generalized version of the Tikhonov's regularization, based on a Q -norm, can be used.

5. Generalized Tikhonov's regularization using a Q -norm

For any norm and any bijective linear transformation D , a new norm of l can be defined to be equal to $\|Dl\|$. For instance, in 2D, with D a rotation by 45 and a suitable scaling, this changes the 1-norm into an ∞ -norm. Consider the Euclidean norm of the transformed vector

$$\|Dl\|_2 = \sqrt{l^T D^T D l} = \sqrt{l^T Q l}, \quad (50)$$

for $Q = D^T D$, $w \in \mathbb{R}^n$ a vector with finite dimension, and $D^T D = Q \in \mathbb{R}^{n \times n}$ a symmetric positive definite matrix, i.e., $l^T Q l > 0$, $\forall l \neq 0$. The Q -norm of w is given by $\sqrt{l^T Q l}$. The regularization function Ω can be written as a Q -norm, where the matrix Q is a function of the nonlinear parameters N :

$$\Omega(l, N) = l^T Q(N)l. \quad (51)$$

Therefore, the solution of the linear ill-posed problem can be generalized as

$$l_Q = \arg \min_{l \in \mathbb{W}} J_\lambda = \|Al - \tilde{y}\|_{Q_1}^2 + \lambda \|l\|_{Q_2}^2 \quad (52)$$

Thus, it is need to define a matrix Q_2 such that it considers the influence of the nonlinear parameters of the PLP, while only adjusting the linear ones. This will be achieved using the Minimum Gradient Method (MGM).

5.1 The Minimum Gradient Method

By calculating the derivative of (46) with respect to x_k , one obtains, Vieira et al. (2008):

$$\frac{\partial f(x, w)}{\partial x_k} = \sum_{j=1}^m \left[\left(\frac{\partial \phi}{\partial b_j} N_{jk} \right) \left(\sum_{i=0}^n L_{ji} x_i \right) + \phi(b_j) L_{jk} \right]. \quad (53)$$

where $b_j = \sum_{i=0}^n N_{ji} x_i$. For all j and $z = (n+1)(j-1) + i$ the following holds true

$$\frac{\partial f(x, w)_j}{\partial x_k} = \left[\left(\frac{\partial \phi}{\partial b_j} N_{jk} x_k \right) + \phi(b_j) \right] l_z, \quad k = i, \quad (54)$$

$$\frac{\partial f(x, w)_j}{\partial x_k} = \left[\left(\frac{\partial \phi}{\partial b_j} N_{jk} x_i \right) \right] l_z, \quad k \neq i. \quad (55)$$

The derivatives in relation to the vector $x_k = [x_{k1}, \dots, x_{kh}, \dots, x_{kt}]^T$, where t is the number of samples, can be written in a vector form as follows:

$$\frac{\partial f(x, w)}{\partial x_k} = D_k l. \quad (56)$$

To exemplify the construction of the matrix D_k , where $D_k \in \mathbb{R}^{t \times (n+1)m}$, consider the following cases when the derivatives in relation to x_1 and x_2 are computed, for b_{jh} , N_{ji} , x_{ih} , where j represents the neuron, i the input and h the sample number.

$$D_1 = \begin{bmatrix} \frac{\partial \phi}{\partial b_{11}} N_{10} & \frac{\partial \phi}{\partial b_{11}} N_{11} x_{11} + \phi(b_{11}) & \dots & \frac{\partial \phi}{\partial b_{m1}} N_{mn} x_{n1} \\ \vdots & \vdots & \dots & \vdots \\ \frac{\partial \phi}{\partial b_{1t}} N_{10} & \frac{\partial \phi}{\partial b_{1t}} N_{11} x_{1t} + \phi(b_{1t}) & \dots & \frac{\partial \phi}{\partial b_{mt}} N_{mn} x_{nt} \end{bmatrix} \quad (57)$$

$$D_2 = \begin{bmatrix} \frac{\partial \phi}{\partial b_{11}} N_{10} & \frac{\partial \phi}{\partial b_{11}} N_{11} x_{11} & \frac{\partial \phi}{\partial b_{11}} N_{12} x_{21} + \phi(b_{11}) & \dots & \frac{\partial \phi}{\partial b_{m1}} N_{mn} x_{n1} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \frac{\partial \phi}{\partial b_{1t}} N_{10} & \frac{\partial \phi}{\partial b_{1t}} N_{11} x_{1t} & \frac{\partial \phi}{\partial b_{1t}} N_{12} x_{2t} + \phi(b_{1t}) & \dots & \frac{\partial \phi}{\partial b_{mt}} N_{mn} x_{nt} \end{bmatrix} \quad (58)$$

In the matrices D_k , when $i = k$, the columns related to the weights $L_{jk} = l_z$ are composed by two terms, as can be noticed in the second column of D_1 and in the third one of D_2 . In the other columns just one term is used. Remembers that $i = 0$ represents the bias term. Therefore, the complexity function Ω can be defined as the minimization of the norm of the output gradient

$$\Omega^{PLP} = \sum_{k=1}^n (D_k l)^T (D_k l) = l^T Q l, \quad (59)$$

where $Q \in \mathbb{R}^{(n+1)m \times (n+1)m} = \sum_{k=1}^n D_k^T D_k$. Clearly, $l^T Q l \geq 0 \forall l$, noticing that the sum of symmetric positive-definite matrices are also symmetric positive-definite matrices. The construction of the matrix $D_k^T D_k$ is exemplified taking $k = 2$:

$$D_2^T D_2 = \begin{bmatrix} \sum_{h=1}^t \left(\frac{\partial \phi}{\partial b_{1h}} N_{10} \right)^2 & \sum_{h=1}^t \left(\frac{\partial \phi^2}{\partial b_{1h}} N_{10} N_{11} x_{1h} \right) \\ \sum_{h=1}^t \left(\frac{\partial \phi^2}{\partial b_{1h}} N_{10} N_{11} x_{1h} \right) & \sum_{h=1}^t \left(\frac{\partial \phi}{\partial b_{1h}} N_{11} x_{1h} \right)^2 \\ \vdots & \ddots \\ \sum_{h=1}^t \left(\frac{\partial \phi}{\partial b_{1h}} \frac{\partial \phi}{\partial b_{mh}} N_{10} N_{mn} x_{nh} \right) & \sum_{h=1}^t \left(\frac{\partial \phi}{\partial b_{1h}} \frac{\partial \phi}{\partial b_{mh}} N_{11} N_{mn} x_{1h} x_{nh} \right) \\ \sum_{h=1}^t \left(\frac{\partial \phi}{\partial b_{1h}} N_{10} \right) \left(\frac{\partial \phi}{\partial b_{1h}} N_{12} x_{2h} + \phi(b_{1h}) \right) & \dots \sum_{h=1}^t \left(\frac{\partial \phi}{\partial b_{1h}} \frac{\partial \phi}{\partial b_{mh}} N_{10} N_{mn} x_{nh} \right) \\ \vdots & \dots \vdots \\ \sum_{h=1}^t \left(\frac{\partial \phi}{\partial b_{1h}} N_{12} x_{2h} + \phi(b_{1h}) \right)^2 & \ddots \vdots \\ \dots & \dots \sum_{h=1}^t \left(\frac{\partial \phi}{\partial b_{mh}} N_{mn} x_{nh} \right)^2 \end{bmatrix} \quad (60)$$

Since J^{PLP} and Ω^{PLP} are convex functions, the regularization based on the least-squares solution as presented in (52) does not loose any potential solution.

$$\begin{aligned} l_\lambda = \arg \min J_\lambda^{PLP} &= \lambda J^{PLP} + (1 - \lambda) \Omega^{PLP} \\ &= \lambda (Al - \tilde{y})^T (Al - \tilde{y}) + (1 - \lambda) l^T Q l. \end{aligned} \quad (61)$$

where the optimum l , (i.e., l_λ), can be calculated by making the derivative of (61) equal to zero. The derivative of (61) in relation to l can be calculated as:

$$\frac{dJ_\lambda^{PLP}}{dl} = \lambda (-2A^T \tilde{y} + 2A^T Al) + (1 - \lambda) 2Ql \quad (62)$$

In order to find l_λ , the previous relation should be made equal to zero, as given below:

$$\begin{aligned} \lambda (-2A^T \tilde{y} + 2A^T Al) + (1 - \lambda) 2Ql &= 0 \\ -2\lambda A^T \tilde{y} + 2\lambda A^T Al + (1 - \lambda) 2Ql &= 0 \\ [\lambda A^T A + (1 - \lambda) Q]l &= \lambda A^T \tilde{y} \\ l_\lambda &= [\lambda A^T A + (1 - \lambda) Q]^{-1} \lambda A^T \tilde{y}, \end{aligned} \quad (63)$$

if the matrix $[\lambda A^T A + (1 - \lambda) Q]$ is non-singular. The Pareto-Optimum set can be found by varying λ between zero and one. This work applied the golden section algorithm in the validation error criteria to define λ_* . The validation error for the given formulation is a convex function of the linear parameters l .

5.2 Generalized Singular Value Decomposition

Consider the following properties of the Generalized Singular Value Decomposition (GSVD) Hansen (1998):

$$GSVD = \begin{cases} A = U_A S_A V^T \\ D = U_D S_D V^T \\ S_A^T S_A + S_D^T S_D = I \end{cases}, \quad (64)$$

where U is a unitary matrix, i.e., $U^{-1} = U^T$, and $S_A = diag(s_{A1}, \dots, s_{A(n+1)m})$, $S_D = diag(s_{D1}, \dots, s_{D(n+1)m})$ such that $s_{A1} \geq \dots \geq s_{A(n+1)m} \geq 0$ and $s_{D(n+1)m} \geq \dots \geq s_{D1} \geq 0$. Applying (64) in (63) the following is obtained:

$$\begin{aligned} l_\lambda &= [\lambda A^T A + (1 - \lambda) Q]^{-1} \lambda A^T \tilde{y} \\ &= \lambda [V S_A^2 V^T + (1 - \lambda) V S_D^2 V^T]^{-1} V S_A U_A^T \tilde{y} \\ &= \lambda [V (\lambda S_A^2 + (1 - \lambda) S_D^2) V^T]^{-1} V S_A U_A^T \tilde{y} \\ &= \lambda [(\lambda S_A^2 + (1 - \lambda) S_D^2) V^T]^{-1} V^{-1} V S_A U_A^T \tilde{y} \\ &= \lambda (V^T)^{-1} [\lambda S_A^2 + (1 - \lambda) S_D^2]^{-1} S_A U_A^T \tilde{y} \end{aligned} \quad (65)$$

where $[\lambda S_A^2 + (1 - \lambda) S_D^2]$ is a diagonal matrix with elements $[\lambda s_{Ai}^2 + (1 - \lambda) s_{Di}^2]$. The unfiltered solution, disregarding the complexity control, i.e., $\lambda = 1$, is equal to

$$l_*(\lambda = 1) = (V^T)^{-1} S_A^{-1} U_A^T \tilde{y}. \quad (66)$$

The Wiener filter weights are evaluated comparing the unfiltered solution with the general solution of (65). The following is obtained

$$\begin{aligned} \Psi_i &= \frac{\lambda s_{Ai}^2}{\lambda s_{Ai}^2 + (1 - \lambda) s_{Di}^2} \\ &= \frac{1}{1 + \lambda' \frac{s_{Di}^2}{s_{Ai}^2}}, \end{aligned} \quad (67)$$

where $\lambda' = (1 - \lambda)/\lambda$, $\lambda \neq 0$ and s_{Ai}/s_{Di} are the generalized singular values. Similarly to the results using the simple SVD, the components with smaller singular values are filtered the most. Differently from the traditional Wiener filter, which only considers $s_{Di} = 1$, the MGM approach computes a general s_{Di} . It is possible to obtain $s_{Di} = 1$ using a identity matrix in the Q -norm. The Wiener filter weights define the relevance of each nonlinear neuron, filtering the unnecessary ones.

6. Results for benchmark problems

This section presents some experimental results in benchmarking problems considering the proposed ideas. Sigmoidal logistic functions have been used as PLP nonlinear activation function. Data sets from Intelligent data Analysis (IDA) repository are considered here as

presented in K. Muller & Scholkopf (2002). Table 1 summarizes the dimensionality of the input space, the number of training and test samples and the number of realizations for each data set. The results obtained by the PLP-MGM are compared with the results obtained by using the following machine learning techniques: (i) Support Vector Machine (SVM), (ii) kernel Fisher Discriminant (KFD), and (iii) Regularized AdaBoost (ABR) extracted from Muller et al. (2001); (iv)Leave-One-Out KFD (LKFD), and (v) Single objective Parallel Layer Perceptron (PLP) from Caminhas et al. (2003). The results are presented in Table 2.

Name	Dimension	Train	Test	Realizations
Banana	2	400	4900	100
B.Cancer	9	200	77	100
Diabetes	8	468	300	100
German	20	700	300	100
Heart	13	170	100	100
Image	1300	1010	18	20
S. Flare	9	666	400	100
Thyroid	5	140	75	100
Titanic	3	150	2051	100
Twonorm	20	400	700	100

Table 1. Ida repository data set summary.

	SVM	KFD	ABR	LKFD	PLP	PLP-MGM
Banana	11.5±0.7	10.8±0.5	10.9±0.4	10.4±0.4	10.7±.06	10.7±0.6
B. Cancer	26±5	26±5	27 ± 5	26 ± 4	27 ± 5	25 ± 4
Diabetes	23±2	23±2	24±2	23±2	23±2	23±2
German	24±2	24±2	24±2	24±2	30±3	24±2
Heart	16±3	16±4	17±4	16±4	19±3	16±3
Image	3.0±0.6	3.3±0.6	2.7±0.6	4.0±0.6	5±4	3.3±0.7
S. Flare	32±2	33±2	34±2	34±2	37±2	33±2
Thyr.	5±2	4±2	5±2	5±2	4±2	4±2
Titanic	22±1	23±2	23±1	22±1	23±1	22±1
Twon.	3.0±0.2	2.6±0.2	2.7±0.2	2.7±0.2	2.8±0.3	2.6±0.3

Table 2. Ida repository results.

The first noticeable result of Table 2 is that the PLP-MGM has outperformed the conventional PLP in most of the tested examples, and that PLP has never outperformed PLP-MGM. It is clear as well that the PLP-MGM has achieved similar results compared to those produced by the other approaches used for comparison.

7. Denoising Ground Penetrating Radar data

This section considers denoising Ground Penetrating Radar (GPR) using the PLP-MGM technique. This noise can be due to environmental conditions, geometric variations, and sensors characteristics. The numerical simulation follows the results described in Travassos et al. (2008). A block diagram of a typical GPR system to detect underground targets, is given in Fig. 3.

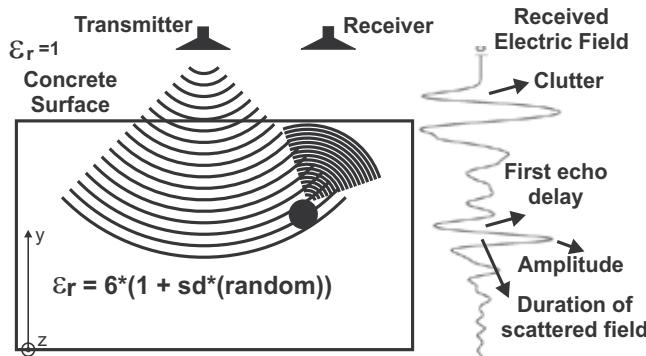


Fig. 3. The GPR problem.

The proposed configuration is tested to filter the noise of the scattered wave from a cylindrical air inclusion buried in a non-homogenous host medium, Vieira et al. (2009). Tables 3 and 4 considers white and colored Gaussian noise respectively. As the noise is stochastic by nature a statistical evaluation of the results is necessary. The simulations were done considering 20 different noises for each SNR, and a Neural Network trained for each of them. The results are presented in 3 and 4 they show a considerable improvement in the SNR, showing the effectiveness of the proposed approach.

Noise (dB)	SNR in the Filtered Wave (dB)		
	Mean	Max	Min
3	14.16	14.65	13.47
6	14.69	15.07	14.14
9	16.55	17.67	15.65
10	20.47	21.35	18.67

Table 3. SNR considering the GPR processed wave (filtered) by the proposed approach corrupted by White Gaussian Noise.

Noise (dB)	SNR in the Filtered Wave (dB)		
	Mean	Max	Min
3	12.76	13.22	11.96
6	15.30	16.21	14.17
9	20.36	20.73	19.96
10	20.58	20.93	20.09

Table 4. SNR considering the GPR wave processed (filtered) by the proposed approach corrupted by Colored Gaussian Noise.

8. Final Comments

This chapter described the use of the multiobjective optimization framework to train the Parallel Layer Perceptron network. This is based on the general concept that learning depends on two functions: the empirical risk and the network complexity. A formulation based on

the Tikhonov's regularization was proposed using a Q -norm as a complexity measure. This has a least-squares like closed form solution; therefore, it relies on simple computational algorithms. Moreover, it bores the good aspects of the Tikhonov's method. It opens discussions about other possible definitions of the matrix Q , different configurations of the PLP layers among others.

The results presented proved the effectiveness of the proposed approach. A wide comparison considering several benchmarking problems and algorithms were presented. Also a complex engineering problem was successfully solved using the proposed approach.

The relationships between the classical regularization, the structural risk minimization and the multiobjective formulation were also explored. These help the understanding concerning the nature of learning and their possibilities. It shows that the convexity is an important issue to the use of the WD method to MLPs. This is indeed a wide subject, and, due to space constraints, this chapter discussed a rather biased point-of-view on those subjects.

9. Acknowledgment

This work was supported by CNPq and FAPEMIG, Brazil.

10. References

- Alavetti, D. C. & Eichel, L. R. (2004). Tikhonov regularization with a solution constraint, *SIAM J. Sci. Comput.* (26).
- Bartlett, P. L. (1998). The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network., *IEEE Trans. on Information Theory* 2(44): 525–536.
- Caminhas, W. M., Vieira, D. A. G. & Vasconcelos, J. A. (2003). Parallel layer perceptron, *Neurocomputing* 3-4(55): 771–778.
- Hansen, P. C. (1998). *Rank-deficient and discrete ill-posed problems: numerical aspects of linear inversion*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA. ISBN 0-89871-403-6.
- Hastie, T., Tibshirani, R. & Friedman, J. H. (2001). *The Elements of Statistical Learning*, first edn, Springer.
- Hinton, G. E. (1989). Connections learning procedures, *Artificial intelligence* 40: 185–234.
- Ivanov, V. V. (1962). On linear problems which are not well-posed, *Soviet Math. Dokl.* 3(4): 981–983.
- Ivanov, V. V. (1976). *The theory of approximate methods and their application to the numerical solution of singular integral equations*, Leyden : Noordhoff International. ISBN: 9028600361.
- K. Muller, S. Mika, G. R. K. T. & Scholkopf, B. (2002). Ida bechmark repository used in several boosting, kfd and svm papers, *Technical report*. ida.first.gmd.de/~raetsch/data/benchmarks.htm.
- Muller, K., Mika, S., Ratsh, G., Tsuda, K. & Scholkopf, B. (2001). An introduction to kernel-based learning algorithms, *IEEE Trans. on Neural Networks* 12(2): 181–201.
- Phillips, D. Z. (1962). A technique for numerical solution of certain integral equation of the first kind, *J. Assoc. Comput. Mach* 9: 84–96.
- S. Geman, E. B. & Doursat, R. (1992). Neural networks and the bias-variance dilemma, *Neural Computation* 1(4): 1–58.
- Shawe-Taylor, J. & Bartlett, P. L. (1998). Structural risk minimization over data-dependent hierarchies, *IEEE Trans. on Information Theory* 44(5): 1926–1940.

- Tikhonov, A. N. (1963). On solving ill-posed problem and the method of regularization, *Doklady Akademii Nauk USSR* **153**: 501–504.
- Tikhonov, A. N. & Arsenin, V. Y. (1977). *Solution of ill-posed problems*, W. H. Winston, Washington, DC.
- Travassos, X., Vieira, D., Ida, N., Vollaire, C. & Nicolas, A. (2008). Characterization of inclusions in a nonhomogeneous gpr problem by artificial neural networks, *Magnetics, IEEE Transactions on* **44**(6): 1630–1633.
- Vapnik, V. N. (1992). Principles of structural risk minimization for learning theroy, *Advances in Neural Information Processing Systems* **4**: 831–838.
- Vapnik, V. N. (1998). *Statistical Learning Theory*, New York: Wiley.
- Vapnik, V. N. (2001). *The Nature of Statistical Learning Theory (Statistics for Engineering and Information Science)*, second edn, Springer.
- Vasin, V. V. (1970). Relationship of several varitional methods for approximate solutions of ill-posed problems, *Math Notes* **7**: 161–166.
- Vieira, D. A. G., Takahashi, R. H. C., Palade, V., Vasconcelos, J. A. & Caminhos, W. M. (2008). The Q-norm complexity measure and the minimum gradient method: A novel approach to the machine learning structural risk minimization problem, *Neural Networks, IEEE Transactions on* **19**(8): 1415–1430.
- Vieira, D., Travassos, L., Saldanha, R. & Palade, V. (2009). Signal denoising in engineering problems through the minimum gradient method, *Neurocomputing* **72**(10-12): 2270 – 2275.

Statistics Character and Complexity in Nonlinear Systems

Yagang Zhang and Zengping Wang

*Key Laboratory of Power System Protection and Dynamic Security Monitoring and Control under Ministry of Education (North China Electric Power University),
China*

1. Introduction

Learning is the process of constructing a model from complex world. And machine learning is concerned with constructing computer programs that automatically improve with experience. Machine learning draws on concepts and results from many fields, including artificial intelligence, statistics, control theory, cognitive science, information theory, etc. Many successful machine learning applications have also been developed in recent years. Obviously, no matter what we adopt new analytical method or technical means, we must have a distinct recognition of system itself and its complexity, and increase continuously analysis, operation and control level.

In mathematics, nonlinear system represents a system whose behavior is not expressible as a linear function of its descriptors. Our world is inherently nonlinear in nature. Generally speaking, there have difficulties in solving nonlinear equations. Especially the nonlinear system may give rise to some interesting phenomena such as chaos, where simple changes in one part of the system will produce complex effects throughout.

It has been half century since the discovery of inherent randomicity in nonlinear systems (Ulam & Von Neumann, 1947). The study of chaotic symbolic sequences is gradually developing in theory. However, applied research of stochastic chaotic sequences has not been fully carried out, for most of studies focus on controlling or avoiding chaos. Chaos, nevertheless, affords inherent randomicity that can be calculated, which is an important applied domain. The stochastic symbolic sequences bear the following three features. First, computer can generate them iteratively. Second, like false stochastic numbers, they can set up a stochastic sequence simulation (in contradiction, they are based on corresponding symbolic spaces). Third, they can produce numerous symbolic spaces, which is not characteristic of common stochastic numbers. Therefore, the symbolic dynamics (Hao, 1989; Hao, 1991; Hao & Zheng, 1998; Collet & Eckmann, 1980; Alekseev & Yakobson, 1981; Xie, 1993; Xie, 1996; Peng & Luo, 1991; Zhou & Peng, 2000) developed by this means is supposed to be very useful.

Our researches are based on this kind of symbolic sequences, the generic iterative map in n symbolic map (Zhou & Cao, 2003) is:

$$x_{m+1} = \sum_{i=0}^n a_i x_m^i = a_0 x_m^0 + a_1 x_m^1 + \cdots + a_n x_m^n, (n \geq 1, m \geq 0)$$

For random n symbolic sequences, their corresponding symbolic spaces, symbolic expression and kneading sequences are listed in Table 1,

Symbolic Spaces	Symbolic Expression	Kneading Sequences
\sum_2	$L.R$	(L^∞, RL^∞)
\sum_3	$L.M.R$	(R^∞, L^∞) --Kneading plane
\sum_4	$L.M.N.R$	(L^∞, RL^∞) --Kneading space
\vdots	\vdots	\vdots

Table 1. The corresponding symbolic character in symbolic spaces

In this chapter, we will clarify the different kinds of statistic character and complexity in nonlinear systems. This chapter includes two parts, the fist part is about unimodal surjective map and Lorenz type maps nonlinear systems, which are two kinds of typical nonlinear systems. The distributions of frequency, inter-occurrence times, first passage time and visitation density in unimodal surjective map and Lorenz type maps are discussed carefully. These two kinds of nonlinear systems have same distributions, which will also be explained in theory, and the catholicity of the statistic character will be elicited. The second part is about the inherent randomicity in 4-symbolic dynamics. The distribution of frequency, inter-occurrence times and the alignment of two random sequences are amplified in detail. By using transfer probability of Markov chain (MC), we will obtain analytic expressions of generating functions in four probabilities stochastic wander model, which can be applied to all 4-symbolic systems. So, a perfect symbolic platform will be set up for our utilizing statistic character. The 4-symbolic sequences have natural relations with bioinformatics sequences, in the field of application, we hope to afford this kind of symbolic platform which satisfies these stochastic properties and study some properties of DNA sequences, 20 amino acids symbolic sequences of protein structure, and the time series that can be symbolic in finance market et al.

2. The statistic character in Unimodal surjective map

2.1 Symbolic dynamics of Unimodal surjective map

The generic iterative form in Unimodal surjective map:

$$x_{n+1} = F(A, x_n) = 1 - 2x_n^2,$$

x_n is defined on interval $[-1, 1]$.

Let us define an alphabet of 2 numbers, which is corresponding to the likely states of a random discrete nonlinear system, or all the likely outcomes of a random experiment:

$$\Omega = \{0, 1\} = \{"failure", "success"\}$$

The forward sequence constitutes a space (or a set) composed of the generated outcomes:

$$\Omega^N = \{(\xi_0, \xi_1, \xi_2, \dots) : \xi_i \in \Omega, \forall i \in \{0, 1, 2, \dots\}\}$$

These sequences themselves are iteratively generated (Collet & Eckmann, 1980; Peng & Luo, 1991), in fact it's a shift map $\sigma : \Omega^N \rightarrow \Omega^N$, which acting on the sequences by $\sigma(\xi_0, \xi_1, \xi_2, \dots) = (\xi_1, \xi_2, \dots)$. Another definition is μ , which is the product measure (Coelho & Collet, 1994; Coelho, 2000; Peng & Cao, 1996; Billingsley, 1986) on Ω^N generated by the measure $(1-p, p)$ on $\{0, 1\}$, and will be denoted by $(1-p, p)^N$.

2.2 The distribution of frequency

Defining $f : \Omega^N \rightarrow \{0, 1\}$ by $f\{\xi_0, \xi_1, \xi_2, \dots\} = \xi_0$, it is coarse graining in theory, one can get:

$$X_i(\xi) = f(\sigma^i \xi), \text{ (for } i = 0, 1, 2, \dots\text{)},$$

which are sequences of independent and identically distributed(i.i.d.) random variables defined on the probability space (Ω^N, μ) (all the following discussions are based on the random variables), that is, the random variables represented by $\xi_0, \xi_1, \xi_2, \dots$ are i.i.d., and $\xi_i (i = 0, 1, 2, \dots)$ is based on Ω ,

$$\begin{aligned} Y_n &= X_0 + X_1 + \dots + X_{n-1} \\ &= \sum_{i=0}^{n-1} f(\sigma^i x) \\ &= \xi_0 + \xi_1 + \dots + \xi_{n-1} \end{aligned}$$

The stochastic symbolic sequences in Unimodal surjective map satisfy Binomial distribution:

$$\mu\{\xi \in \Omega^N : Y_n(x) = k\} = C_n^k p^k (1-p)^{n-k} \quad (1)$$

2.3 The inter-occurrence times in Unimodal surjective map

Now let us make a further study a given word's occurrence times in an independent repeated experiment, such as *success* in the alphabet of 2 numbers. Given outcomes of a random sequence

$$\xi = (\xi_0, \xi_1, \xi_2, \dots) \in \Omega^N,$$

we are mainly interested in n such that $\xi_n = 1$, let

$$\tau(\xi) = \tau_1(\xi) = \inf\{n \geq 0 : \xi_n = 1\},$$

and accordingly, for $j \geq 2$,

$$\tau_j(\xi) = \inf\{n > \tau_{j-1}(\xi) : \xi_n = 1\},$$

then for all $k \geq 0$, the result is, for fixed $k > 0$, and all $k_1 < \dots < k_{j-1}$,

$$\begin{aligned} P(\tau_j - \tau_{j-1} = k \mid \tau_{j-1} = k_{j-1}, \dots, \tau_1 = k_1) \\ = P(\tau_j - \tau_{j-1} = k) \\ = p(1-p)^{k-1} \end{aligned} \tag{2}$$

the inter-occurrence times $1 + \tau_1, \tau_2 - \tau_1, \tau_3 - \tau_2, \dots$ are i.i.d. with parameters p .

2.4 The first passage time in Unimodal surjective map

Using this method similar to study the distribution of first passage time T_y of one-dimensional simple random walk in stochastic processes, one gets τ_j satisfies Negative Binomial distribution $BN(j, p)$:

$$\begin{aligned} P(\tau_j = k) &= C_{k-1}^{r-1} p^r q^{k-r} \\ (k = r, r+1, r+2, \dots, 0 < p < 1, q = 1-p) \end{aligned} \tag{3}$$

3. The statistic character in Lorenz maps

3.1 Symbolic dynamics of Lorenz maps

Lorenz equation:

$$\begin{cases} \dot{x} = \sigma(y - x), \\ \dot{y} = (r - z)x - y \\ \dot{z} = xy - bz \end{cases}$$

On the Poincaré section, some geometrical structure of Lorenz flow may be reduced to a one-dimensional Lorenz map $f : [-\mu, \nu] \rightarrow [-\mu, \nu]$, ($\mu, \nu > 0, \lambda > 1$)

$$f(x) = \begin{cases} f_L(x) = \nu - \alpha |x|^\lambda + h.o.t, & x \leq 0 \\ f_R(x) = -\mu + \beta x^\lambda + h.o.t, & x > 0 \end{cases}$$

Where λ is a constant greater than 1, "h.o.t" represents high-level term. Both of the branches f_L and f_R are monotone increasing. In order to get iterative sequences in the part of chaos, the Lorenz map used in this research is:

$$f(x) = \begin{cases} f_L = 1 - 2|x|^2, & x \leq 0 \\ f_R = -1 + 2x^2, & x > 0 \end{cases} \quad (4)$$

The symbolic dynamics of Lorenz maps is also simple (Peng & Du, 1999). Following the kneading theory, the address $A(x)$ of any point x on the interval $[-1, 1]$ reads

$$A(x) = \begin{cases} R, & x \in [-1, 0) \\ L, & x \in [0, 1] \end{cases}$$

$x = 0$ is the turning (discontinuous) point, and one can define C and D as

$$\begin{aligned} C &= \lim_{x \rightarrow 0^-} f_L(x), \\ D &= \lim_{x \rightarrow 0^+} f_R(x). \end{aligned}$$

Two infinite or finite symbolic sequences starting from C and D are kneading sequences which can be ordered lexicographically by $L \prec C, D \prec R$. For two kneading sequences, $\gamma_1 \gamma_2 \dots \gamma_i \gamma_{i+1} \dots$ and $\eta_1 \eta_2 \dots \eta_i \eta_{i+1} \dots$, with maximal common leading part:

$$\gamma_1 \gamma_2 \dots \gamma_i = \eta_1 \eta_2 \dots \eta_i,$$

one has ,

$$\gamma_1 \gamma_2 \dots \gamma_i \gamma_{i+1} \dots \prec \eta_1 \eta_2 \dots \eta_i \eta_{i+1} \dots$$

if and only if $\gamma_{i+1} \prec \eta_{i+1}$.

The shift operator φ is defined as ,

$$\varphi^k(\xi) = \xi_{k+1}\xi_{k+2} \dots \text{ for } \xi = \xi_1\xi_2 \dots \xi_k\xi_{k+1} \dots$$

For any two sequences ,

$$\xi = \xi_1\xi_2 \dots \xi_i\xi_{i+1} \dots \text{ and } \zeta = \zeta_1\zeta_2 \dots \zeta_j\zeta_{j+1} \dots,$$

$\xi_i, \zeta_j \in \{R, L\}$, if $\varphi^k(\xi) \prec \xi$ and $\zeta \prec \varphi^k(\zeta)$, for all $K \in \mathbb{Z}_+$,then ξ is called maximal, ζ minimal, and $S = (\xi, \zeta)$ is an extremal pair. Let the integers k_L and k_R be the order coordinates of a letter in the sequence such that $\varphi^{k_L-1}(\xi) = L \dots$, and $\varphi^{k_R-1}(\zeta) = R \dots$, the set k_L and k_R describe successive sequences of L or R . Then, if the pair S further satisfies the following condition:

$$\begin{aligned} \varphi^{k_L}(\xi) &\prec K^1, \varphi^{k_R}(\zeta) \succ K^2, \{k_L\} \cup \{k_R\} = \{k\} \in \mathbb{Z}_+, \\ \varphi^{k'_L}(\zeta) &\prec K^1, \varphi^{k'_R}(\zeta) \succ K^2, \{k'_L\} \cup \{k'_R\} = \{k'\} \in \mathbb{Z}_+. \end{aligned}$$

S is admissible with respect to the kneading sequences K^1 and K^2 . All the admissible pairs form an admissible set K and fill up the whole kneading parameter plane of nonlinear systems of two letters.

3.2 The distribution of frequency

See expression (1).

3.3 The inter-occurrence times in Lorenz map

See expression (2).

3.4 The first passage time in Lorenz map

See expression (3).

4. Visitation density function of Unimodal surjective maps and Lorenz map

The orbital points' distribution of the Unimodal surjective maps and Lorenz map is,

$$\rho(x) = \frac{1}{\pi\sqrt{1-x^2}} \tag{5}$$

The concrete resolvent is using Frobenius-Perron operator (Lasota & Mackey, 1985; Yorke & Li, 1975; Ding & Li, 1991; Li, 1976). The general form of resolve visitation density problem by F-P operator P is,

$$Pf(x) = \frac{d}{dx} \int_{S^{-1}(\Delta)} f(u) du ,$$

here, $S = S(x)$ is a given map, Δ is an interval, $f(x)$ is a density function. In fact, it is an iterative process, the initial state is

$$\int_{\Delta} f_1(u) du = \int_{S^{-1}(\Delta)} f_0(u) du .$$

$f_0(x)$ is an arbitrary initial density and $f_1(x)$ is a new density transformed by map $S(x)$, that is,

$$f_1 = Pf_0 ,$$

until,

$$f_*(x) = P^n f(x) \quad \text{as } n \rightarrow \infty .$$

Of course,

$$Pf_*(x) = f_*(x) ,$$

the unique limiting density is just the ultimate visitation density function.

It is mainly in numerical value meaning that getting visitation density functions of higher order maps, if the invariable density does exist. Figure 1 is the U-shaped probability density based on iterates, corresponding analytic form is just expression (5),

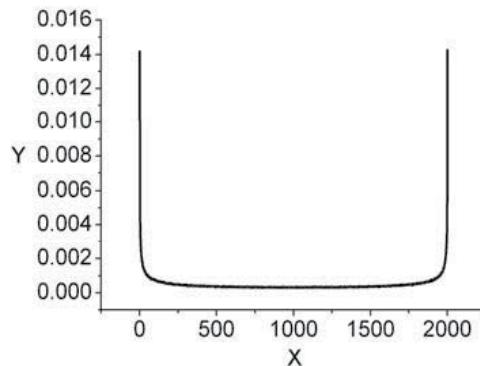


Fig. 1. The visitation density of Unimodal surjective map and Lorenz map based on 1000000 iterates, the interval $[-1,1]$ is divided into 2000 subintervals. X coordinate axis is corresponding interval, Y coordinate axis is the output proportion of each interval.

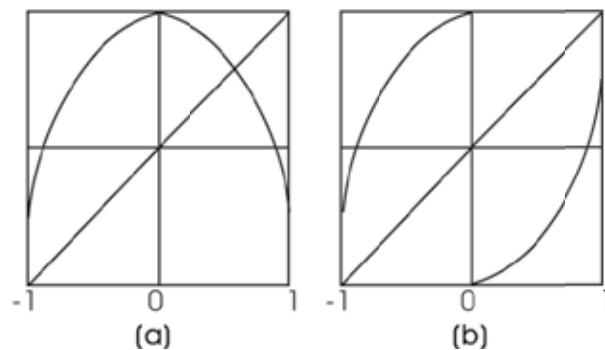


Fig. 2. Unimodal surjective map (a) and Lorenz map (b)

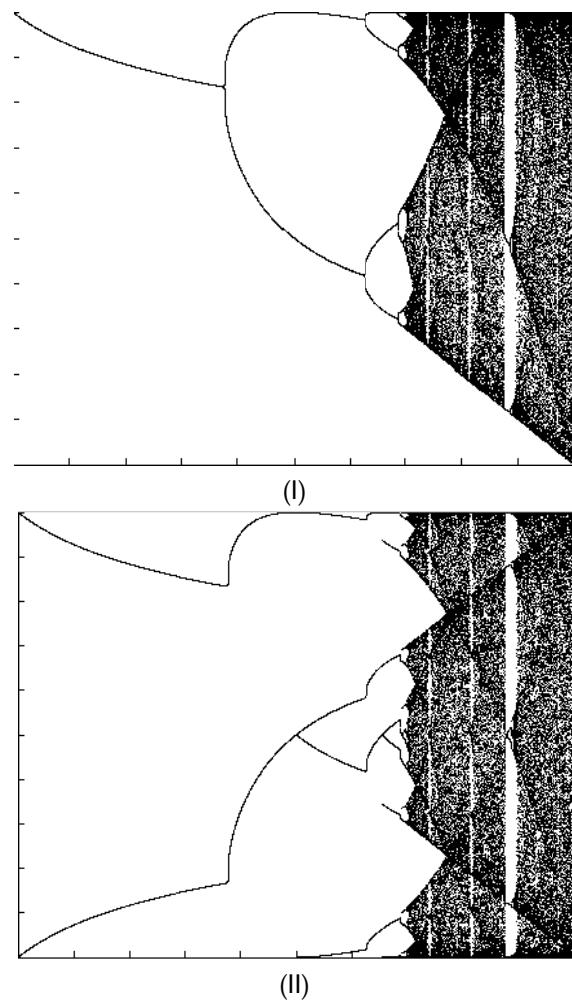


Fig. 3. The bifurcation diagrams of Unimodal surjective map (I) and Lorenz map (II)

5. The comparability of statistic character in the Unimodal map and Lorenz map

The former statistic character in Lorenz map is similar entirely to that in Unimodal surjective map. This kind of comparability is determined by the relationship of Unimodal surjective map and Lorenz map. (See Figure 2)

The iterative form of Lorenz map is (4), and Unimodal surjective map is $y = 1 - 2x^2$.

One can find this characteristic by Figure 2,

$$\begin{cases} f_a = -f_b, & x \geq 0 \\ f_a = f_b, & x < 0 \end{cases}$$

A n -periods orbit of f_a corresponds to a couple of n -periods orbits of f_b . Both of them have the same topological entropy and marker behavior. The fixed point of f_a exhibits two-periods behavior of f_b , which can be found clearly by contrasting their bifurcation diagrams. (See Figure 3)

Compared the right branch of Lorenz map and Unimodal surjective map, the Lorenz map is only overturned by x coordinate axis. As these results reveal that this kind of overturn does not influence statistical properties of random sequences. Compared with Unimodal map, Lorenz map belongs to a more complex category, which presents more abundant dynamics actions. But as above study, these statistical results present regulation as a whole. These are randomicity in deterministic systems.

6. The stochastic properties in 4-letters maps

6.1 The distribution of frequency

Let us define an alphabet of four numbers, which is corresponding to the likely states of a random discrete dynamical system, or all the likely outcomes of a random experiment:

$$\begin{aligned} \Omega &= \{0, 1, 2, 3\} = \{L, M, N, R\} = \{A, G, C, T\} \\ &= \{"Spring", "Summer", "Autumn", "Winter"\} \end{aligned}$$

The forward sequence constitutes a space (or a set) composed of the generated outcomes:

$$\Omega^N = \{(\xi_0, \xi_1, \xi_2, \dots) : \xi_i \in \Omega, \forall i \in \{0, 1, 2, \dots\}\}$$

These sequences themselves are iteratively generated, in fact it's a shift map $\sigma : \Omega^N \rightarrow \Omega^N$, which acting on the sequences by $\sigma(\xi_0, \xi_1, \xi_2, \dots) = (\xi_1, \xi_2, \dots)$.

Another definition is μ , which is the product measure[6] on Ω^N generated by the measure

(p_1, p_2, p_3, p_4) on $\{0, 1, 2, 3\}$ ($p_1 + p_2 + p_3 + p_4 = 1$), and will be denoted by $(p_1, p_2, p_3, p_4)^N$. Defining $\varphi: \Omega^N \rightarrow \{0, 1, 2, 3\}$ by $\varphi\{\xi_0, \xi_1, \xi_2, \dots\} = \xi_0$, it is also coarse graining in theory, one can get:

$$X_i(\xi) = \varphi(\sigma^i \xi), \text{ (for } i = 0, 1, 2, \dots)$$

Which are sequences of independent and identically distributed (i.i.d.) random variables defined on the probability space (Ω^N, μ) (all the following discussions are based on the random variables), that is, the random variables represented by $\xi_0, \xi_1, \xi_2, \dots$ are i.i.d., and $\xi_i (i = 0, 1, 2, \dots)$ is based on Ω .

The numeric examinations reveal that the stochastic symbolic sequences in 4-letters maps satisfy multinomial distribution:

$$T(N_1 = n_1, N_2 = n_2, N_3 = n_3, N_4 = n_4) = \frac{n!}{n_1! n_2! n_3! n_4!} p_1^{n_1} p_2^{n_2} p_3^{n_3} p_4^{n_4}$$

$$\left(\sum_{i=1}^4 p_i = 1, \quad \sum_{i=1}^4 n_i = n \right)$$

The theoretic foundation of these results is that the topological entropy (Shi et al., 1996; Zhang et al., 1996; Cao et al., 1995; Chen et al. 1995; Peng et al., 1994; Chen & Zhou, 2003; Chen & Zhou, 2003; Liang & Jiang, 2002) in n letters surjective maps is $\ln(n)$, which is a deduction from chaotic symbolic sequences' Bernoulli property.

6.2 The inter-occurrence times of 4-letters maps

Now let us make a further study a given word's occurrence times in an independent repeated experiment, such as "R", "T" or "Winter" in the alphabet of four numbers. Given outcomes of a random sequence

$$\xi = (\xi_0, \xi_1, \xi_2, \dots) \in \Omega^N,$$

we are mainly interested in n such that $\xi_n = T$, let

$$\tau^\lambda(\xi) = \tau_1^\lambda(\xi) = \inf\{n \geq 0 : \xi_n = T\}, \quad \lambda \in \Omega$$

and accordingly, for $j \geq 2$, λ corresponds to T ,

$$\tau_j^\lambda(\xi) = \inf\{n > \tau_{j-1}^\lambda(\xi) : \xi_n = T\},$$

then for all $k \geq 0$, the result is, for fixed $k > 0$ and all $k_1 < \dots < k_{j-1}$,

$$\begin{aligned} & P(\tau_j^\lambda - \tau_{j-1}^\lambda = k \mid \tau_{j-1}^\lambda = k_{j-1}, \dots, \tau_1^\lambda = k_1) \\ &= P(\tau_j^\lambda - \tau_{j-1}^\lambda = k) \\ &= p_1^a p_2^b p_3^c p_4 \end{aligned} \quad (6)$$

$$\left(\sum_{i=1}^4 p_i = 1, a, b, c \in \{0, 1, 2, \dots\}, a+b+c = k-1 \right) \quad (7)$$

the inter-occurrence times $1 + \tau_1^\lambda, \tau_2^\lambda - \tau_1^\lambda, \tau_3^\lambda - \tau_2^\lambda, \dots$ are i.i.d. with parameters p_1, p_2, p_3, p_4 .

Correspondingly, for $j \geq 2$,

If λ corresponds to A , and

$$\tau_j^\lambda(\xi) = \inf\{n > \tau_{j-1}^\lambda(\xi) : \xi_n = A\}$$

Then

$$P(\tau_j^\lambda - \tau_{j-1}^\lambda = k) = p_2^a p_3^b p_4^c p_1 \quad (8)$$

If λ corresponds to G , and

$$\tau_j^\lambda(\xi) = \inf\{n > \tau_{j-1}^\lambda(\xi) : \xi_n = G\}$$

Then

$$P(\tau_j^\lambda - \tau_{j-1}^\lambda = k) = p_1^a p_3^b p_4^c p_2 \quad (9)$$

If λ corresponds to C , and

$$\tau_j^\lambda(\xi) = \inf\{n > \tau_{j-1}^\lambda(\xi) : \xi_n = C\}$$

then

$$P(\tau_j^\lambda - \tau_{j-1}^\lambda = k) = p_1^a p_2^b p_4^c p_3 \quad (10)$$

the conditions of expression (8)–(10) are also expression (7).

6.3 Exponential distribution of 4-letters maps

Suppose there are two random sequences of outcomes corresponding to the repetition of an experiment with four likely results. Let $\xi = (\xi_0, \xi_1, \xi_2, \dots)$ and $\zeta = (\zeta_0, \zeta_1, \zeta_2, \dots)$ denote the sequence of outcomes (ξ independent of ζ). There is an alignment at time n if $\xi_n = \zeta_n$. The alignment at time n as a success and no alignment is failure. Then note that, for all $n > 0$, $0 < p < 1$ and $q = 1 - p$,

$$P(\xi_n = \zeta_n) = P(\xi_0 = \zeta_0) = p$$

Now consider having a successive sequence of alignments. Define $\phi : \Omega^N * \Omega^N \rightarrow N$ by $\phi(\xi, \zeta) = k$ if $\xi_0 = \zeta_0, \xi_1 = \zeta_1, \dots, \xi_{k-1} = \zeta_{k-1}, \xi_k \neq \zeta_k$, namely,

$$\phi(\xi, \zeta) = k, \text{ if } \xi_0 = \zeta_0, \xi_1 = \zeta_1, \dots, \xi_{k-1} = \zeta_{k-1}, \xi_k \neq \zeta_k,$$

introduced shift arithmetic operators σ ,

$$\begin{aligned}\sigma(\xi) &= (\xi_1, \xi_2, \xi_3, \dots), \\ \sigma(\zeta) &= (\zeta_1, \zeta_2, \zeta_3, \dots),\end{aligned}$$

then,

for $i = 1, 2, 3, \dots$,

$$\begin{aligned}\phi(\sigma(\xi), \sigma(\zeta)) &= k, \text{ if } \xi_1 = \zeta_1, \dots, \xi_k = \zeta_k, \xi_{k+1} \neq \zeta_{k+1}; \\ \phi(\sigma^2(\xi), \sigma^2(\zeta)) &= k, \text{ if } \xi_2 = \zeta_2, \dots, \xi_{k+1} = \zeta_{k+1}, \xi_{k+2} \neq \zeta_{k+2}; \\ &\vdots && \vdots \\ \phi(\sigma^i(\xi), \sigma^i(\zeta)) &= k, \text{ if } \xi_i = \zeta_i, \dots, \xi_{k-1+i} = \zeta_{k-1+i}, \xi_{k+i} \neq \zeta_{k+i}\end{aligned}$$

in fact, alignment from i term is just keeping the former k terms success and the $k+i$ term failure.

Denote random variables

$$Z_n = \phi(\sigma^n \xi, \sigma^n \zeta), \quad \gamma_k = \inf\{n \geq 0 : Z_n \geq k\}.$$

So,

$$\begin{aligned}\therefore \sigma^{n-1}(\xi) &= (\xi_{n-1}, \xi_n, \xi_{n+1}, \dots), \\ \sigma^{n-1}(\zeta) &= (\zeta_{n-1}, \zeta_n, \zeta_{n+1}, \dots)\end{aligned}$$

and

$$\xi_{n-1} = \zeta_{n-1}, \xi_n = \zeta_n, \dots, \xi_{n+k-2} = \zeta_{n+k-2}, \xi_{n+k-1} \neq \zeta_{n+k-1},$$

therefore,

$$Z_{n-1} = \phi(\sigma^{n-1}\xi, \sigma^{n-1}\zeta) = k.$$

The same way,

$$\begin{aligned} \because \sigma^n(\xi) &= (\xi_n, \xi_{n+1}, \xi_{n+2}, \dots), \\ \sigma^n(\zeta) &= (\zeta_n, \zeta_{n+1}, \zeta_{n+2}, \dots) \end{aligned}$$

and

$$\xi_n = \zeta_n, \xi_{n+1} = \zeta_{n+1}, \dots, \xi_{n+k-2} = \zeta_{n+k-2}, \xi_{n+k-1} \neq \zeta_{n+k-1}$$

therefore,

$$Z_n = \phi(\sigma^n\xi, \sigma^n\zeta) = k - 1$$

so,

$$P(Z_n = k - 1 | Z_{n-1} = k) = 1, \text{ if } k \geq 1$$

for every $t > 0$, one gets the asymptotic exponential distribution of γ_k :

$$\lim_{k \rightarrow \infty} P(\gamma_k > t E(\gamma_k) | Z_0 = 0) = e^{-t},$$

$E(\gamma_k)$ represents mathematical expectation of γ_k , t is a time coordinate. Furthermore, if

$$Z_n = \phi(\sigma^n\xi, \sigma^n\zeta) = k - 1$$

and $\xi_{n+k-1} = A$, then,

$$\lim_{k \rightarrow \infty} P(\gamma_k > t_A E(\gamma_k)) = e^{-t_A}$$

accordingly, one also gets,

$$\lim_{k \rightarrow \infty} P(\alpha_k > t_G E(\alpha_k)) = e^{-t_G},$$

$$\lim_{k \rightarrow \infty} P(\beta_k > t_C E(\beta_k)) = e^{-t_C},$$

$$\lim_{k \rightarrow \infty} P(\omega_k > t_T E(\omega_k)) = e^{-t_T}.$$

$\alpha_k, \beta_k, \omega_k$ correspond to γ_k when $\xi_{n+k-1} = G, \xi_{n+k-1} = C, \xi_{n+k-1} = T$.

We also get,

$$\begin{aligned} \lim_{k \rightarrow \infty} P(\gamma_k > t_A E(\gamma_k), \alpha_k > t_G E(\alpha_k), \beta_k > t_C E(\beta_k), \omega_k > t_T E(\omega_k)) \\ = e^{-(t_A + t_G + t_C + t_T)} \end{aligned}$$

Fig. 4 represents the alignment of two random sequences.

6.4 Transfer probability of Markov chain (MC) in 4-letters maps

We choose one of these transfer models (Figure. 5), such as Figure. 6.

If n fixation, the transfer probability of (i, j) is defined $p_{ij}^{(n)}$. The generating function is,

$$G^n(z) = (qz^{-1} + r + pz + vz^2)^n, \quad (11)$$

which is determined by Figure. 3.

$$G^n(z) = \sum_{k \in R} p_{1k}^{(n)} z^k = \sum_{j-i \in R} p_{ij}^{(n)} z^{j-i}. \quad (12)$$

Hereinto, $p_{1k}^{(n)} = P(X_n = k | X_0 = 1)$, $p_{ij}^{(n)} = P(X_n = j | X_0 = i)$, $k = j - i$,

$R = (-n, \dots, 0, \dots, 2n)$, R represents all coordinates a particle could attain during n steps transfer if n fixation. By using (11) and (12), one gets:

under restrictive condition of $\delta_{(2n_3+n_2-n_0),k}$:

$$p_{1k}^{(n)} = \frac{n!}{n_0! n_1! n_2! n_3!} q^{n_0} r^{n_1} p^{n_2} v^{n_3},$$

$$(\delta : k = 2n_3 + n_2 - n_0, \quad n = n_0 + n_1 + n_2 + n_3)$$

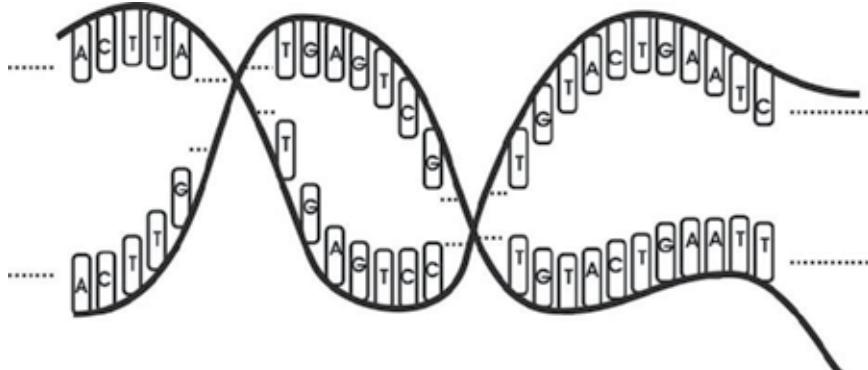


Fig. 4. The alignment of two random sequences

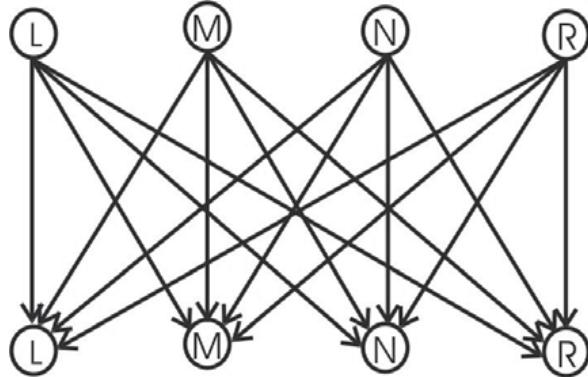


Fig. 5. Four probabilities stochastic wander model

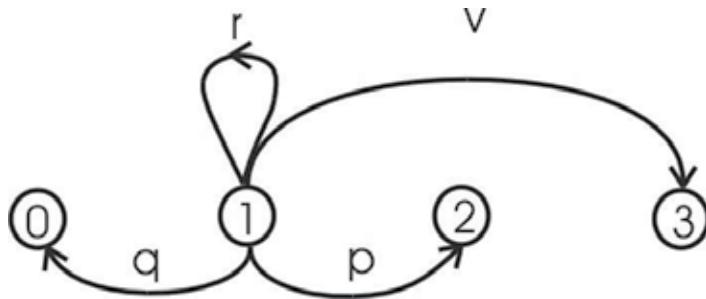


Fig. 6. One of these transfer models

Let,

$$n - n_1 + n_3 - k = 2l, \quad n_0 = l, \quad n_1 = m, \quad n_2 = l + k - 2\lambda, \quad n_3 = \lambda,$$

then,

$$p_{1k}^{(n)} = \frac{(2l+m+k-\lambda)!}{l!m!(l+k-2\lambda)!\lambda!} q^l r^m p^{(l+k-2\lambda)} v^\lambda, \quad k \in [-n, 2n].$$

The generating function $P_{1k}(s)$ if k fixation. During different n steps transfer, all the probabilities $p_{1k}^{(n)}$ are introduced into a generating function $P_{1k}(s)$, which is defined as

$$\begin{aligned} P_{1k}(s) &= \sum_{n=0}^{\infty} p_{1k}^{(n)} s^n \\ &= (ps)^k \sum_{\lambda=0}^{\infty} \sum_{l=0}^{\infty} C_{2l+k-\lambda}^l C_{l+k-\lambda}^{\lambda} (qs)^l (ps)^{(l-2\lambda)} (vs)^{\lambda} (1-rs)^{-(2l+k-\lambda+1)} \\ &= \left(\frac{ps}{1-rs}\right)^k \frac{1}{1-rs} \sum_{\lambda=0}^{\infty} \sum_{l=0}^{\infty} C_{2l+k-\lambda}^l C_{l+k-\lambda}^{\lambda} \left(\frac{qs}{1-rs}\right)^l \left(\frac{ps}{1-rs}\right)^{(l-2\lambda)} \left(\frac{vs}{1-rs}\right)^{\lambda} \end{aligned}$$

let,

$$x_0 = \frac{qs}{1-rs}, \quad x_2 = \frac{ps}{1-rs}, \quad x_3 = \frac{vs}{1-rs};$$

$$x = x_0 x_2 = \frac{qps^2}{(1-rs)^2}, \quad y = x_0^2 x_3 = \frac{q^2 vs^3}{(1-rs)^3};$$

then,

$$P_{1k}(s) = \left(\frac{qs}{1-rs}\right)^{-k} \frac{1}{1-rs} \sum_{\lambda=0}^{\infty} \frac{y^\lambda}{\lambda!} \frac{d^\lambda}{dx^\lambda} \left(\sum_{l=0}^{\infty} C_{2l+k+\lambda} x^{l+k} \right).$$

So, for $k = 1$,

$$P_{11}(s) = \frac{1}{1-rs} \sum_{\lambda=0}^{\infty} \frac{y^\lambda}{\lambda!} \frac{d^\lambda}{dx^\lambda} \left(\sum_{l=0}^{\infty} C_{2l+1} x^l \right)$$

for $k = 2$,

$$P_{12}(s) = \frac{1}{qs} \sum_{\lambda=0}^{\infty} \frac{y^\lambda}{\lambda!} \frac{d^\lambda}{dx^\lambda} \left(\sum_{l=0}^{\infty} C_{2l+2+\lambda} x^{l+1} \right)$$

for $k = 3$,

$$P_{13}(s) = \frac{1-rs}{q^2 s^2} \sum_{\lambda=0}^{\infty} \frac{y^\lambda}{\lambda!} \frac{d^\lambda}{dx^\lambda} \left(\sum_{l=0}^{\infty} C_{2l+3+\lambda} x^{l+2} \right)$$

7. Conclusion and discussion

The statistic character and complexity in nonlinear systems have been clarified in this chapter. These stochastic symbolic sequences bear three characters. In two kinds of typical nonlinear systems-unimodal surjective map and Lorenz type maps nonlinear systems, the distributions of frequency, inter-occurrence times, first passage time and visitation density in unimodal surjective map and Lorenz type maps are discussed carefully. These two kinds of nonlinear systems have same distributions, which have also been explained in theory, and the catholicity of the statistic character has been elicited. In the 4-symbolic dynamics, the distribution of frequency, inter-occurrence times and the alignment of two random sequences have been amplified in detail. By using transfer probability of Markov chain (MC), we have obtained analytic expressions of generating functions in four probabilities stochastic wander model, which can be applied to all 4-symbolic systems. So, a perfect symbolic platform has been set up for our utilizing statistic character, in fact, it is a stochastic signal platform of symbolic simulation. The 4-symbolic sequences have natural relations with bioinformatics sequences, in the field of application, we hope to afford a symbolic platform which satisfies these statistic character and study some properties of DNA sequences (Hao, 2000; Hao et al., 2000; Bershadskii, 2001; Grimm & Rupprecht, 1997; Allegrini et al., 1996; Natalia & Avy, 2005; Elena et al., 2005), 20 amino acids symbolic sequences of protein structure, and the time series that can be symbolic in finance market et al, which are part of our future work. The symbolic platform provides a set of effective

methods to approach problems of this kind. The establishment of this symbolic platform will open up a vast vista.

8. Acknowledgements

The first author would like to thank Prof. Shou-Li Peng for numerous discussions and valuable comments. This research was supported partly by National Natural Science Foundation of China (50837002) and the Science Foundation for the Youth Scholars of NCEPU.

9. References

- Alekseev, V.M. & Yakobson, M.V. (1981). Symbolic dynamics and hyperbolic dynamics systems. *Physics Reports*, Vol. 75, 287-325, ISSN: 0370-1573
- Allegrini, P., Grigolini, P. & West B.J. (1996). A dynamical approach to DNA sequences. *Physics Letters A*, Vol. 211, 217-222, ISSN: 0375-9601
- Bershadskii, A. (2001). Multifractal and probabilistic properties of DNA sequences. *Physics Letters A*, Vol. 284, 136-140, ISSN: 0375-9601
- Billingsley, P. (1986). *Probability and Measure*, John Wiley & Sons, ISBN: 0-471-80478-9, New York
- Cao, K.F., Chen, Z.X. & Peng, S.L. (1995). Global metric regularity of the devil's staircase of topological entropy. *Physical Review E*, Vol. 51, 1989-1995, ISSN: 1539-3755
- Chen, Z.-X., Cao, K.-F. & Peng, S.-L. (1995). Symbolic dynamics analysis of topological entropy and its multifractal structure. *Physical Review E*, Vol. 51, 1983-1988, ISSN: 1539-3755
- Chen, Z.X. & Zhou, Z. (2003). Entropy invariants: I . The universal order relation of order-preserving star products. *Chaos, Solitons & Fractals*, Vol. 15, 713-727, ISSN: 0960-0779
- Chen, Z.X. & Zhou, Z. (2003). Entropy invariants: II . The block structure of Stefan matrices. *Chaos, Solitons & Fractals*, Vol. 15, 729-742, ISSN: 0960-0779
- Coelho, Z. (2000). On discrete stochastic processes generated by deterministic sequences and multiplication machines. *Indagationes Mathematicae*, Vol. 11, 359–378, ISSN: 0019-3577
- Coelho, Z. & Collet, P. (1994). Asymptotic limit law for the close approach of two trajectories in expanding maps of the circle. *Journal of Probability Theory and related fields*, Vol. 99, 237-250, ISSN: 0178-8051
- Collet, P. & Eckmann, J.P. (1980). *Iterated Maps on the Interval as Dynamical Systems*, Birkhauser, ISBN-13: 978-0817-63-026-3, Boston
- Ding, J. & Li, T.Y. (1991). Markov finite approximation of Frobenius-Perron operator. *Nonlinear Analysis: Theory, Methods & Applications*, Vol. 17, 759-772, ISSN: 0362-546X
- Elena, I., Andrei, P., Sergey V. & Yegor S. (2005). Mapping long-range chromatin organization within the chicken-globin gene domain using oligonucleotide DNA arrays. *Genomics*, Vol. 85, 143-151, ISSN: 0888-7543
- Grimm, H. & Rupprecht, A. (1997). Low frequency dynamics of DNA. *Physica B*, Vol. 234-236, 183-187, ISSN: 0921-4526
- Hao, B.L. (1989). *Elementary Symbolic Dynamics and Chaos in Dissipative Systems*, World Scientific, ISBN: 978-9971-50-682-7, Singapore

- Hao, B.L. (1991). Symbolic Dynamics and characterization of complexity. *Physica D*, Vol. 51, 161-176, ISSN: 0167-2789
- Hao, B.L. (2000). Fractals from genomes-exact solutions of a biology-inspired problem. *Physica A*, Vol. 282, 225-246, ISSN: 0378-4371
- Hao, B.L., Lee, H.C. & Zhang, S.Y. (2000). Fractals related to long DNA sequences and complete genomes. *Chaos, Solitons & Fractals*, Vol. 11, 825-836, ISSN: 0960-0779
- Hao, B.L. & Zheng, W.M. (1998). *Symbolic Dynamics and Chaos. Directions in Chaos* Vol. 7, World Scientific, ISBN: 978-9971-50-698-8, Singapore
- Lasota, A. & Mackey, M.C. (1985). *Probabilistic properties of deterministic systems*, Cambridge University Press, ISBN: 0-521-30248-X, Cambridge
- Li, T.Y. (1976). Finite approximation for the Frobenius-Perron operator, a solution to Ulam's conjecture. *Journal of Approximation Theory*, Vol. 17, 177-186, ISSN: 0021-9045
- Liang, X. & Jiang, J.F. (2002). On the topological entropy, nonwandering set and chaos of monotone and competitive dynamical systems. *Chaos, Solitons & Fractals*, Vol. 14, 689-696, ISSN: 0960-0779
- Natalia, L. & Avy, S. (2005). Nonlinear waves in double-stranded DNA. *Bulletin of Mathematical Biology*, Vol. 67, 701-718, ISSN: 0092-8240
- Peng, S.L. & Cao, K.F. (1996). Global scaling behaviors and chaotic measure characterized by the convergent rates of period-p-tupling bifurcations. *Physical Review E*, Vol. 54, 3211-3220, ISSN: 1539-3755
- Peng, S.L., Cao, K.F. & Chen, Z.X. (1994). Devil's staircase of topological entropy and global metric regularity. *Physics Letters A*, Vol. 193, 437-443, ISSN: 0375-9601
- Peng, S.L. & Du, L.M. (1999). Dual star products and symbolic dynamics of Lorenz maps with the same entropy. *Physics Letters A*, Vol. 261, 63-73, ISSN: 0375-9601
- Peng, S.L. & Luo, L.S. (1991). The ordering of critical periodic points in coordinate space by symbolic dynamics. *Physics Letters A*, Vol. 153, 345-352, ISSN: 0375-9601
- Shi, J.X., Cao, K.F., Guo, T.L. & Peng, S.L. (1996). Metric universality for the devil's staircase of topological entropy. *Physics Letters A*, Vol. 211, 25-28, ISSN: 0375-9601
- Ulam, S.M. & Von Neumann, J. (1947) . On combination of stochastic and deterministic processes. *Bulletin of the American Mathematical Society*, Vol. 53, 1120, ISSN: 0273-0979
- Xie, H.M. (1993). On formal languages of one-dimensional dynamics systems. *Nonlinearity*, Vol. 6, 997-1007, ISSN: 0951-7715
- Xie, H.M. (1996). Grammatical Complexity and one-dimensional dynamics systems. *Directions in Chaos* Vol. 6, World Scientific, ISBN 978-9810-22-398-4, Singapore
- Yorke, J. & Li, T.Y. (1975). Period three implies chaos. *The American Mathematical Monthly*, Vol. 82, 985-992, ISSN: 0002-9890
- Zhou, Z. & Cao, K.F. (2003). An effective numerical method of the word-lifting technique in one-dimensional multimodal maps. *Physics Letters A*, Vol. 310, 52-59, ISSN: 0375-9601
- Zhou, Z. & Peng, S.L. (2000). Cyclic star products and universalities in symbolic dynamics of trimodal maps. *Physica D*, Vol. 140, 213-226, ISSN: 0167-2789
- Zhang, X.S., Liu, X.D., Kwek, K.H. & Peng, S.L. (1996). Disorder versus order: Global multifractal relationship between topological entropies and universal convergence rates. *Physics Letters A*, Vol. 211, 148-154, ISSN: 0375-9601

Adaptive Basis Function Construction: An Approach for Adaptive Building of Sparse Polynomial Regression Models

Gints Jekabsons
*Riga Technical University
Latvia*

1. Introduction

The task of learning useful models from available data is common in virtually all fields of science, engineering, and finance. The goal of the learning task is to estimate unknown (input, output) dependency (or model) from training data (consisting of a finite number of samples) with good prediction (generalization) capabilities for future (test) data (Cherkassky & Mulier, 2007; Hastie et al., 2003). One of the specific learning tasks is regression – estimating an unknown real-valued function. The process of regression model learning is also called regression modelling or regression model building.

Many practical regression modelling methods use basis function representation – these are also called dictionary methods (Friedman, 1994; Cherkassky & Mulier, 2007; Hastie et al., 2003), where a particular type of chosen basis functions constitutes a “dictionary”. Further distinction is then made between non-adaptive methods and adaptive (also called flexible) methods.

The most widely used form of basis function expansions is polynomial of a fixed degree. If a model always includes a fixed (predetermined) set of basis functions (i.e. they are not adapted to training data), the modelling method is considered non-adaptive (Cherkassky & Mulier, 2007; Hastie et al., 2003). Using adaptive modelling methods however the basis functions themselves are adapted to data (by employing some kind of search mechanism). This includes methods where the restriction of fixed polynomial degree is removed and the model’s degree now becomes another parameter to fit. Adaptive methods use a very wide dictionary of candidate basis functions and can, in principle, approximate any continuous function with a pre-specified accuracy. This is also known as the universal approximation property (Kolmogorov & Fomin, 1975, Cherkassky & Mulier, 2007).

However, in polynomial regression the increase in the model’s degree leads to exponential growth of the number of basis functions in the model (Cherkassky & Mulier, 2007; Hastie et al., 2003). With finite training data, the number of basis functions along with the number of model’s parameters (coefficients) quickly exceeds the number of data samples, making model’s parameter estimation impossible. Additionally the model should not be overly

complex even if the number of its basis functions is lower than the number of data samples, as too complex models will overfit the data and produce large prediction errors.

To obtain a polynomial regression model that does not overfit (nor underfit) and describes the relations in data sufficiently well, typically the subset selection approach (Hastie et al., 2003; Reunanen, 2006) is used where the goal is from a fixed full predetermined dictionary of basis functions to find a subset which corresponds to a model (a sparse polynomial) with the best predictive performance. This is done via combinatorial optimization. However, for the subset selection approach still the two issues remain – deficiency of adaptation as well as computational inefficiency.

Searching through all the possible combinations of basis functions takes double-exponential runtime as the number of combinations grows exponentially in the number of basis functions of the predetermined dictionary while the number of the basis functions in the dictionary grows exponentially in the number of input variables and “full” model’s degree (Hastie et al., 2003). This makes the exhaustive search through all the combinations impractical. The heuristic greedy search algorithms, such as forward selection (Hastie et al., 2003; Reunanen, 2006), substantially reduce the time and make it practical for not too large number of input variables and not too high degree. Nevertheless, the search time actually is still exponential, hindering their use in problems of larger dimensionality and hindering the removal of the restriction of a fixed degree.

The approach of subset selection assumes that the chosen fixed finite dictionary of the predefined basis functions contains a subset that is sufficient to describe the target relation sufficiently well. However, in most practical situations the required dictionary (and “full” model’s degree) is not known beforehand and needs to be either guessed or found by an additional search loop over the whole model building process, since it will differ from one regression task to another. In many cases, especially when the studied data dependencies are complex and not well studied, this means either a non-trivial and long trial-and-error process or acceptance of a possibly inadequate model.

This chapter presents a sparse polynomial regression model building approach which enables adaptive model building without restrictions on model’s degree and does it in polynomial time instead of exponential time (in the number of input variables, required degree, and target model’s complexity) as well as without the requirement to repeat the model building process. The required basis functions are automatically iteratively constructed using heuristic search specifically for the particular data at hand instead of choosing a subset from a very restricted finite user-defined dictionary (hence the approach is called Adaptive Basis Function Construction, ABFC). The basis function dictionary now becomes infinite and polynomials of arbitrary complexity can be generated bringing the desired flexibility to the model building process.

The remainder of this chapter is organized as follows. The next two sections give brief overview of polynomial regression and the subset selection approach. In Section 4 the ABFC approach is described. Section 5 outlines the related work. The results of the empirical evaluations of the proposed methods and their comparison to other well-known regression modelling methods are presented in Section 6. Section 7 concludes this chapter.

2. Polynomial regression

In standard regression formulation (Vapnik, 1995; Cherkassky & Mulier, 2007; Hastie et al., 2003) the goal is to estimate unknown real-valued function in the relationship

$$y = G(x) + \varepsilon, \quad (1)$$

where ε is independent and identically distributed random noise with zero mean, $x = (x_1, x_2, \dots, x_d)$ is d -dimensional input, and y is scalar output. The estimation is made based on a finite number of samples (training data) provided in form of matrix \mathbf{x} of input values for each sample and vector \mathbf{y} of output values for each corresponding sample. Using the finite number n of training samples $(\mathbf{x}_j, y_j), j=1,2,\dots,n$ one wants to build a model F that allows predicting the output values for yet unseen input values as closely as possible. Generally, a linear regression model may be defined as a linear expansion of basis functions:

$$F(x) = \sum_{i=1}^k a_i f_i(x), \quad (2)$$

where $\mathbf{a} = (a_1, a_2, \dots, a_k)^T$ are model's parameters, k is the number of basis functions included in the model (equal to the number of model's parameters), and $f_i(x), i=1,2,\dots,k$ are the included basis functions of the input x . As the model is linear in the parameters, the estimation of its parameters is typically done using the Ordinary Least-Squares (OLS) method (Hastie et al., 2003) minimizing the squared-error:

$$\mathbf{a} = \arg \min_{\mathbf{a}} \sum_{j=1}^n (y_j - F(\mathbf{x}_j))^2. \quad (3)$$

The basis function representation enables moving beyond pure linearity, by defining nonlinear transformations of x while still working with linear models (and employing OLS). For example, for $d=1$ a polynomial model of fixed degree p can be defined as follows:

$$F(x) = \sum_{i=0}^p a_i x^i. \quad (4)$$

Generally for a given d and p the total number of basis functions in a "full" polynomial, i.e. the total number of basis functions in the dictionary, is

$$m = \prod_{i=1}^p (1 + d/i). \quad (5)$$

3. Subset selection

Models which are too complex (i.e. that fit the training data too well causing overfitting) or too simple (i.e. that fit the data poorly causing underfitting) provide poor predictive performance for the future data. The most popular approach of controlling model's complexity is subset selection. The goal of subset selection is from a fixed full predetermined dictionary of basis functions to find a subset that provides the best predictive performance of the model (Hastie et al., 2003; Reunanen, 2006). Now in addition to the estimation of model's parameters, the structure of the model itself needs to be found.

The total number of possible subsets from a dictionary of size m is 2^m . This means that searching through all the possible subsets is in most cases impractical. Hence in subset selection heuristic search algorithms are used. They efficiently traverse the space of subsets, by adding and deleting basis functions of the model, and use model evaluation measure to direct the search into areas of increased performance. The typical examples of heuristic search algorithms are the greedy hill-climbing algorithms – Forward Selection (also known as Sequential Forward Selection, SFS) and Backward Elimination (also known as Sequential Backward Selection, SBS) (Hastie et al., 2003; Reunanen, 2006). However, there exist also more recently developed search strategies, such as Beam Search, Floating Search, Simulated Annealing, Genetic Algorithms etc. (Reunanen, 2006; Pudil et al., 1994; Russel & Norvig, 2002).

Summarizing (Russel & Norvig, 2002; Molina et al., 2002; Kohavi & John, 1997), in order to characterize a heuristic search problem one must define the following: 1) initial state of the search; 2) available state-transition operators; 3) search strategy; 4) evaluation measure; 5) termination condition. Note that in the context of model building the "initial state" is also called "initial model" and the "state-transition operators" are also called "model refinement operators".

In the subset selection approach for polynomial regression, typically the *initial state* is the model that corresponds to the empty subset, the subset with only the intercept term in it, full set of all the defined basis functions, or a randomly chosen subset. The typical basic *state-transition operators* are addition and deletion of a basis function. The typical *search strategy* is the hill-climbing (Russel & Norvig, 2002) which, in combination with the empty (or sufficiently small) subset as initial state and the addition operator, becomes SFS, but, in combination with the full subset as initial state and the deletion operator, becomes SBS. As the *evaluation measures* classically the statistical significance tests are used (Hastie et al., 2003; Dreyfus & Guyon, 2006). However, in model building currently two other strategies predominate (Cherkassky & Mulier, 2007; Dreyfus & Guyon, 2006): employment of complexity penalization criteria (also known as analytical criteria), e.g., the well-known Akaike's Information Criterion, AIC (Akaike, 1974; Burnham & Anderson, 2002), and the resampling techniques, e.g., Hold-Out, Cross-Validation (CV), and Bootstrap (Kohavi, 1995; Hastie et al., 2003; Dreyfus & Guyon, 2006). The *termination condition* typically corresponds to finding of a state in that none of the state-transition operators can lead to a better state (i.e. a local minimum).

In polynomial regression, increase in the model's degree leads to exponential growth of the number of basis functions in the dictionary, i.e. $O(m) = O(d^p)$ (Cherkassky & Mulier, 2007; Hastie et al., 2003) and to double-exponential growth of the number of all possible subsets (or the number of states in the state space): $O(2^m) = O(2^{d^p})$. When using one or both of the

two basic state-transition operators, the order of the branching factor of a state in the state space in the very first iteration of the search is already equal to the number of basis functions in the dictionary, i.e. it also increases exponentially.

Assuming that the “best” model found in the search process includes a total of k_* basis functions and that in each iteration the number of basis functions of the current model is increased by 1, the total number of evaluated models (subsets) is of order

$$O\left(\sum_{i=1}^{k_*} d^p\right) = O(d^p k_*) . \quad (6)$$

Hence for larger values of d and p (e.g., when m reaches thousands) subset selection is rendered impractical. Additionally, because of the branching factor’s direct dependence on the number of basis functions in the dictionary, the idea of unrestricted degree (i.e. dictionary of infinite size) is hardly applicable.

The computational problem could be somewhat reduced by choosing a sufficiently small but useful value of p before the actual model building is performed. However, generally the required maximal degree is not known beforehand and needs to be either guessed or found by additional search loop over the whole model building process, since it will differ from one regression task to another, which means either a non-trivial and long trial-and-error process or acceptance of a possibly inadequate model.

4. Adaptive Basis Function Construction

This section introduces Adaptive Basis Function Construction – a possible alternative to the classical subset selection approach. The goal of the ABFC approach is to overcome some of the limitations associated with the subset selection, outlined in the previous section. The ABFC approach is developed for sparse polynomial regression model building without restrictions on model’s degree, enables model building in polynomial time, and does not require repetition of the building process (in contrast to the subset selection approach). The required basis functions are automatically adaptively constructed specifically for data at hand, without using a restricted fixed finite user-defined dictionary. The dictionary in the ABFC is infinite and polynomials of arbitrary complexity can be constructed.

4.1 The models and the basis functions

Generally, a basis function in a polynomial regression model can be defined as a product of original input variables each with an individual exponent:

$$f_i(x) = \prod_{j=1}^d x_j^{r_{ij}} , \quad (7)$$

where \mathbf{r} is a $k \times d$ matrix of nonnegative integer exponents such that r_{ij} is the exponent of the j th variable in the i th basis function. Note that, when for a particular i th basis function $r_{ij} = 0$ for all j , the basis function is the intercept term.

Given a number of input variables d , matrix \mathbf{r} , with a specified number of rows k and with specified values for each of its elements, completely defines the structure of a polynomial model with all its basis functions. The set of basis functions, included in a model, is then

$$f = \left\{ \prod_{j=1}^d x_j^{r_j} \mid i = 1, 2, \dots, k \right\}. \quad (8)$$

For example, if $d = 3$ and $k = 4$, then the matrix

$$\mathbf{r} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 3 \\ 1 & 1 & 1 \end{bmatrix} \quad (9)$$

corresponds to the set

$$f = \{x_1^0 x_2^0 x_3^0, x_1^1 x_2^0 x_3^0, x_1^0 x_2^1 x_3^3, x_1^1 x_2^1 x_3^1\} = \{1, x_1, x_2 x_3^3, x_1 x_2 x_3\}, \quad (10)$$

which in turn corresponds to the model

$$F(x) = a_1 + a_2 x_1 + a_3 x_2 x_3^3 + a_4 x_1 x_2 x_3. \quad (11)$$

Formally, the problem of finding the best set of basis functions can be defined as finding the best matrix \mathbf{r} with the best combination of nonnegative integer values of its elements:

$$\mathbf{r}^* = \arg \min_{\mathbf{r}} J \left(\left\{ \prod_{j=1}^d x_j^{r_j} \mid i = 1, 2, \dots, k \right\} \right), \quad (12)$$

where $J(\cdot)$ is an evaluation criterion that evaluates the predictive performance of the regression model which corresponds to the set of basis functions.

As neither the upper bounds of \mathbf{r} elements' values nor the upper bound of k are defined, it is possible to generate sparse polynomials of arbitrary complexity, i.e. of arbitrary number of basis functions each with an arbitrary exponent for each input variable. This also means that the searchable state space is infinite.

4.2 The search process

Finding the "best" structure of matrix \mathbf{r} requires search. In this section the five components (outlined in Section 3) of a heuristic search problem are analyzed in the context of the ABFC approach.

Initial state. In ABFC, the state space is infinite therefore a natural initial state of the search is the state that corresponds to the simplest model located in the space. In the current study it is assumed that the simplest model is the one with a single basis function corresponding to

the intercept term. However, also other models could be used as initial states, e.g., an empty model (without any basis functions), a first degree “full” polynomial, or a small randomly generated model. Note that in the current study the basis function corresponding to the intercept term stays in the model at all times and is not allowed to be modified or deleted.

State-transition operators. Using efficient state-transition operators is vital for the search process to be efficient. The employed state-transition operators are the main methodological difference between the subset selection approach and the ABFC approach. Generally, there are two different basic types of modifications to an existing polynomial model: complication and simplification (Jekabsons & Lavendels, 2008a). In the subset selection approach, these are the addition and deletion operators. The addition operator makes the model more complex (by adding a new basis function) but the deletion operator makes it simpler (by deleting an existing basis function).

In the ABFC, the two standard operators from subset selection are replaced with other operators that not only add or delete basis functions but also work on the level of individual exponents, modifying the existing basis functions and creating modified copies of them. The basic idea is to use an operator that adds only the simplest (i.e. linear) basis functions which serve as a basic material for further construction of more complex functions using other operators. In this manner there is no need for an operator that explicitly tries to add basis functions of each possible combination of exponent values (as the addition operator in the subset selection). Hence the branching factor of the state space stays not only finite but also relatively small while the state space itself is infinite.

In this study, a set of the following four state-transition operators for the polynomial regression model building are proposed. Operator1: Addition of a new linear basis function with one of its exponents set to one and all the others set to zero. Operator2: Addition of an exact copy of an already existing (in the current model) basis function with one of its exponents increased by 1. Operator3: Decreasing of one of the exponents in one of the existing basis functions by 1. Operator4: Deleting of one of the existing basis functions. Figure 1 gives examples of the operators operating on a simple matrix.

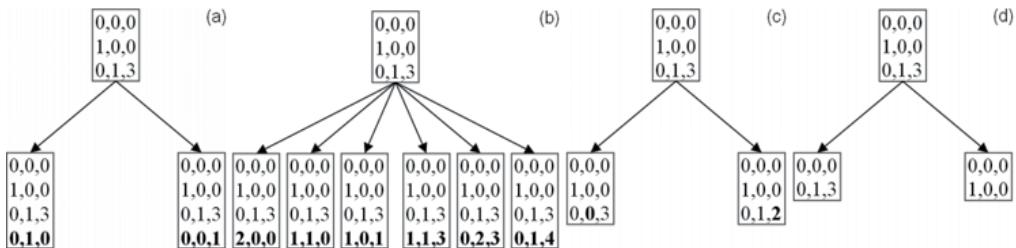


Fig. 1. Example of the four state-transition operators operating on a simple matrix:
(a) Operator1; (b) Operator2; (c) Operator3; (d) Operator4

The set of the four state-transition operators is sufficient to generate any polynomial model definable by the matrix \mathbf{r} . Their use can also be viewed as a piece of application-domain knowledge. While starting the search from the simplest model, the complication operators (the first two) do the main job – they “grow” the model. The simplification operators (the last two), on the other hand, work as “purifiers” – they decrease the unnecessarily high exponents and delete the unnecessary basis functions. Without the use of simplification operators, a regression model may contain unnecessarily high exponents and include too

many unnecessary basis functions, at the same time preventing truly necessary modifications (this is also known as the nesting effect (Pudil et al., 1994)) and increasing overfitting. Additionally, for all the state-transition operators a special care is taken to prevent basis function duplicates in the resulting model as well as to preserve the intercept term.

The initial state and the state-transition operators together form a state space. Figure 2 shows a small example of a state space in ABFC when the number of input variables is three and all the four state-transition operators are used. Each state represents a set of basis functions included in the regression model. The ordering of the states in the space is such that the simplest models and the simplest basis functions are reached first and, as the search goes on, increasingly complex models and basis functions can be reached.

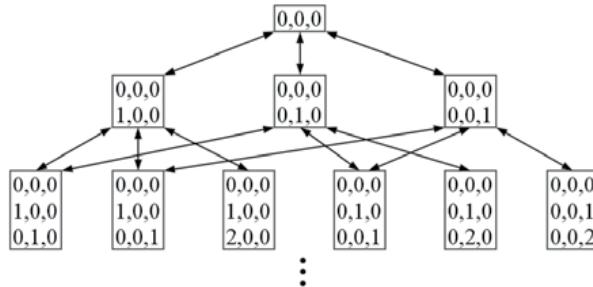


Fig. 2. A small example of the first three layers of a state space in ABFC when $d = 3$ (the space is infinite in the direction of more complex models)

In the Section 3, it is stated that in the subset selection approach the branching factor of a state in the state space increases exponentially with respect to the number of input variables d and pre-specified maximal degree p . In ABFC, the branching factor of the current state in the state space depends on d and on the number of basis functions k , already included in the current model. The upper bound of the number of possible modifications to a model using Operator1 is equal to d ; using Operator2 and Operator3 it is equal to dk ; and using Operator4 it is equal to k . So the upper bound of the branching factor is of order $O(d + 2dk + k) = O(dk)$ that is linear in respect to both d and k .

Search strategy. Most of the heuristic search algorithms of the hill-climbing type can be divided in two categories: those that assume the model state-transition operators to be of either or both the forward and the backward type (e.g., SFS, SBS, and Floating Search algorithms) and those that do not distinguish between the two types (e.g., Steepest Descent Hill-Climbing and Simulated Annealing). The four operators proposed in this study are naturally divided in forward (complication) and backward (simplification) operators; therefore in ABFC both categories of the search algorithms can be applied.

On the other hand, non-hill-climbing search algorithms, e.g., Genetic Algorithms and the like, employ completely different kind of operators (i.e. Crossover and Mutation). While they could be adapted to work with the infinite dictionary of basis functions, their major disadvantage is that, in contrast to the simple hill-climbing algorithms, they are not generally biased towards simpler models. In large state spaces they often spend most of the time exploring too complex models while the “best” ones are in fact mostly the relatively simple ones.

Evaluation measure. The proposed state-transition operators allow using the same methods for model evaluation and comparison as those used in subset selection. However, note that the model complexity penalization criteria, in contrast to the resampling techniques, usually require substantially lower computational resources as well as are less noisy creating less local minima in the state space.

Termination condition. Many different termination conditions can be used to terminate the search process. Some of most widely used ones are the following: a) a user pre-specified number of iterations is reached; b) a user pre-specified size of the model is reached; c) using the available state-transition operators the model could not be improved any further (evaluated by the chosen evaluation measure). The first two termination conditions require the user to set a hyperparameter value. This is a non-trivial task as usually the required information is not available. Adjusting such a hyperparameter may also require too large amounts of computational resources. In this study, the termination condition listed here as the last (c) is employed.

4.3 A concrete practical model building method

This section proposes Floating Adaptive Basis Function Construction (F-ABFC) – a concrete practical polynomial regression model building method, which is a special case of the ABFC approach.

The search procedure of the F-ABFC starts with the simplest model (with only the intercept term included) and uses the Floating Search strategy (hence the name of the method), in particular the Sequential Floating Forward Selection algorithm, SFFS (Pudil et al., 1994), together with the set of the four state-transition operators proposed in the previous section. In SFFS, the search process consists of two phases – the forward phase and the backward phase. In each iteration of the search, the forward phase is done only once but the number of times the backward phase is performed is determined dynamically. In the forward phase, all the models, which can be generated using the complication operators on the current best model, are evaluated and, if there is improvement over the current best model, the best of the new models is chosen as the new current best model and the search proceeds to the second phase. If there is no improvement, the whole search procedure is stopped. In the backward phase, on the other hand, all the models, which can be generated using the simplification operators on the current best model, are evaluated. In this phase ever simpler models are repeatedly generated and the phase is ended only when, using the available simplification operators, it is impossible to generate a model which is better than the current best one. After the second phase, the search process always proceeds to the next iteration (starting again with the first phase).

According to the studies of many researchers, the Floating Search algorithms, including SFFS, are some of the most efficient heuristic search algorithms for deterministic combinatorial optimization in terms of both required computational resources and quality of the results (Ferri et al., 1994; Jain & Zongker, 1997; Jain et al., 2000; Zongker & Jain, 1996; Pudil et al., 1994; Kudo & Sklansky, 2000; Reunanen, 2006). SFFS also does not have any adjustable hyperparameters, has a tendency to generate simpler models than many other algorithms, and is very simple to implement.

As in (Jekabsons & Lavendels, 2008a; Jekabsons, 2008), to evaluate the predictive performance of a newly generated model, to perform model comparisons, and to steer the

search in direction of the most promising models, in F-ABFC the Corrected Akaike's Information Criterion, AICC (Hurvich & Tsai, 1989) is used. AICC is defined as follows:

$$AICC = n \ln(MSE) + 2k + \frac{2k(k+1)}{n-k-1}, \quad (13)$$

where MSE is the Mean Squared Error of the model of interest in the training data. AICC evaluates model's predictive performance as a trade-off between its accuracy in the training data (the first term of (13)) and its complexity (the last two terms of (13)). Calculation of the AICC for a single model requires a single estimation of model's parameters using OLS and calculation of MSE in training data. The "best" model is that whose AICC value is the lowest.

The AICC is an improvement over the classical AIC (Akaike, 1974) with the third term in (13) added as a correction term intended for working with small-sized data sets. For problems with relatively small n , AICC is suited better than AIC but converges to AIC as n becomes large (Hurvich & Tsai, 1989). AIC and AICC theoretical justification is based on the relationship between the Kullback-Leibner information and the maximum likelihood principle (Burnham & Anderson, 2002). Note that AIC as well as AICC does not assume that the "true model" (which was presumably used to generate the data) is one of the candidates (Burnham & Anderson, 2002).

In (Jekabsons & Lavendels, 2008b), an issue of the F-ABFC is stated, that, because the branching factor of the ABFC's state space increases very slowly together with d and k , in special cases when the data is of low dimensionality (e.g., $d \leq 4$) and/or the existing structure in the data is very complex (i.e. a very complex model is required) the search algorithm may get stuck in a local minimum too early in the search returning a too simple and underfitted model.

As a remedy for this, here an additional recursion of the state-transition operators is proposed introducing one hyperparameter for the F-ABFC. The idea is to recursively create additional regression models from models already created from the current best model using the same state-transition operators with which they were initially created. This essentially means that if, for example, the recursion depth is set to 2, Operator1 will create not only linear basis functions but also basis functions of the second degree, Operator2 will create not only copies of basis functions with degree increased by 1 but also by 2, and Operator3 will not only try to decrease degrees by 1 but also by 2. However, as still none of the operators add more than one basis function to the model at a time, for the Operator4 the recursion is not used.

The recursion of the operators reduces the number of local minima in the state space which is especially important near the starting-point of the search (the initial model) and enables the search algorithm to find a much better model.

Presence of such a "recursion depth" hyperparameter is a disadvantage as now a user intervention might be required. However, for larger dimensionalities of the input space (when also the increased computational resources are required) it is reasonable to completely disable the recursion (by setting the hyperparameter equal to 1) as with large dimensionalities the branching factor increases sufficiently fast and the problem of too early local minima diminishes.

Figure 3 shows pseudo-code of F-ABFC's search procedure. Note that in practical implementations of F-ABFC maintaining the set of the newly generated models ("MODELS") is not required as a single model can be created, evaluated, and, if it turns out not to be an improvement, immediately discarded.

```

BestModel ← the simplest model
BestModel.PerformOLSandCalculateAICC
loop
    //forward phase
    MODELS ← {all models created from BestModel using Operator1 and Operator2,
    with no basis function redundancy}
    if RecursionDepth > 1 then
        for i ← 2 to RecursionDepth do
            MODELS ← MODELS ∪ {all models created from MODELS using the same
            operator (with which they were initially created), with no basis function
            redundancy}
        foreach Model in MODELS do
            Model.PerformOLSandCalculateAICC
        TestModel ← best of MODELS according to AICC
        if TestModel.AICC < BestModel.AICC then
            BestModel ← TestModel
        else
            break //break the main loop (exit the procedure)
    //backward phase
    loop
    MODELS ← {all models created from BestModel using Operator3 and Operator4,
    with no basis function redundancy}
    if RecursionDepth > 1 then
        for i ← 2 to RecursionDepth do
            MODELS ← MODELS ∪ {all models created from MODELS using Operator3
            (with which they were initially created), with no basis function redundancy}
        foreach Model in MODELS do
            Model.PerformOLSandCalculateAICC
        TestModel ← best of MODELS according to AICC
        if TestModel.AICC < BestModel.AICC then
            BestModel ← TestModel
        else
            break //break the sub-loop
    end loop
end loop
return BestModel

```

Fig. 3. Pseudo-code of F-ABFC's search procedure

In (Jekabsons & Lavendels, 2008a), a version of F-ABFC was developed that slightly differs from the one proposed here in that the method used one additional state-transition operator and the "recursion depth" hyperparameter was not introduced. The paper (Jekabsons & Lavendels, 2008a) empirically demonstrated the computational and predictive performance advantages of F-ABFC comparing to subset selection and a number of other popular regression modelling methods. F-ABFC advantages in real-world practical applications are demonstrated in (Kalnins et al., 2008a; Kalnins et al., 2009b) where it is applied for modelling bending and buckling behaviour of different composite material structures.

4.4 Computational considerations

Assuming that the “best” model found by the F-ABFC search procedure includes a total of k_* basis functions and in each iteration the number of basis functions in the current model is increased by 1, the total number of evaluated models is of order

$$O\left(\sum_{i=1}^{k_*} di\right) = O\left(dk_* \sum_{i=1}^{k_*} i\right) = O\left(dk_* \frac{k_*(k_*+1)}{2}\right) = O\left(dk_*^3 + dk_*^2\right) = O\left(dk_*^3\right). \quad (14)$$

Consequently, relatively to the typical subset selection methods, the efficiency of the F-ABFC increases together with the increase in the number of input variables and in the required nonlinearity of the model (the value of p) but decreases together with the increase in the complexity k_* of the “best” found model. Moreover, the relative efficiency of the subset selection additionally substantially decreases in the common case when the required value of p is unknown and needs to be found by trying different values.

Using F-ABFC together with OLS, the associated linear least-squares fitting, required for a single model to be evaluated, demand computations of order $O(nk^2 + k^3)$, where nk^2 operations are required for filling a $k \times k$ matrix and k^3 operations are required for solving a linear equation system (Hastie et al., 2003). However, none of the proposed state-transition operators operate on more than one basis function of a model at a time meaning that, each time the parameters of a newly created model are calculated, only one row and one column of the $k \times k$ matrix will change. Recalculating only the elements of the corresponding row and column, reduces the order of the computations to $O(nk + k^3)$. Moreover, as the Operator4 does not modify any basis function (only deletes one), the order of the computations for this particular operator reduces further to $O(k^3)$.

Yet it must be noted that the F-ABFC can still become computationally rather demanding, especially when the number of input variables and/or the number of samples in the training data gets very large. This is the price to pay for the high flexibility of the method.

4.5 Convergence of the search process

The F-ABFC’s search algorithm is cycle-free because a new model is allocated to “BestModel” (Figure 3) only if it is better than the old one (according to AICC). Moreover, as the AICC criterion tries to estimate model’s true predictive performance, the algorithm will seek for the best trade-off between too simple and too complex models and will stop somewhere in-between them. Additionally there is also a hard bound – the number of basis functions in a model will never exceed the number of samples in the training data as otherwise the OLS cannot estimate model’s parameters.

It should also be noted that, although the state space of F-ABFC is infinite, in practice the models of the best predictive performance are normally located in the part of the space that is relatively near to the initial state where all the models (and their basis functions) are relatively simple and do not yet neither overfit the data nor have basis functions more than samples in the training data. This also means that really only a small finite fraction of the whole infinite state space must be explored.

4.6 Selection bias, selection instability, and model averaging

There are two issues that to some extent plague all the methods of model building (including subset selection and ABFC), especially when working with relatively little data – selection bias and selection instability (also called selection variance). While the issues are attributable to virtually any model building method, they are commonly ignored frequently resulting in models of lower predictive performance.

Selection bias occurs when in the search procedure one uses the same data to compute model's parameters, to perform model building (i.e. evaluation of candidate models, selection of the best one, and steering the search in direction of the most promising models), and to select the final "best" model which will be returned as the result of the model building process (Reunanen, 2003; Reunanen, 2006, Loughrey & Cunningham, 2004; Jekabsons, 2008). The problem is that the more candidates are visited during the search, the greater the likelihood of finding a model that has high accuracy in the training set while having a very low predictive performance (accuracy in the test set) (Reunanen, 2003; Reunanen, 2006; Kohavi & John, 1997; Loughrey & Cunningham, 2004). The random fluctuations in the data will improve the evaluations of some models more than others.

The problem is relevant regardless of the model evaluation measure used – statistical significance tests, complexity penalization criteria, or resampling techniques. In addition, the selection bias occurs even when performing model evaluation using completely independent validation data set (Kohavi & John, 1997; Reunanen, 2006). In any case, the more intensive (relative to the number of samples) is the search process, the larger is the selection bias, and, the larger is the noise in the data, the potentially larger is the harm (in terms of overfitting) done by the selection bias.

While the deterministic search algorithms of the hill-climbing type (including the SFFS algorithm of the F-ABFC) are usually less intensive and consequently more robust against overfitting than, for example, Simulated Annealing or Genetic Algorithms (Loughrey & Cunningham, 2004; Guyon & Elisseeff, 2003), the problem of selection bias remains relevant. The second issue, selection instability, is related to the fact that small perturbations of the data (deleting or adding samples, adding noise, rescaling the values) can lead the model building process to vastly different models. This is because the large variability of estimates of the evaluation methods can lead to different local minima (Breiman, 1996; Kotsiantis & Pintelas, 2004; Guyon & Elisseeff, 2003; Cherkassky & Mulier, 2007). This variance is undesirable because variance is often the symptom of a "bad" model that does not generalize well and because the model may be failing to capture the "whole picture" (Guyon & Elisseeff, 2003).

One of the ways to reduce both the selection bias and the selection instability, is to employ model combining (also called model ensembling or averaging) techniques (Breiman, 1996; Opitz & Maclin, 1999; Cherkassky & Mulier, 2007; Jekabsons, 2008). While a typical model building process usually consists in choosing only one best description for the data discarding the remainder, combining a number of models in some reasonable manner appears more reliably accurate as this can have the effect of smoothing out erratic models that overfit the data and gain more stability in the modelling process.

A typical model combination procedure consists of a two-stage process (Cherkassky & Mulier, 2007). In the first stage, a number of different models are constructed. The parameters of these models are then held fixed. In the second stage, these individual models are linearly combined to produce the final model.

Both stages can be done in different ways. In this study, to increase the predictive performance of models built by the F-ABFC, a CV-type resampling of the training data together with unweighted model averaging (Opitz & Maclin, 1999; Duin, 2002) is employed. As this resampling and model averaging works on top of the F-ABFC, the method is called Ensemble of Floating Adaptive Basis Function Construction (EF-ABFC). During resampling, the whole training data is randomly divided into v disjoint subsets (v typically being equal to 10). Then v overlapping training data sets are constructed by dropping out a different one of these v subsets. Such procedure is also employed to construct training sets for v -fold CV, so model ensembles constructed in this way are also called cross-validated committees (Parmanto et al., 1996).

Combining models via simple unweighted averaging requires them to be not too underfitted as well as not too overfitted (Duin, 2002). To lower the overfitting, in each CV iteration the unused 10th data subset is used as a validation data set for “re-evaluation” (using MSE) of the best models of each F-ABFC iteration and for selection of the one “final best” model from any iteration. Note that this validation set is never used for model evaluation during the search. Instead it is used strictly only for the “re-evaluation” and “re-selection” after the F-ABFC search process has already ended. Also note that as an evaluation measure in the search algorithm still the AICC is applied. This “re-evaluation” using the validation data set can detect whether the search process at some iteration may have started to generate overfitted models and select a model of some earlier iteration that is (hopefully) not (or at least less) overfitted (see Figure 4).

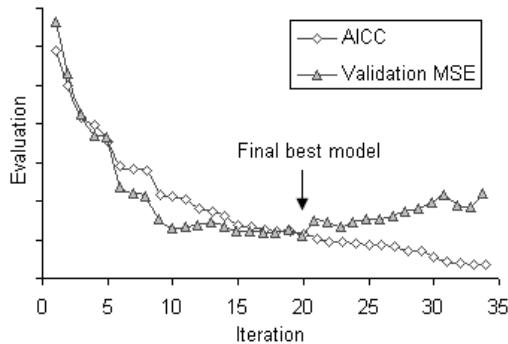


Fig. 4. An example of how a less overfitted model is selected using “re-evaluation” in validation set. Note that here starting from the 35th iteration the AICC values also start to increase (in contrast to the training error which always decreases) however this might be too late due to selection bias

The so far described process produces v models built by v independent F-ABFC runs each using a different combination of CV-partitioned data subsets. Next, the v models from the v CV iterations are combined using the unweighted model averaging. Note that prior to combining, all the models are re-fitted to the whole training data set (without the CV partitioning). This is done to compensate for the smaller training sets used during the individual model building.

Model combining by unweighted model averaging consists in taking an unweighted average of predictions of all the models:

$$F_{comb} = \frac{1}{v} \sum_{i=1}^v F_i , \quad (15)$$

where F_i is i th individual model from the i th CV iteration and F_{comb} is the combined model. For polynomial regression this simply means summation of all the polynomials and then a division of all the parameters of F_{comb} (that is also a polynomial) by v . Note that the parameter values of F_{comb} will not necessarily be optimal in the sense of the least-squares loss (in fact they will be optimal only in special cases, e.g., when all F_i 's are identical).

The employed model combining method is similar to Bagging (bootstrap aggregating (Breiman, 1996)) where the training set is bootstrapped (usually to build varied decision trees), and the unweighted average of the resulting models is taken.

Figure 5 gives an outline of the EF-ABFC model building process when the number of CV folds v is three. Note however that for all the practical applications of this study $v=10$ is used. This is because too small number of models in ensemble will yield too little diversity hindering the models to correct each others errors, but, on the other hand, using too many models will yield no further improvement (Breiman, 1996; Opitz & Maclin, 1999; Kotsiantis & Pintelas, 2004; Parmanto et al., 1996). Moreover, too large number of CV folds can yield unreliable validation MSE estimates for the selection of the individual final best models, as then the individual validation sets may be too small.

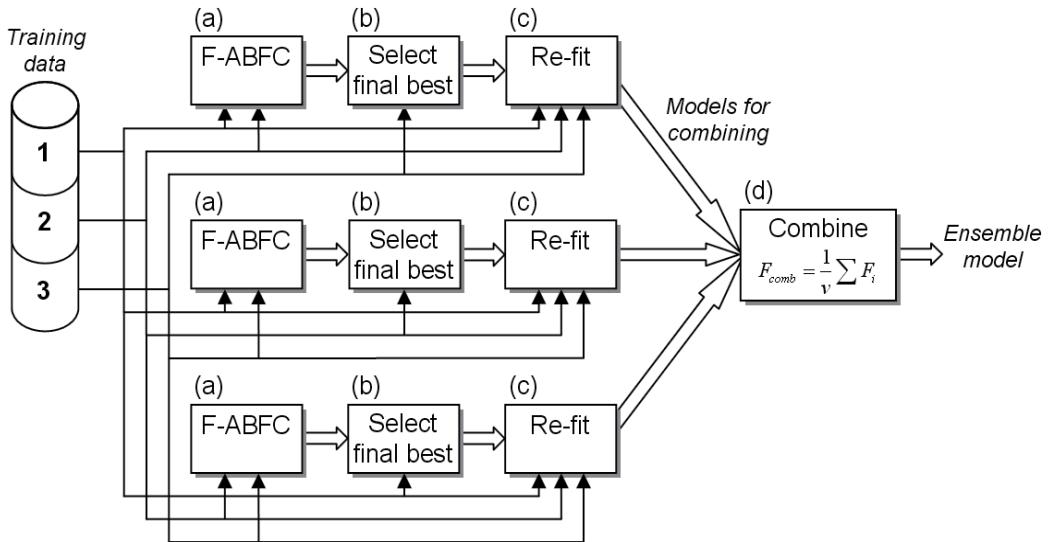


Fig. 5. An outline of the EF-ABFC modelling process when $v = 3$: (a) search for the best model according to AICC using F-ABFC; (b) select the one final best model according to MSE in validation data set; (c) ret-fit the model (recalculate its parameters) using the whole training data; (d) combine the models

In recent literature, there is ever growing confidence that model ensembles often perform better than individual models and consistently reduce prediction error (Breiman, 1996; Opitz & Maclin, 1999; Kotsiantis & Pintelas, 2004; Jekabsons, 2008). However, model ensembles are not always the best solutions (Kotsiantis & Pintelas, 2004): if there is too little

data, the gains achieved via an ensemble may not compensate for the decrease in accuracy of individual models, each of which now sees an even smaller training set. On the other end, if the data set is sufficiently large, even a single flexible model can be quite adequate. Using large data sets also substantially decreases potential selection bias, so superiority of EF-ABFC over F-ABFC in such situations is expected to diminish.

The most significant disadvantage of the EF-ABFC compared to F-ABFC is that it requires larger computational resources. However, the fact, that before the model combining the v models are built completely separately, allows for an easy parallelization of the process dividing the execution time by v . In this study however the parallelization is not done.

The paper (Jekabsons, 2008) empirically demonstrated the computational and predictive performance advantages of EF-ABFC comparing to subset selection and a number of other popular regression modelling methods. EF-ABFC advantages in real-world practical applications are demonstrated in (Kalnins et al., 2008b; Kalnins et al., 2009a) where it is applied for modelling bending and buckling behaviour of different composite material structures.

4.7 Remarks

This section covers various aspects (extensions, limitations, etc.) of the ABFC not discussed in the previous sections.

4.7.1 Incorporating domain knowledge

The ABFC methods attempt to model arbitrary dependencies in data with little or no knowledge of the system under study. In problems of moderate and large dimensionality the user usually is not required to tune any hyperparameters. However, if there is sufficient additional domain knowledge outside the specific data at hand, it may be appropriate to place some constraints on the final model. If the knowledge is fairly accurate, such constraints can improve the accuracy while saving computational resources.

For example, the constraints might be one or more of the following: 1) limiting the maximal degree of all the basis functions (similarly as in the subset selection), i.e. $0 \leq \sum_{j=1}^d r_{ij} \leq p$ for all i ; 2) limiting the maximal value of exponents for each particular input variable in all the basis functions, i.e. $0 \leq r_{ij} \leq p_j$ for all i , where p_j is maximal exponent of the j th variable; 3) restricting contributions of specific input variables that are not likely to interact with others so that those variables can enter the model in basis functions only solely – with exponents of all other variables fixed to zero. These constraints, as well as far more sophisticated ones, can be easily incorporated in the ABFC. However, note that in all the experiments described in this chapter no constraints are used.

4.7.2 Robustness

The ABFC methods described in this study estimate model parameters via minimization of the squared-error loss, i.e. using OLS. However, while the squared-error loss is the most commonly used, it is known that it loses its robustness against grossly outlying samples as well as in very sparse high-dimensional data sets (Cherkassky & Ma, 2002).

One solution of this problem is to use a more robust loss function. The squared-error loss in ABFC is not fundamental. Any other loss function can be used to estimate the parameters

and to evaluate the models by simply replacing the routine “PerformOLS and Calculate AICC” of the search procedure (Figure 3) with a more robust one. Note that while this would make the methods more robust, the computational advantage of OLS would be lost. In any case, gross outliers (in output variable as well as input variables) that can be detected through a preliminary data analysis should be considered for removal before applying ABFC.

4.7.3 Other types of basis functions

The ABFC methods described in this study can generate regression models with basis functions of only nonnegative integer exponents. However, in principle the exponents can also be allowed to take negative or even fractional values. Appropriate adaptation of the state-transition operators can enable generating such models. Keeping the same initial model as before, the search now could go in direction of both positive and negative exponents.

4.7.4 Integrating ABFC into other modelling methods

The result of running an ABFC procedure is a simple polynomial regression model. Such models are also utilized as “sub-models” in a number of other regression modelling methods. For example, the ABFC methods can be used in Polynomial Neural Networks (usually induced by Group Method of Data Handling) (Nikolaev & Iba, 2006) for adaptation of each individual neuron’s functional form and degree. The methods also can serve for generation of local regression models in Locally-Weighted Regression (also called Moving Least Squares) (Cleveland & Devlin, 1988; Kalnins et al., 2008b; Kalnins et al., 2005) adaptively generating a model each time a query is received. ABFC can also induce piecewise polynomial models for appropriately partitioned data sets.

The polynomial basis functions can also be viewed as nonlinear transformations (or features) of the original input variables. In this manner the ABFC methods can also be viewed as methods for automatic adaptive feature construction. For example, the constructed features can further serve as inputs for Support Vector Machines (Vapnik, 1995; Smola & Scholkopf, 2004) similarly to the features constructed using genetic algorithm in (Ritthoff et al., 2002).

All these applications of ABFC can make the original methods more flexible and therefore, if treated appropriately, produce models of higher predictive performance.

4.7.5 Using ABFC for solving classification problems

The ABFC methods can also be used for solving binary classification problems where the output variable y can take value of only either 0 or 1. This can be done, for example, by constructing basis functions for logistic regression (also called maximum entropy classifier) models. Logistic regression (Hastie et al., 2003; Witten & Frank, 2005) represents log odds of y being equal to 1 as a linear model:

$$\ln(P/(1-P)) = F(x) = \sum_{i=1}^k a_i f_i(x), \quad (16)$$

where P is the predicted probability of y being equal to 1. It is equivalent to the following representation of P :

$$P = 1/(1 + \exp(-F(x))). \quad (17)$$

The parameters \mathbf{a} of the model are usually estimated by minimizing the deviance:

$$-2 \sum_{j=1}^n (y_j \ln F(\mathbf{x}_j) + (1 - y_j) \ln(1 - F(\mathbf{x}_j))) \rightarrow \min. \quad (18)$$

Since there is no closed form solution to this minimization, the standard approach to solving it is to use iterative algorithms such as Iteratively Re-weighted Least-Squares (Hastie et al., 2003; Witten & Frank, 2005). Note that, in order to evaluate a model using AICC, the first term of (13) is replaced by the deviance.

F-ABFC and EF-ABFC for classification problems are implemented in the VariClass software tool freely available for non-commercial research and educational purposes at <http://www.cs.rtu.lv/jekabsons/>.

5. Related work

There exist also other polynomial regression modelling methods which use wide, potentially infinite, dictionaries of basis functions. In (Sutton & Matheus, 1991) an algorithm is proposed which starts model building with a first-degree model, with all the input variables already included in the model, and iteratively creates a user-predefined number of products of the already included basis functions thereby creating new basis functions. In (Orosz & Anderson, 1994) a modification of the algorithm is proposed where the initial model has none of the input variables included, however there was no empirical success and it was concluded that in practical applications the algorithms have three major disadvantages: inability to construct all the necessary basis functions, inability to discard unnecessary basis functions, and high sensitivity to noise and to number of samples in data.

More recently a different method was developed which can be seen also as a special case of the ABFC approach – Constrained Induction of Polynomial Equations for Regression, CIPER (Todorovski et al., 2004). CIPER was initially developed in the context of differential equation discovery, inductive databases, and constraint-based data mining. CIPER uses two state-transition operators and a Beam Search strategy. The first state-transition operator adds a new linear basis function while the second increases a single exponent of a single basis function. In (Jekabsons & Lavendels, 2008a), CIPER was empirically compared to F-ABFC and it was concluded that CIPER suffers from the nesting effect (Pudil et al., 1994) and has a tendency of getting stuck in local minima too early in the search. This is because CIPER is not able to preserve the structure of any of included basis functions (its second operator increases an exponent in an existing basis function but does not take into consideration the possibility that both versions of the basis function may be required) as well as because it is not able to simplify a model – decrease unnecessarily high exponents or discard unnecessary basis functions. In F-ABFC these issues are solved using Operator2 and the simplification operators.

Some similar ideas of constructing new features as combinations of original input variables are applied also in different other approaches. For example, in (Ritthoff et al., 2002) a feature construction method is proposed in which a genetic algorithm constructs linear and nonlinear combinations of original input variables further used as inputs for Support Vector Machines. In (Bloedorn & Michalski, 1998), on the other hand, the feature construction idea is used for data-driven expansion of the input space for induction of decision rules and decision trees.

6. Experiments

This section presents the results of comparisons of the proposed ABFC methods to the methods of subset selection and to a number of other well known state-of-the-art regression modelling methods using a series of synthetic and real-world regression data sets. The goal is to gain some understanding of the properties of F-ABFC and EF-ABFC and to evaluate their performance in both accuracy and speed. All the experiments were performed on a Pentium IV 2.4GHz machine with 1.5GB RAM.

In all the experiments, predictive performance of a model is measured either using a completely independent test data set or using Cross-Validation. In any case the performance of a model is measured in terms of Relative Root Mean Squared Error:

$$RRMSE = 100\% \times RMSE / SD = 100\% \times \sqrt{\frac{1}{n_t} \sum_{j=1}^{n_t} (y_j - F(\mathbf{x}_j))^2} \Bigg/ \sqrt{\frac{1}{n_t} \sum_{j=1}^{n_t} (y_j - \bar{y})^2}, \quad (19)$$

where n_t is the number of samples in the test data set, $F(\mathbf{x}_j)$ is the predicted value corresponding to the value of y_j , and \bar{y} is the mean of all the y values in the test set. While RMSE (Root Mean Square Error) represents model's deviation from the data, the SD (Standard Deviation) captures how irregular the problem is. The lower the value of RRMSE, the more accurate is the model. The final RRMSE values stated are the values averaged over all evaluations.

All the employed regression modelling methods, except Regression Trees, Model Trees, Support Vector Machines, and Multi-Layer Perceptrons, are implemented in VariReg software tool version 0.9.21 freely available for non-commercial research and educational purposes at <http://www.cs.rtu.lv/jekabsons/>.

6.1 Synthetic data sets

To compare the performance of the proposed ABFC methods against subset selection (as well as against "full" polynomials with no subset selection) in different conditions of signal-to-noise ratio (SNR) and training data size, here two test functions are used – Synth1 (4 input variables) and Synth2 (10 input variables):

$$y_{Synth1} = \exp(2x_1 \sin(\pi x_4)) + \sin(x_2 x_3), \quad (20)$$

$$y_{Synth2} = (x_1^3 + (x_2 + x_3)(x_4 + x_5)) / (1 + x_6 x_7) + 0x_8 + 0x_9 + 0x_{10}. \quad (21)$$

For Synth1 the values of x are uniformly distributed in the interval [-0.25, 0.25]. For Synth2 they are uniformly distributed in the interval [0, 1]. For each test function three training set

sizes (25 samples, 50 samples, and 100 samples) and three signal-to-noise ratios (no noise, SNR = 4, and SNR = 2) are used – a total of nine cases for each function. For each case a series of 20 training data sets are generated (randomly sampled in the domain of \mathbf{x}) so that in each case for each regression modelling method the model building task is performed 20 times. For each test functions a single test data set is generated containing 5000 samples randomly sampled in the domain of \mathbf{x} . The test data sets do not contain noise.

The heuristic search algorithms used for the subset selection are the SFS and the SFFS (the same algorithm adaptation of which is used in the ABFC methods). The algorithms are used together with the AICC criterion (also the same which is used in the ABFC methods). Note that the “recursion depth” hyperparameter of F-ABFC is set equal to 2 for Synth1 and equal to 1 (no recursion) for Synth2.

As for the full polynomials (FP) and the subset selection methods the desirable degree p is not known beforehand, the modelling results of these methods are stated in two forms: 1) average performance of models of a fixed p ; 2) average performance when a range of values for p are tried and the model of the lowest RRMSE value is picked. However, note that this second type of procedure for FP/SFS/SFFS is rather optimistic (in the sense of both predictive performance and speed) as for correct and fair evaluations there would be an additional validation data set or a Cross-Validation loop required.

No noise	$n = 25$		$n = 50$		$n = 100$	
Method	RRMSE	Time (s)	RRMSE	Time (s)	RRMSE	Time (s)
FP, $p \in [1, 4]$	9.29 (1.88)	-	6.74 (0.55)	-	0.78 (0.21)	-
SFS, $p = 2$	7.17 (1.03)	< 0.1	6.38 (0.58)	< 0.1	5.63 (0.28)	< 0.1
SFS, $p = 6$	0.77 (2.24)	0.3	0.06 (0.02)	1.4	0.04 (1e-2)	8.0
SFS, $p = 10$	3.19 (7.41)	1.8	0.03 (0.04)	16.1	2e-3 (7e-3)	104.3
SFS, $p \in [1, 10]$	0.77 (2.24)	4.4	0.03 (0.03)	34.1	2e-3 (7e-3)	226.1
SFFS, $p \in [1, 10]$	0.77 (2.24)	4.5	0.03 (0.03)	30.4	2e-4 (1e-4)	236.9
F-ABFC	0.11 (0.15)	0.1	0.01 (0.02)	2.0	3e-7 (5e-7)	43.8
EF-ABFC	0.27 (0.31)	0.8	0.02 (0.02)	11.5	1e-4 (4e-4)	250.6
SNR = 4						
FP, $p \in [1, 4]$	42.71 (16.31)	-	18.36 (2.53)	-	12.12 (1.59)	-
SFS, $p = 2$	24.24 (10.87)	< 0.1	15.62 (2.99)	< 0.1	10.64 (2.38)	< 0.1
SFS, $p = 6$	59.37 (28.09)	0.1	41.37 (11.76)	0.3	25.60 (9.49)	0.8
SFS, $p = 10$	112.02 (149.28)	0.7	74.10 (40.38)	3.4	39.23 (10.92)	7.9
SFS, $p \in [1, 10]$	24.24 (10.87)	1.7	15.62 (2.99)	8.7	10.64 (2.38)	19.5
SFFS, $p \in [1, 10]$	24.24 (10.87)	1.8	15.62 (2.99)	7.6	10.64 (2.38)	21.0
F-ABFC	39.05 (17.97)	< 0.1	33.13 (15.64)	0.1	22.64 (10.73)	0.3
EF-ABFC	20.24 (6.76)	0.3	13.65 (3.82)	0.8	9.08 (3.16)	2.1
SNR = 2						
FP, $p \in [1, 4]$	79.40 (37.25)	-	35.97 (8.35)	-	21.79 (4.29)	-
SFS, $p = 2$	36.35 (12.40)	< 0.1	26.51 (9.87)	< 0.1	18.55 (4.35)	< 0.1
SFS, $p = 6$	88.32 (32.57)	0.1	70.34 (24.81)	0.3	47.11 (16.95)	1.0
SFS, $p = 10$	209.98 (213.00)	0.8	99.26 (40.07)	2.8	78.04 (31.78)	8.1
SFS, $p \in [1, 10]$	36.35 (12.40)	1.7	26.51 (9.87)	5.9	18.55 (4.35)	18.7
SFFS, $p \in [1, 10]$	36.35 (12.40)	1.7	26.64 (10.08)	6.3	18.55 (4.35)	19.5
F-ABFC	58.43 (19.72)	< 0.1	72.44 (62.43)	< 0.1	39.93 (19.91)	0.2
EF-ABFC	35.23 (11.04)	0.3	24.94 (6.18)	0.7	17.67 (4.45)	1.8

Table 1. The results of the performed experiments for function Synth1

The results of the performed experiments are summarized in Table 1 and Table 2 in terms of mean RRMSE value, with its standard deviation reported in parenthesis, and elapsed time. Note that, due to the space constraints, for fixed degrees only the results of $p \in \{2, 6, 10\}$ (for Synth1), $p \in \{2, 5\}$ (for Synth2) are given. Detailed results are available at <http://www.cs.rtu.lv/jekabsons/>.

Figure 6 and Figure 7 visualizes the performance changes of the methods for different training set sizes and SNRs.

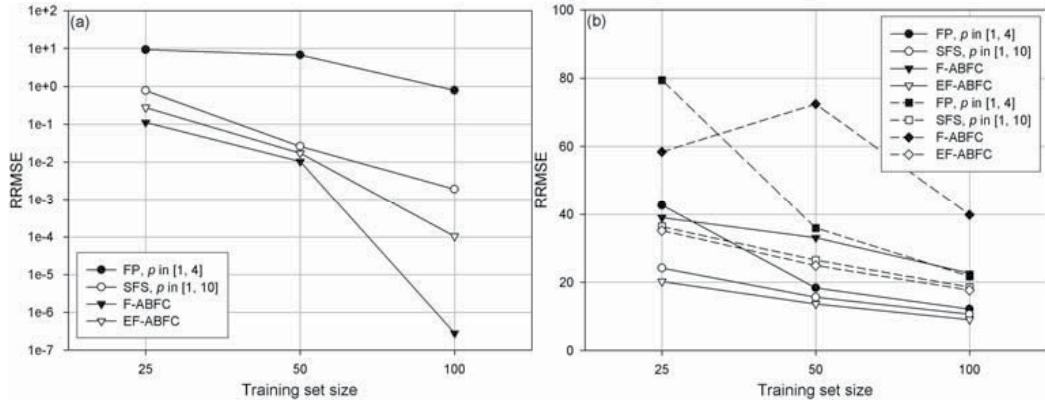


Fig. 6. Performance of the methods for function Synth1 for the different training set sizes and SNRs: (a) no noise; (b) SNR = 4 (solid lines) and SNR = 2 (dashed lines)

No noise		$n = 25$		$n = 50$		$n = 100$	
Method	RRMSE	Time (s)	RRMSE	Time (s)	RRMSE	Time (s)	
FP, $p \in [1, 2]$	45.40 (4.86)	-	38.73 (2.06)	-	13.07 (1.37)	-	
SFS, $p = 2$	38.17 (13.99)	< 0.1	19.13 (3.20)	0.2	11.27 (1.35)	1.0	
SFS, $p = 5$	80.25 (23.80)	12.8	29.13 (12.73)	53.6	4.66 (2.79)	542.9	
SFS, $p \in [1, 5]$	38.17 (13.99)	14.7	19.13 (3.20)	57.1	4.64 (0.88)	658.7	
SFFS, $p \in [1, 5]$	37.40 (12.36)	15.9	20.20 (4.03)	82.5	4.01 (2.87)	680.4	
F-ABFC	52.86 (11.46)	< 0.1	13.14 (6.96)	0.7	1.59 (1.58)	16.5	
EF-ABFC	56.39 (16.40)	0.3	12.92 (3.08)	4.2	0.95 (0.46)	98.7	
SNR = 4							
FP, $p \in [1, 2]$	51.78 (8.53)	-	41.31 (3.25)	-	37.38 (1.51)	-	
SFS, $p = 2$	58.85 (15.18)	< 0.1	35.44 (7.05)	0.1	23.63 (4.02)	0.3	
SFS, $p = 5$	180.68 (68.02)	10.7	82.78 (23.99)	66.0	62.00 (10.43)	258.7	
SFS, $p \in [1, 5]$	58.85 (15.18)	13.1	35.44 (7.05)	77.8	23.63 (4.02)	298.8	
SFFS, $p \in [1, 5]$	61.01 (17.19)	14.2	35.88 (8.69)	78.2	24.21 (3.57)	373.5	
F-ABFC	79.07 (35.39)	< 0.1	45.59 (9.28)	0.1	28.89 (7.25)	0.5	
EF-ABFC	62.79 (11.47)	0.3	35.57 (7.35)	1.3	20.37 (3.51)	6.3	
SNR = 2							
FP, $p \in [1, 2]$	61.23 (9.17)	-	46.61 (4.73)	-	40.81 (3.12)	-	
SFS, $p = 2$	73.81 (17.63)	< 0.1	51.69 (8.15)	0.1	37.20 (6.57)	0.2	
SFS, $p = 5$	180.68 (68.02)	11.0	135.99 (37.13)	47.6	115.01 (31.12)	208.9	
SFS, $p \in [1, 5]$	73.81 (17.63)	13.3	47.92 (6.96)	57.4	37.20 (6.57)	253.0	
SFFS, $p \in [1, 5]$	76.43 (11.56)	14.3	50.41 (9.44)	64.3	35.15 (5.16)	369.3	
F-ABFC	82.42 (18.79)	< 0.1	68.08 (15.40)	0.1	49.61 (13.73)	0.3	
EF-ABFC	70.84 (9.52)	0.2	51.11 (8.79)	0.8	34.54 (5.93)	3.9	

Table 2. The results of the performed experiments for function Synth2

The results in Table 1 indicate that for noise-free data the F-ABFC outperforms its much slower ensembled extension EF-ABFC while for noisy data it is vice versa. When the data contains noise, the F-ABFC here can be outperformed even by full polynomials which mostly give some of the worst performances. This suggests that for noisy data it is important to curb the flexibility of F-ABFC – to use the EF-ABFC even when the data is sparse.

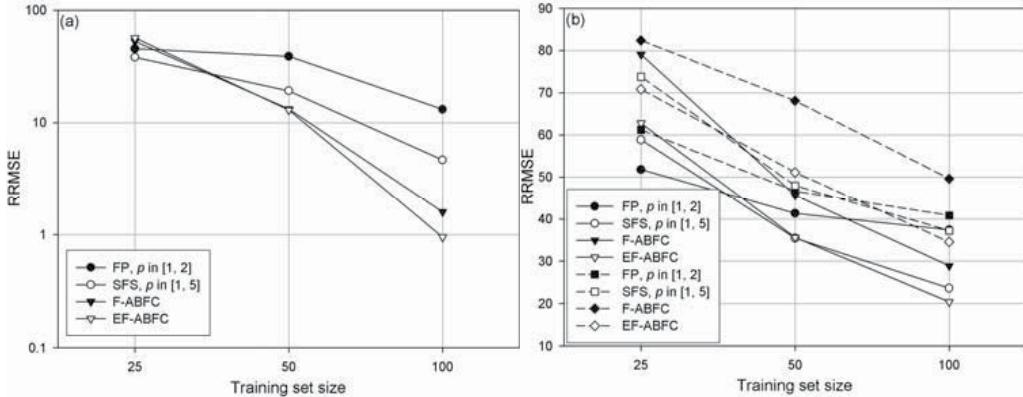


Fig. 7. Performance of the methods for function Synth2 for the different training set sizes and SNRs: (a) no noise; (b) SNR = 4 (solid lines) and SNR = 2 (dashed lines)

The results in Table 2 partially confirm those in Table 1 except that this time the EF-ABFC is always more accurate than F-ABFC which may be caused by the three irrelevant input variables (pure noise) in the data on which the Synth2 does not depend. Additionally, as now in the case of $n = 25$ the data are very sparse, for this case the ABFC methods are just too flexible – they largely overfit the data even when there is no additional noise.

Overall, the results for both Synth1 and Synth2 indicate the computational advantage of the ABFC methods in situations when the required regression model is more complex (of higher degree). And this advantage grows with the dimensionality of the problem.

For noisy data the best choice of p for SFS/SFFS almost always was 2. Then the speed of a single SFS/SFFS search can be outperformed only by F-ABFC. However, as the best p value is actually unknown and a number of values must be tried, F-ABFC as well as EF-ABFC is still faster than the subset selection.

Finally, it must also be noted that the overall results show evidence that for subset selection the choice of the search algorithm (either SFS or SFFS) was of no great importance. Therefore further in this study only the SFS algorithm for subset selection is considered.

6.2 Real-world machine learning data sets

The real-world machine learning regression data sets used are: autoMPG (7 input variables, 392 samples), AutoPrice (15 input variables, 159 samples), Bodyfat (14 input variables, 252 samples), Fishcatch (7 input variables, 158 samples), Housing (13 input variables, 506 samples), HousingNOX (13 input variables, 506 samples), MachineCPU (6 input variables, 209 samples), Pyrimidines (27 input variables, 73 samples), Servo (4 input variables, 167 samples), and Stock (9 input variables, 950 samples). The data sets are from UCI Machine Learning Repository (<http://www.ics.uci.edu/~mlearn/MLRepository.html>), Luis Torgo's data sets repository (<http://www.liaad.up.pt/~ltorgo/Regression/DataSets.html>), and

Weka collection of data sets (<http://www.cs.waikato.ac.nz/ml/weka/>). They are chosen because of the relatively low number of samples, which is common in real-world practical applications, as well as because of mostly continuous input variables and no missing values. Here the performances of the different regression modelling methods are evaluated using 10-fold Cross-Validation. Note that prior to dividing the data into Cross-Validation folds, the order of the samples was randomized.

The goal of the performed experiments is to compare the proposed ABFC methods to the methods of subset selection and to other well known state-of-the-art regression modelling methods using a set of real-world regression data sets. The compared methods are the following: FP, SFS, F-ABFC, EF-ABFC, Multivariate Adaptive Regression Splines (MARS) (Friedman, 1993), M5' Regression Trees (RT) (Witten & Frank, 2005), M5' Model Trees (MT) (Witten & Frank, 2005), Support Vector Machines (SVM) (Vapnik, 1995; Smola & Scholkopf, 2004), and Multi-Layer Perceptrons (Witten & Frank, 2005). Note that for none of the methods any of the hyperparameters were manually tuned. MARS was that of piecewise-cubic type essentially without special limitation of the number of basis functions (i.e. the limit was 500) and with the smoothing parameter (the number of degrees of freedom associated with one basis function) either fixed to the default value of 3 or found using an additional 10-fold Cross-Validation from the range [1, 5] with step size 0.5. SVM used Radial Basis Function kernel and improved Sequential Minimal Optimization algorithm (Shevade et al., 1999) for which the complexity parameter and the gamma parameter were found using grid search and Cross-Validation from the range $\{10^{-1}, 10^0, 10^1, 10^2\}$ for the complexity parameter and $\{10^{-2}, 10^{-1}, 10^0, 10^1\}$ for the gamma parameter. MLP had one hidden layer with the “best” number of neurons determined by 10-fold Cross-Validation from the range $\{10, 20, 30, 40\}$ and the weights were optimized using backpropagation. As implementations of RT, MT, SVM, and MLP the Weka software (Witten & Frank, 2005) was employed with its default parameters. Also note that for the ABFC methods the recursion of the state-transition operators was never used.

The results of the performed experiments are summarized in Table 3 in terms of mean RRMSE value, with the standard deviation reported in parenthesis, and elapsed time. Here the modelling results of SFS are stated in the same two forms as in Section 6.1 except that for the different data sets (different in size, in number of input variables, and in required model complexity) the values of p are tried in different intervals (named “ $p = \text{automatic}$ ”) – the search for the best p is started with the first degree and p is increased as long as the RRMSE value improves. The results of FP are not stated, as due to matrix singularity in OLS for Pyrimidines data set the parameter values of FP models could not be calculated. Also note that, due to the space constraints, only the results averaged over all the data sets are given. Detailed results are available at <http://www.cs.rtu.lv/jekabsons/>.

From the results of the experiments it is concluded that in terms of predictive performance, the EF-ABFC outperformed all the other regression modelling methods involving polynomials as well as showed high competitiveness against the other “non-polynomial” methods. In terms of computational cost, both ABFC methods outperformed subset selection but were inferior to some of the “non-polynomial” methods, especially RT, MT, and MARS without CV.

Method	RRMSE	Time (s)
SFS, $p = 1$	49.64 (12.05)	< 0.1
SFS, $p = 2$	40.89 (15.11)	4.8
SFS, $p = 3$	37.83 (15.70)	227.4
SFS, $p = 4$	47.10 (29.17)	2486.8
SFS, $p = \text{automatic}$	34.83 (10.20)	2207.5
F-ABFC	39.61 (16.93)	108.7
EF-ABFC	31.24 (10.86)	607.8
RT	50.76 (9.85)	0.3
MT	34.79 (12.35)	0.4
MARS	40.81 (17.29)	3.2
MARS + CV	39.87 (15.57)	265.8
SVM	31.87 (10.73)	360.5
MLP	41.50 (18.04)	345.2

Table 3. The average results of the performed experiments for the ten machine learning data sets

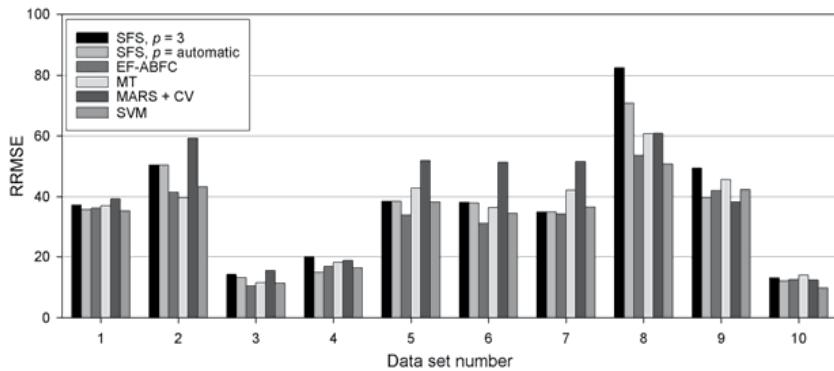


Fig. 8. RRMSE values of the six best methods for the ten data sets

For the different data sets, the best found degree p for SFS with “ $p = \text{automatic}$ ” varied in range [1, 6] (with the average value of 3.0), meaning that the maximal checked value of p was 7 (though for only one of the data sets). However, the average degree of models constructed by F-ABFC and EF-ABFC was 6.4 and 7.2 correspondingly. If on average for SFS such large values of p would be tried (instead of only 3.0), the SFS would take considerably more time (orders of magnitude) to complete.

6.3 Real-world metamodeling data sets

In many different industrial applications, to cut down the computational cost of complex, high fidelity scientific and engineering simulations, regression models (in the context also referred to as metamodels or surrogate models) are constructed that mimic the behaviour of the simulation models as closely as possible while being computationally much cheaper to employ (Myers & Montgomery, 2002; Chen et al., 2006; Martin & Simpson, 2005; Kalnins et al., 2008b; Kalnins et al., 2008a; Kalnins et al., 2009a; Kalnins et al., 2009b). The process of design optimization involving metamodeling usually comprises three major steps which may be interleaved iteratively: 1) selection of samples (known as design of experiments); 2) construction of metamodel and estimation of its predictive performance; 3) employment of the metamodel in design optimization (i.e., finding the best values for input variables

with which the studied system achieves the optimum response), design space exploration, what-if analysis, sensitivity analysis, and other routine tasks.

The metamodelling problem addressed here is modelling of the behaviour of “I-core” all-metal laser-welded sandwich panels under bending load for further design optimization and analysis in application as deck panels in a modularised watercraft concept (Kalnins et al., 2008a). The problem has six input variables and four output variables. The data are generated using finite element simulations and contains 500 samples distributed in the input space using sequential experimental design (Auzins, 2004).

Originally, metamodeling was associated with low-degree (usually quadratic) polynomial models. They have been well accepted in engineering practice, as they require only little data and are computationally very efficient. However, it is understood that they are loosing efficiency when highly nonlinear behaviour should be approximated.

In this section the compared regression modelling methods are the same as in the Section 6.2 with an addition of three methods which are rather popular in metamodeling literature: Locally-Weighted Polynomials (LWP) (Cleveland & Devlin, 1988; Kalnins et al., 2008b; Kalnins et al., 2005), Radial Basis Functions (RBF) (Gutmann, 2001), and Kriging (Martin & Simpson, 2005, Lophaven et al., 2002). Note again that for none of the methods any of the hyperparameters were manually tuned. LWP used the Gaussian weight function with the value of the bandwidth parameter found by Leave-One-Out Cross-Validation. Note that the LWP has a similar issue of degree p selection as FP and SFS, so here a number of different degrees are tried in the interval [1, 4]. RBF used the multi-quadratic basis functions with the shape parameter fixed to 1. Kriging used first-degree polynomial as a trend function and employed the Gaussian correlation function. Note that the used source code for the Kriging technique was developed by (Lophaven et al., 2002). Also note that for the ABFC methods in the performed experiments the recursion of the state-transition operators was never used.

The results of the performed experiments are summarized in Table 4 in terms of mean RRMSE value, with its standard deviation reported in parenthesis, and elapsed time. Here the performances of the different regression modelling methods are evaluated using 5-fold Cross-Validation. The modelling results of FP and SFS are stated in the same two forms as in Section 6.1. Note that, due to the space constraints, only the results averaged over all the data sets are given. Detailed results (as well as the utilized data sets) are available at <http://www.cs.rtu.lv/jekabsons/>.

The results in Table 4 indicate that with the four metamodeling data sets (all of which are essentially noise-free) the ensembling of F-ABFC models was not necessary – the accuracy advantage of EF-ABFC is negligible while it is computationally about ten times slower than simple F-ABFC. However, both F-ABFC and EF-ABFC outperformed subset selection in terms of predictive performance as well as in terms of speed. In respect to the other methods the ABFC approach once again is highly competitive, especially the faster F-ABFC method.

With the metamodeling data sets, on average the best degree p for SFS was 6.3 while the average degree of models constructed by F-ABFC and EF-ABFC was 9.2 and 7.9 correspondingly. Similarly to the conclusions of the previous section, trying these larger values of p for SFS would take orders of magnitude more time to complete.

Method	RRMSE	Time (s)
FP, $p = 1$	49.85 (4.82)	-
FP, $p = 2$	23.81 (3.10)	-
FP, $p = 3$	12.81 (1.77)	-
FP, $p = 4$	9.88 (1.46)	-
FP, $p \in [1, 4]$	9.17 (1.28)	-
SFS, $p = 1$	49.75 (4.68)	< 0.1
SFS, $p = 2$	23.42 (3.15)	0.2
SFS, $p = 3$	11.74 (1.84)	4.2
SFS, $p = 4$	7.31 (1.35)	41.1
SFS, $p = 5$	5.62 (1.14)	220.3
SFS, $p = 6$	5.03 (0.78)	959.1
SFS, $p = 7$	5.05 (1.12)	1828.4
SFS, $p \in [1, 7]$	4.92 (0.75)	3053.4
F-ABFC	4.28 (0.55)	71.9
EF-ABFC	4.19 (0.55)	715.4
RT	60.18 (7.87)	1.0
MT	22.27 (4.97)	4.7
MARS	5.87 (0.96)	0.9
MARS + CV	5.31 (0.84)	77.5
SVM	13.14 (2.57)	414.7
MLP	8.47 (1.03)	331.3
LWP, $p = 1$	40.22 (4.12)	2.8
LWP, $p = 2$	20.23 (2.79)	26.2
LWP, $p = 3$	11.66 (1.68)	210.6
LWP, $p = 4$	9.76 (1.42)	1576.7
RBF	14.48 (3.42)	1.9
Kriging	7.40 (1.21)	16.3

Table 4. The average results of the performed experiments for the four metamodelling data sets

Note that in practice it turns out that the user all too often does model building in a “one-shot” manner, without consideration of different settings for a modelling method. With FP and SFS (as well as LWP) it could mean that almost any of the results stated in Table 4 (as well as in the other tables from previous sections) may be accepted as the final. Iterative and adaptive methods like those of ABFC, on the other hand, have the potential of relatively rapidly producing accurate models without the configuration burden.

7. Conclusion

This chapter introduced Adaptive Basis Function Construction – an adaptive sparse polynomial regression model building approach which can also be viewed as an alternative to the classical subset selection approach. In contrast to subset selection, the ABFC approach does not require putting restrictions on model’s degree, enables model building in polynomial time, and does not require repetition of the model building process. The basis functions required for the model are automatically adaptively constructed using heuristic search specifically for data at hand without using a restricted fixed finite user-predefined dictionary. The dictionary in the ABFC is infinite and polynomials of arbitrary complexity can be constructed.

In most of the performed empirical experiments, the ABFC methods outperformed subset selection in terms of predictive performance as well as in terms of the amount of required

computational resources. Moreover, in respect to the other well-known state-of-the-art regression methods, the ABFC approach is also highly competitive. Additionally, the ABFC methods have advantages also in their simple application – the underlying algorithms have very small number of hyperparameters for the user to tune and result in simple explicit equations employable without specialized software.

Comparing the two specific methods F-ABFC and EF-ABFC, it is concluded that EF-ABFC has predictive performance advantage over F-ABFC when the data contains noise, be it in terms of signal-to-noise ratio or in terms of irrelevant input variables. On the other hand, F-ABFC is much faster than EF-ABFC and can produce more accurate models when the data is noise-free. Nevertheless, both methods may require the “recursion depth” hyperparameter to be set to a value higher than 1 when the data is of low dimensionality (e.g., $d \leq 4$) and/or the existing structure in the data requires a very complex model.

As future work, some of the ideas described in Section 4.7 could be pursued.

Software (including open source) implementing the ABFC methods, as well as most of the other regression methods employed in this chapter, can be downloaded at the author's webpage: <http://www.cs.rtu.lv/jekabsons/>.

8. References

- Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, AC-19, 716-723
- Auzins, J. (2004). Direct optimization of experimental designs, *Proceedings of 10th AIAA/ISSMO Conference*, AIAA 2004-4578, Albany, NY
- Bloedorn, E. & Michalski, R.S. (1998). Data-driven constructive induction. *Intelligent Systems*, 13, 2, 30-37, IEEE
- Breiman, L. (1996). Heuristics of instability and stabilization in model selection. *Annals of Statistics*, 24, 2350-2383
- Burnham, K.P. & Anderson, D.R. (2002). *Model selection and multimodel inference: a practical information-theoretic approach*, Springer-Verlag, NY
- Chen, V.C.P., Tsui, K-L., Barton, R.R. & Meckesheimer, M. (2006). A review on design, modeling and applications of computer eksperiments. *IIE Transactions*, 38, 4, 273-291
- Cherkassky, V. & Ma, Y. (2002). Selecting of the loss function for robust linear regression, *Neural Computation*
- Cherkassky, V. & Mulier, F.M. (2007). *Learning from Data: Concepts, Theory, and Methods*, 2nd ed., Wiley-IEEE Press
- Cleveland, W. & Devlin S. (1988). Locally weighted regression: an approach to regression analysis by local fitting. *American Statistical Association*, 83, 596-610
- Dreyfus, G. & Guyon, I. (2006). Assessment methods, In: *Feature Extraction: Foundations and Applications*, Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L.A. (Eds.), 65-88, Springer
- Duin, R.P.W. (2002). The combining classifier: to train or not to train?, *Proceedings of 16th International Conference on Pattern Recognition*, pp. 765-770
- Ferri, F., Pudil, P., Hatef, M. & Kittler, J. (1994). Comparative study of techniques for large-scale feature selection, In: *Pattern Recognition in Practice IV, Multiple Paradigms, Comparative Studies and Hybrid Systems*, Gelsema, E.S., Kanal, L.S. (Eds.), 403-413, Elsevier

- Friedman, J.H. (1993). *Fast MARS*, Tech. Report LCS110, Department of Statistics, Stanford University
- Friedman, J.H. (1994). An overview of predictive learning and function approximation, In: *From Statistics to Neural Networks*, Cherkassky, V., Friedman, J., Wechsler, H. (Eds.), Springer, NY
- Gutmann, H.-M. (2001). A radial basis function method for global optimization. *Journal of Global Optimization*, 19, 201-227
- Guyon, I. & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3, 1157-1182
- Hastie, T., Tibshirani, R. & Friedman J. (2003). *The Elements of Statistical Learning*, Springer
- Hurvich, C.M. & Tsai, C.-L. (1989). Regression and time series model selection in small samples. *Biometrika*, 76, 297-307
- Jain, A. & Zongker, D. (1997). Feature selection: evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19, 2, 153-158
- Jain, A.K., Duin, R.P.W. & Mao J. (2000). Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22, 1
- Jekabsons, G. (2008). Ensembling adaptively constructed polynomial regression models. *International Journal of Intelligent Systems and Technologies*, 3, 2, 56-61
- Jekabsons, G. & Lavendels, J. (2008a). Polynomial regression modelling using adaptive construction of basis functions, *Proceedings of IADIS International Conference, Applied Computing*, pp. 269-276, Mondragon unibertsitatea, April 2008, Algarve
- Jekabsons, G. & Lavendels, J. (2008b). A heuristic approach for surrogate modelling, *Proceedings of Applied Information and Communication Technologies*, pp. 11-20, April 2008, Jelgava
- Kalnins, K., Skukis, E. & Auzins, J. (2005). Metamodels for I-core and V-core sandwich panel optimization, In: *Shell Structures: Theory and Applications*, Pietraszkiewicz, W., Szymczak, C. (Eds.), 569-572, Taylor & Francis, London
- Kalnins, K., Eglitis, E., Jekabsons, G. & Rikards, R. (2008a). Metamodels for optimum design of laser welded sandwich structures, *Proceedings of Welded Structures, Design, Fabrication, and Economy*, pp. 119-126, April 2008, Miskolc
- Kalnins, K., Ozolins, O. & Jekabsons, G. (2008b). Metamodels in design of GFRP composite stiffened deck structure, *Proceedings of 7th ASMO-UK/ISSMO International Conference on Engineering Design Optimization*, July 2008, Bath
- Kalnins, K., Jekabsons, G. & Rikards, R. (2009a). Metamodels for optimisation of post-buckling responses in full-scale somposite structures, *Proceedings of 8th World Congress on Structural and Multidisciplinary Optimization*, June 2009, Lisbon
- Kalnins, K., Jekabsons, G., Zudrags, K. & Beitlers, R. (2009b). Metamodels in optimisation of plywood sandwich panels, In: *Shell Structures: Theory and Applications*, Pietraszkiewicz, W., Szymczak, C. (Eds.), Taylor & Francis, London (accepted)
- Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 1137-1145, Morgan Kaufmann, San Mateo, CA
- Kohavi, R. & John, G.H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, 97, 273-324
- Kolmogorov, A. & Fomin, S. (1975). *Introductory Real Analysis*, Dover Publications, NY

- Kotsiantis, S. & Pintelas, P. (2004). Combining Bagging and Boosting. *International Journal of Computational Intelligence*, 1, 324-333
- Kudo, M. & Sklansky, J. (2000). Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition*, 33, 1, 25-41
- Lophaven, S.N., Nielsen, H.B. & Sondergaard, J. (2002). *DACE – A Matlab Kriging Toolbox*, Tech. Report IMM-TR-2002-12, Informatics and Mathematical Modelling, Technical University of Denmark
- Loughrey, J. & Cunningham, P. (2004). Overfitting in wrapper-based feature subset selection: the harder you try the worse it gets, *Proceedings of 24th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pp. 33-43
- Martin, J.D. & Simpson, T.W. (2005). Use of Kriging models to approximate deterministic computer models. *AIAA Journal*, 43, 4, 853-863
- Molina, L.C., Belanche, L. & Nebot, A. (2002). Feature selection algorithms: a survey and experimental evaluation, *Proceedings of the International Conference on Data Mining*, pp. 306-313, IEEE Computer Society, Maebashi
- Myers, R.H. & Montgomery, D.C. (2002). *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, 2nd ed., John Wiley & Sons, NY
- Nikolaev, N.Y. & Iba H. (2006). *Adaptive Learning of Polynomial Networks*, Springer
- Opitz, D. & Maclin, R. (1999). Popular ensemble methods: an empirical study. *Journal of Artificial Intelligence Research*, 11, 169-198
- Orosz, E.S. & Anderson C.W. (1994). *Classification of EEG Signals Using a Sparse Polynomial Builder*, Tech. Report 94-111, Computer Science, Colorado State University
- Parmanto, B., Munro, P.W. & Doyle, H.R. (1996). Improving committee diagnosis with resampling techniques, In: *Advances in Neural Information Processing Systems*, Touretzky, D.S., Mozer, M.C., Hesselmo, M.E. (Eds.), 882-888, MIT Press, Cambridge, MA
- Pudil, P., Ferri, F.J., Novovicova, J. & Kittler, J. (1994). Floating search methods for feature selection with nonmonotonic criterion functions, *Proceedings of the International Conference on Pattern Recognition*, pp. 279-283, IEEE, Los Alamitos, CA
- Reunanen, J. (2003). Overfitting in making comparisons between variable selection methods. *Journal of Machine Learning Research*, 3, 371-382
- Reunanen, J. (2006). Search strategies, In: *Feature extraction: foundations and applications*, Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L.A. (Eds.), 119-137, Springer
- Ritthoff, O., Klinkenberg, R., Fischer, S. & Mierswa, I. (2002). A hybrid approach to feature selection and generation using an evolutionary algorithm, *Proceedings of the 2002 UK Workshop on Computational Intelligence*, pp. 147-154
- Russell, S.J. & Norvig, P. (2002). *Artificial intelligence: a modern approach*, 2nd ed., Prentice Hall, Englewood Cliffs, NJ
- Shevade, S.K., Keerthi, S.S., Bhattacharyya, C. & Murthy, K.R.K. (1999). Improvements to the SMO algorithm for SVM regression. *Transactions on Neural Networks*, IEEE
- Smola, A.J. & Scholkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, 14, 199-222
- Sutton, R.S. & Matheus, C.J. (1991). Learning polynomial functions by feature construction, *Proceedings of 8th International Workshop on Machine Learning*, June 1991, Chicago, IL

- Todorovski, L., Ljubic, P. & Dzeroski, S. (2004). Inducing polynomial equations for regression, *Proceedings of Fifteenth International Conference on Machine Learning*, pp. 441-452, Springer, Berlin
- Vapnik, V. (1995). *The Nature of Statistical Learning Theory*, Springer-Verlag, NY
- Witten, I.H. & Frank, E. (2005). *Data mining: practical machine learning tools and techniques with Java implementations*, 2nd ed., Morgan Kaufmann, SF
- Zongker, D., & Jain, A. (1996). Algorithms for feature selection: an evaluation. *Pattern Recognition*, 2, 18-22

On The Combination of Feature and Instance Selection

Jerffeson Teixeira de Souza, Rafael Augusto Ferreira do Carmo
and Gustavo Augusto Campos de Lima
Universidade Estadual do Ceará
Brazil

1. Introduction

In the last decades, huge amounts of data became omnipresent in diverse areas of knowledge, such as business, astronomy, biology, and so on. Machine Learning and Knowledge Discovery in Databases (KDD) are fields in Computer Science that focus on the task of transforming these data into useful knowledge. In (Fayyad et al., 1996), KDD is defined as "*the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data*". Feature and Instance Selection belong to the practice of data preparation (or pre-processing), which is a preliminary process that transforms raw data into a format that is convenient to the data mining (or machine learning) algorithm.

Usually, data is stored in a *table-like* format: the columns of these tables are the *attributes* or *features* - they describe the data - and the rows, or lines, are the *records* or *instances* - they are the examples of the concept stored in the data. Feature and Instance selection processes allow applications, such as classification or clusterization, to focus only on the important (or relevant) attributes and records to the specific concept that is in study.

As important machine learning problems, Feature and Instance Selection have been studied systematically over the last decades, when several algorithms for solving them individually have been proposed. Such selection problems play a fundamental role in the pre-processing step of any learning task. By removing noise, irrelevant and redundant features and instances, and reducing the overall dimensionality of a dataset, feature and instance selection have been demonstrated to improve the performance of most machine learning algorithms, speed up the output of models and allow algorithms to deal with datasets whose sizes are gigantic. Even though the specialized literature have exhibited remarkable results in solving both the feature and instance selection problems individually, little work has been done to manage these solutions to work together in order to solve these related problems simultaneously or even understand the relationship between features and instances.

This chapter initially discusses the feature and instance selection problems and their relevance to machine learning, giving an accurate definition of both problems. Next, it surveys different approaches for dealing with feature selection and instance selection separately and some works that tried to integrate the solutions for these two problems,

demonstrating the unexplored potential of such combination. Following the single and multi-objective models to these problems, it is presented and evaluated a metaheuristic-based framework for integrating the problems. Several experimental results demonstrate the interesting performance of the framework when compared to other standalone and combinational approaches over several natural datasets collected in the literature. Some conclusions and ideas for future works are given in the end of the chapter.

1.1 Problem's Definition

In this chapter we are going to use the following formalization when referring to *datasets*, *features*, and *evaluation functions*. Based in (John et al., 1994) "each instance X is an element on the set $F_1 \times F_2 \times \dots \times F_m$, where F_i is the domain of the i th feature". A dataset D is a set of tuples $\langle X, C \rangle$ where C is the class value of this example.

Given a classifier C and a dataset D , we define $G(C, D)$ as a function that measures the *error rate* of this classifier on this dataset D .

1.2 The Feature Selection Problem

The Feature Selection problem involves discovering a subset of features such that a classifier built only with this subset would have better predictive accuracy than a classifier built from the entire set of features. Other benefits of feature selection include a reduction in the amount of training data needed to induce an accurate classifier, that is consequently simpler and easier to understand, and a reduced execution time. In practice, feature selection algorithms will discover and select features of the data that are relevant to the task to be learned.

In addition to irrelevant features, feature selection researchers have identified other examples of problematic features which may have a negative impact on the performance of learning systems such as redundant features and randomly class-correlated features. Irrelevant features are those that do not contribute to the predictive accuracy of a particular target concept. Redundant features refer to those that, even when relevant to a target concept, provide mostly information already present in another feature and, in fact, do not contribute to getting better predictors. Randomly class-correlated features are correlated to the target class most of the time, and random otherwise. Thus, irrelevant, redundant and randomly class-correlated features are worthless and removing them can improve the learning process. In fact, the feature selection process can be seen alternatively as the process of identifying and removing as many irrelevant, redundant and randomly class-correlated features as possible.

Then we can formulate the problem of feature selection as:

$$\begin{aligned} & \text{Max } G \\ & \text{Subject to} \\ & |F'| \geq 1 \end{aligned} \tag{1}$$

A multiobjective version of it can seen as

$$\begin{aligned} & \text{Max } G, \text{Min } |F| \\ & \text{Subject to} \\ & |F'| \geq 1 \end{aligned} \tag{2}$$

Clearly a classifier built with a set of features $F' \subseteq F$ which is more accurate than one built with the whole set F is more interesting to use. Additionally the smaller it is the less computationally expensive it is. This characteristic is very important due to the datasets with high number of features found nowadays.

1.3 The Instance Selection Problem

The Instance Selection problem is basically the orthogonal version of the Feature Selection problem, as it involves discovering a subset of instances such that a classifier built only with this subset would have better predictive accuracy than a classifier built from the entire set of instances. In (Liu & Motoda, 2002), this problem is defined as "*to choose a subset of data to achieve the original purpose of a data mining application as if the whole data is used*". Clearly, instance selection cleans the dataset that is in use: it removes irrelevant examples, as well noisy and redundant ones. Instance Selection plays, consequently, two important roles: to improve computational efficiency, since the learning algorithm will consider only a subset of the original data, and to allow the induction of better classifiers (Blum & Langley, 1997).

Let's define a function $\text{Freq}(D^*, c)$ that calculates the frequency of the class c in the given dataset D^* . A Δ is a value in the interval $[0..1]$. Given these two definitions, the initial dataset D , a generic subset of it D' and I the set of instances in this dataset then we can formulate the problem of instance selection as

$$\begin{aligned} & \text{Max } G \\ \text{Subject to} \\ & \forall c \in C, \text{Freq}(D', c) - \Delta \leq \text{Freq}(D, c) \leq \text{Freq}(D', c) + \Delta \\ & |I'| \geq 1 \end{aligned} \quad (3)$$

A multiobjective version of it can seen as

$$\begin{aligned} & \text{Max } G, \text{Min } |I| \\ \text{Subject to} \\ & \forall c \in C, \text{Freq}(D', c) - \Delta \leq \text{Freq}(D, c) \leq \text{Freq}(D', c) + \Delta \\ & |I'| \geq 1 \end{aligned} \quad (4)$$

Like in the feature selection problem, a classifier built with a set of instances $I' \subseteq I$ which is more accurate than one built with the whole set I is more interesting to use. Additionally the smaller it is the less computationally expensive it is.

2. Related Works

Up to this date, several solutions have been proposed to deal with the feature and instance selection problems. In this section we briefly describe some important algorithms that work on each problem separately and show some approaches that handle both problems in a simultaneous way.

2.1 On Feature Selection

Well known feature selection algorithms perform very differently in identifying and removing irrelevant, redundant and randomly class-correlated features. Feature weighting

algorithms such as Relief (Kira & Rendell, 1992), for instance, usually cannot identify redundant features since they evaluate features individually, not in sets of features like other feature selection algorithms. However, they are often very efficient in estimating feature relevance. Relief also suffers with randomly class-correlated features. In (Dash & Liu, 1997), the authors report that Relief preferred a correlated feature rather than a relevant one in the Corral dataset. The Focus algorithm (Almuallim & Dietterich, 1991), on the other hand, deals really well with irrelevant, redundant and class-correlated features since it looks for subsets that generate no inconsistency. Unless the redundant or class-correlated features perfectly duplicate their pairs (another feature or class label, respectively), they will be eventually eliminated by Focus. The LVF algorithm (Liu & Setiono, 1996) presents a similar behavior, since it will search for more consistent subsets. For small datasets or given enough time, LVF tends to get rid of undesirable features. Other results regarding the ability of feature selection algorithms in dealing with these problematic features can be found in (Dash & Liu, 1997). In this paper, the authors report results of several well known selection algorithms over three datasets (Corral, Parity3+3 and Monk3) containing together irrelevant, redundant and randomly class-correlated features.

As for the use of metaheuristics for the Feature Selection problem, authors have tried different approaches. A Genetic Algorithm-based feature selector (GA) proposed in (Vafaie & De Jong, 1992) applies a simple genetic algorithm to search through the subsets of features. Other examples of applications of genetic algorithms for feature selection can be found in (Beritelli et al., 2005) and (Sun et al., 2002). In (Tahir et al., 2004), the authors report the use of Tabu Search to select attributes to improve the classification of prostate needle biopsies. The paper reports a reduction of 50% in the classification error rate due to the proposed approach. The Simulated Annealing metaheuristic was used in (Filippone et al., 2006) to develop the SAIS (Simulated Annealing Input Selection) algorithm. Good experimental results were reported when using several datasets, including two bioinformatics datasets.

In (Souza, 2004) the author describes and discusses dozens of feature selection algorithms and expands a framework proposed earlier by (Dash & Liu, 1997) which classifies several algorithms according to their generation procedure and evaluation criterion. All these algorithms can be classified into three broad categories: *filters*, *wrappers* and *hybrid approaches*. *Filters* are those algorithms which perform the selection of features using an evaluation measure that classify the “quality” of these elements to differentiate classes without making use of any machine learning algorithm. *Wrappers* explicitly make use of machine learning algorithms in order to perform this measurement. Usually, *filters* are much less computationally expensive than *wrappers* but they produce subsets with less quality than those produced by *wrappers*. *Hybrid approaches* combine the best characteristics of both approaches, trying to produce very good subsets efficiently.

2.2 On Instance Selection

Most of the works on instance selection have been based on Nearest Neighbor classification. In (Hart, 1968), the author proposed the Condensed Nearest Neighbor Rule (CNN), which finds a subset such that every member of the original dataset is closer to a member of the subset of the same class than to a member of the subset of a different class. This approach was extended in (Ritter et al., 1975) in the Selective Nearest Neighbor Rule (SNN) where every member of the original dataset must be closer to a member of the dataset of the same

class than to any member of the original dataset of a different class. The Reduced Nearest Neighbor Rule (RNN) was proposed in (Gates, 1972). It removes each instance if such a removal does not cause any other instances to be misclassified by the instances remaining. In (Cano et al., 2003), the authors describe and evaluate four evolutionary approaches, including genetic algorithms, for the instance selection problem and report better data reduction percentages and higher classification accuracy in the experimental evaluation when using these approaches. Another application of genetic algorithms can be found in (Ramirez-Cruz et al., 2006). A description and comparison of several instance selection algorithms can be found in (Jankowski & Grochowski, 2004).

Like feature selection algorithms, instance selection algorithms can be classified in those three broad categories.

2.3 On the Combination of Feature and Instance Selection

The most natural and straight-forward way to combine feature and instance selection is to perform one process after the other. In practice, that has been the way the two problems have been integrated. Let's consider as FSIS, the application of a feature selection process followed by an instance selection process, and ISFS the opposite. Since these approaches are general, in the sense that they can be applied to any domain, we will use them as comparison base in our experiments.

Besides this approaches, in (Fragoudis et al., 2002) the authors propose the FIS (Feature and Instance Selection) algorithm, which targets both problems simultaneously in the context of text classification. It considers a set of documents, classified in one of two classes C and C', which contain a group of words each and operates in two steps. In the first step, it searches for a subset of the original vocabulary that contains the words that are the best predictors of the given class C. Next, only the documents which contain at least one word from this subset are kept. The second step searches, similarly, on the resulting dataset for a subset of words that are the best predictors of class C'. The output of the algorithm FIS contain the two subsets of features over the resulting documents from the first step. The authors reported a great decrease in the number of feature and training instances. It terms of accuracy, the algorithm, using the Naive Bayes classifier, performed in some cases equally or more accurate them SVM.

Some works also use metaheuristics to solve these two problems. In (Souza et al., 2008) the authors use two simultaneous Simulated Annealing (Kirkpatrick et al., 1983) runs to solve each problem separately but use the actual solution of each process to calculate the quality of both of them. There has been made a lot of work using genetic and evolutionary algorithms. It is quite natural to design the solution to these two problems as a chromosome which is the vector of all feature and all instances and then run a genetic algorithm to solve these problems. In (Ramirez-Cruz et al., 2006) a simple approach that splits the chromosome in two areas, the one of features which are coded as real values in [0..1] to weight the features, and the area of instances which are coded as boolean values to select the instances is presented. In (Kuncheva & Jain, 1999) the authors use boolean value coding to select feature and instances. The objective function used is the composition of the precision of 1-nn plus a value that penalizes the cardinality of each set. In (Sierra et al., 2001) the authors apply an adaptation of genetic algorithms, called Estimation of Distribution Algorithm (EDA), to select instances and features in the problem of estimating the likelihood of cirrhotic patients to die in at most 6 months after the interventional treatment called

Transjugular Intrahepatic Portosystemic Shunt (TIPS). In (Chen et al., 2005) it is made a study using an explicit multi-objective design to the problems of feature and instance selection, in which the goal is to maximize the performance of the 1-nn classifier and minimize both the number of attributes and instances. In (Ros et al., 2007) the authors model the problem in a multi-objective approach and solve them by a two-phase genetic algorithm. In (Ishibuchi & Nakashima, 2000) the authors use a genetic algorithm which is biased to decrease the number of features selected, by giving a bigger probability to the changing that exclude features from the solutions.

3. A Framework for Simultaneous and Independent Feature and Instance Selection

A deeper look into the related works on the combination of feature and instance selection shows some points already addressed by these solutions and new questions. The first point is that the majority of the approaches which try to solve these problems in a broad generic formulation solve them by using specialized versions of genetic algorithms, trying to separate the chromosome into two different areas, one for features and one for instances, and applying separate operators to each area. These approaches arise from the natural easiness of modeling the solutions to these problems as chromosomes and the need to cope with these two different problems separately. The other point addressed is that either these approaches work on a specific field of supervised learning (e.g. text mining) or depend on a specific classifier (e.g. kNN). The reader may question “How can we use other metaheuristics beyond genetic algorithms to solve these two problems?” or “How could we build a general framework to with them simultaneously?”. This section tries to give answers to these questions.

Here we describe an extension of the work presented in (Souza et al., 2008) as a general metaheuristics-based framework for simultaneous and independent feature and instance selection. This framework is an effort to build an algorithm that can deal with both problems simultaneously, since these problems are clearly related to one another and the work made to select a subset of features can also be reused to select instances (and vice-versa) but they are also independent, meaning that the algorithms that solve one of these problems do not have to tackle the other one as well. The key idea here is to provide a *joint subset evaluation*, in which the quality of a subset of features depends on the quality of a subset of instances (and vice-versa). This means that although the search processes are independent, they are guided by this *joint evaluation function*, which gives what we call a “power of influence” of each solution of the separate problems over the other.

3.1 The Framework for Feature and Instance Selection

In order to make definitions clear, we must explain that this framework works with two different solutions for each problem: the *best* and the *actual* solution. The *best* solution is, as the name itself explains, the solution which achieved the best evaluation value so far in the search process. The *actual* solution is the solution generated at every iteration to be tested to see whether it is better than the *best* solution so far. In some metaheuristics, like Simulated Annealing, the process of search does not depend on the *best* solution, although this solution is stored, but in others like VNS (Mladenović & Hansen, 1997) the *best* solution is the one which guides the search.

When applied to feature and instance selection, search metaheuristics can be seen as wrappers, as they generate subsets (solutions), evaluate them using some classifier to test whether they are good solutions or not and guide the search process by this evaluation value achieved by each solution. This version of the framework works basically controlling this evaluation process of each subset generated by the search. The framework can be described as follows in figure 1

Framework for Mono-Objective Simultaneous and Independent Feature and Instance Selection

Input: Dataset dt, Feature Selection Algorithm fs, Instance Selection Algorithm is, Evaluation Function ef

Output: Dataset ndt

1. **While**(Has Iterations(fs) || Has Iterations(is))
2. **Do**
3. $fs_{ss} = \text{Next Solution}(fs, dt)$
4. $is_{ss} = \text{Next Solution}(is, dt)$
5. eval = Evaluation(ef, dt, fs_{ss} , is_{ss})
6. Update(fs, eval, fs_{ss})
7. Update(is, eval, is_{ss})
8. **Done**
9. ndt = Create Subset(dt, Best Subset(fs), Best Subset(is))
10. **Return** ndt

Fig. 1. The Framework

In this framework, the relationship between these entities, features and instances, is treated as something related to their quality to a supervised learning task. This means that the quality of features used in the supervised learning task is intrinsically related to the examples that represent the concept to be learned, and vice-versa. Examples are only considered “good” ones if they are described by attributes that represent the concept to be learned clearly, and features are only important if they capture this concept present in these examples. This is the justification to the presented approach.

A textual description of the framework can be seen as: Initially the complete sets of features and instances are set as initial solutions. There are two separated processes for selecting features and instances. The main loop started in the line number 1 controls the search processes. Starting in line number 3 (4) the new solution is generated using the metaheuristic for feature (instance) selection. New solutions are generated only when the *Has Iterations* test has *true* value, otherwise the *Next Solution* function must return the best solution found in the search process. In line 5 the new solutions are evaluated. This step is the *joint evaluation function* that works by getting the *actual solutions* from the feature selection and instance selection processes, then creating a subset from the initial dataset by using these two subsets and then evaluate them using *k-fold* cross validation, for example. Finally the search processes are updated. This update is basically the exchange of solutions if the new one achieved a better evaluation than the old one and any other process needed by the metaheuristic like for example in Simulated Annealing, when even if the actual solution is worse than the best one, it can be the next which guides the following steps of the search. In line 9 the whole procedure is ended, and the subset generated by the two solutions is returned as a new dataset to the supervised learning task. The figure 2 shows a

graphical representation of the framework. The blue boxes are the *best solutions* in a given iteration of the processes. The red boxes are the *actual solutions* in them and as it can be seen, they are evaluated together using the function G . In some iteration they replace the *best solution* but in other ones they do not have better evaluation so the best solutions remain the same.

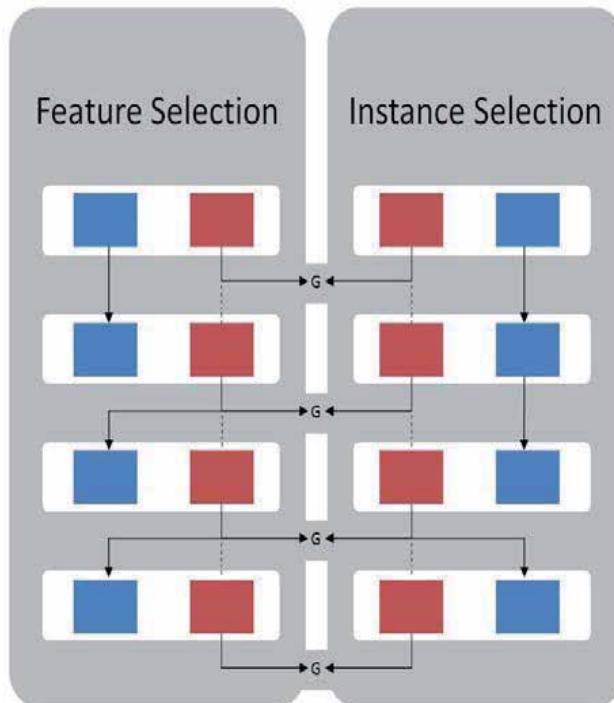


Fig. 2. A graphical view of the framework (Blue box – *best solution*; Red box – *actual solution*)

3.2 Extension to the Framework

The framework described in the last section is a basic view of it. An interesting extension can be made for handling populational metaheuristics.

Populational metaheuristics create, at each iteration, a set of new *actual solutions*. Then the evaluation of each new solution is calculated and operators of intensification and diversification are applied. If we remember the fact that in this framework the evaluation of a solution does not depend on itself solely, this fact adds the question of "Which solution from the other process should I use in the *joint evaluation function*?" or "How can I calculate the *best actual solution* in this given set of solutions?".

Our answers to these questions are quite simple. The answer to the first question is "the *best actual solution* from the last iteration". In the first iteration the whole set of features or instances is used to evaluate the new solutions and the search continues always reusing the best actual solution of the last iteration. By doing this, the searches are still guided by both solutions and only good solutions will guide this process. Nevertheless, the operators of *intensification* and *diversification* will work normally, without any loss to the search process.

The answer to the second question is “by using the *best actual* solution from the last iteration” as showed in the previous explanation.

Figure 3 gives a graphical explanation to the idea presented here. The reader must pay attention to the green arrows. They show that the *best actual* solution (the yellow one) is being used to evaluate the subsets of features (or instances) of the next generation. Besides, there are separate evaluation procedures to the searches. These are the biggest differences to the initial framework. Although there are these two separated procedures, the evaluations of the actual solutions still depend on the other search process. Once more as in the initial framework, in some iterations one of the solutions present in the actual population might replace the *best solution* found so far but in other ones they do not have better evaluation so the best solutions remains the same.

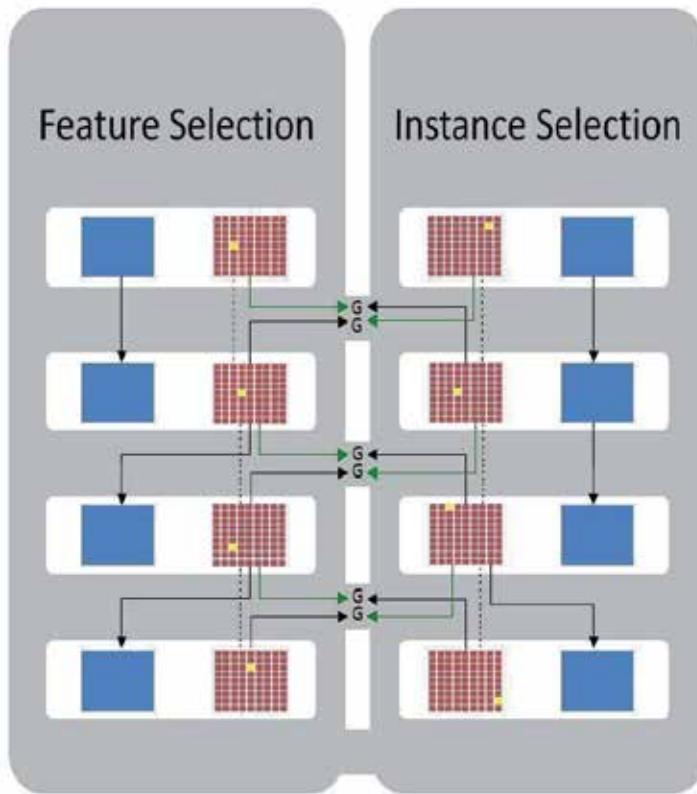


Fig. 3. A graphical view of the framework (Blue box – *best solution*; Red box – *actual solution*; Yellow box – *best actual solution*. Green arrow – The *best actual solution* is being used to evaluate the next generation of solutions)

4. Framework Evaluation

In this section we present and discuss the results obtained in several simulations executed in order to test the effectiveness of the proposed framework. This section tries to make clear the answer to the question “Is it worth using this framework?”.

To make these simulations we have chosen some well-known datasets used for machine learning tasks found at the UCI Machine Learning Repository (Asuncion and Newman, 2007). These datasets are the Audiology (70 attributes, 226 instances), Autos (26, 205), Colic (23, 368), Credit (16, 690), Ionosphere (35, 351), Labor (17, 57), Lymph (19, 148), Primary-Tumor (18, 339), Sonar (61, 208), Soybean (36, 683) and Vote (17, 435).

We implemented seven different strategies to tackle with the feature and instance selection problems. The first one, here called **ind**, consists in making two separate selection processes and then joining the subsets generated by these processes in the end. The solution generated by the feature selection process and the other generated by the instance selection one are joined to create the subset only when the search processes are ended. The **fsis** is a sequential approach in which it is run a feature selection process followed by an instance selection process. The dataset used in the feature selection is the whole initial dataset, but in the dataset used by the instance selection process, only the best set of features found is used to represent the examples. The **isfs** approach follows the same idea, but now the first process is an instance selection and the second is a feature selection. Finally **comb** is the name given to the approach presented in the framework.

Some pieces of different software were used to make these simulations. From the Weka Data Mining Software (Witten & Frank, 2005) we used several classes to represent datasets, attributes and examples and to create and evaluate models. From jMetal Metaheuristics Framework (Durillo et al., 2006) we used some classes to represent the solutions to these problems and also some classes of metaheuristics. The Evaluation method chosen to be used in these simulations was a 10-fold cross-validation. The classifiers used were the C4.5, Naive Bayes and kNN.

4.1 Simulation Using the Simulated Annealing Metaheuristic

The results presented in this section are the same presented in the former work of (Souza et al., 2008). The architecture implemented in that work is the same of this general framework but it was implemented using the Simulated Annealing metaheuristic.

For this simulation we implemented the Simulated Annealing metaheuristic to use it in both selection problems. Simulated Annealing is a metaheuristic that consists in a randomized local search, which simulates the process of physical annealing. This physical process consists in heating a material to a desired temperature, followed by a slow cooling process. The first step gives energy to the atoms and they move randomly through states of high energy, changing the material's structure fast. The second step, which is performed slowly, gives them the chance to arrange themselves into a configuration of lower energy.

In analogy with the physical process, Simulated Annealing changes the actual solution to a neighbor solution, depending on the quality of this neighbor solution or the value of a function that is calculated in accordance with the temperature parameter, which decreases during the process.

The coding of solutions to this problem is basically an array of boolean values which has length equal to the number of features or instances. Using this coding, we defined that two solutions are considered *neighbors* only and if only they have at most 10% of bits set to different values, i.e., when applied a XOR operator to these to problems, the result contains only 10% of bits set to *true*.

	< 0.001	< 0.005	< 0.01
Comb vs IND	8 x 0	1 x 0	1 x 0
Comb vs FSIS	0 x 0	0 x 0	0 x 0
Comb vs ISFS	0 x 0	0 x 0	2 x 0

Table 1. Pairwise comparison between **comb** and other approaches

We have run all seven approaches described earlier in two different scenarios. In the first one, each approach was given an unlimited time to run and generate a solution. After two executions of each approach in every dataset, there were twenty *Error Rate* values available.

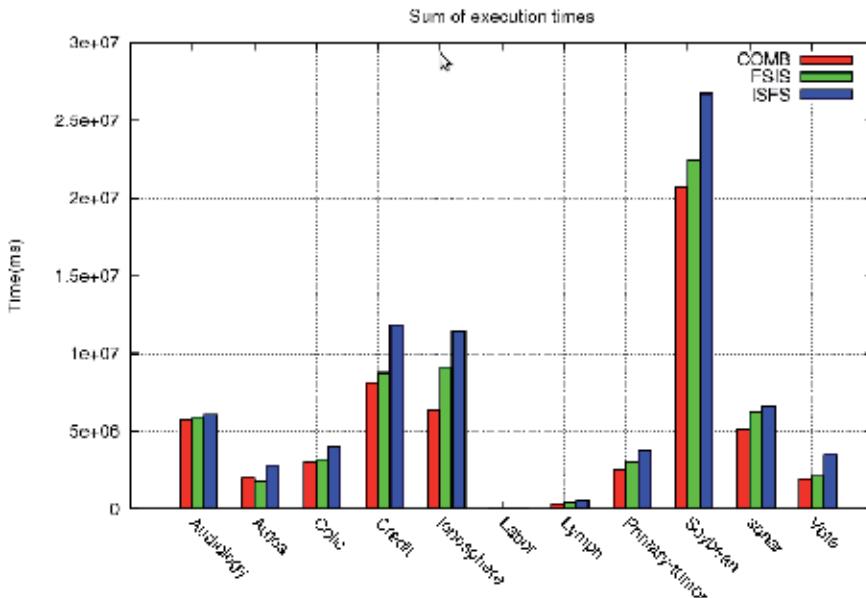


Fig. 4. The sum of execution times – Unlimited Time Scenario

Table 1 summarizes the results of a pairwise comparison between **comb** and the other approaches that solve both problems. The results represent the number of times each strategy outperformed the other, in terms of the accuracy of the final classifier, using the student's t-test with the corresponding confidence levels (0.001, 0.005 and 0.01).

Clearly the performance of **comb** is much better than the **ind** and slightly better than the other two approaches. So when talking about performance it is not clear why to use this approach. But looking at figure 3 we see that the **comb** approach usually requires less time to reach the best error rate in the datasets. This figure shows the sum of time of all tests executed by each approach.

In the second scenario we defined a limit to the execution time of every run. Figure 4 shows that when this constraint is added to the problem and this time is not enough to complete the search, the **comb** approach converges to low values of error rate faster than the other two approaches.

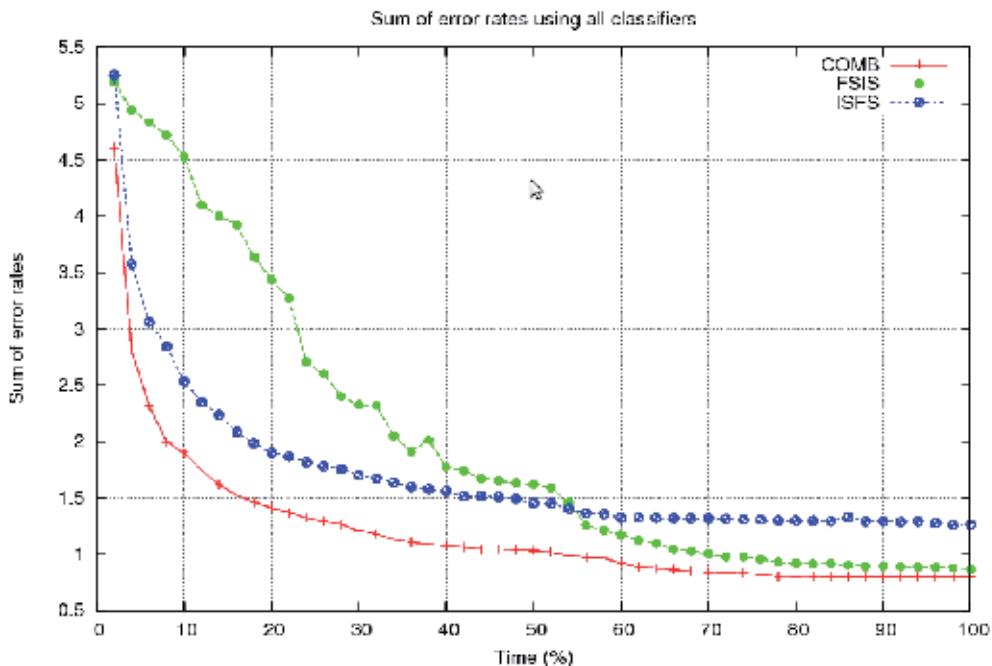


Fig. 5. The sum of error rates – Limited Time Scenario

5. Conclusion

In this chapter we discussed two important problems in the pre-processing step of many supervised learning tasks. A list of well-known algorithms were presented and discussed. A new framework was proposed, extending the concept proposed by the authors in a previous work. This framework was validated by some simulations using the metaheuristic Simulated Annealing and NSGA-II. These simulations show that although the quality of solutions generated by this framework is quite similar to those obtained by sequential executions, this approach reaches the better solutions faster than the other approaches.

The framework is based on what we called “*power of influence*”, i.e. the quality of features in a given supervised learning task is intrinsically related to the quality of instances used in this task, and vice-versa. Based on this we created the framework that work with two separated *wrappers* for these two problems, jointing them in a single evaluation procedure.

5.1 Future Work - The Framework for Multi-Objective Feature and Instance Selection

An important characteristic we want to add to this framework in the future is the possibility to handle the multi-objective versions of the two selection problems. The usage of multi objectives brings new power but also new problems to the search processes. In these formulations, the characteristic of *total ordering* is replaced by *partial ordering*, using the concept of *Pareto optimality*. The ideas of *better* and *worse* are replaced by *dominance*, *non-dominance*. Given two solutions a, b and a set of functions F to be minimized (or maximized, but in this explanation we suppose they are to be minimized), we say that a weakly dominates b if and only if

$$\forall f_i \in F, f_i(a) \leq f_i(b) \text{ and } \exists f_i \in F, f_i(a) < f_i(b) \quad (5)$$

The concept of *strong* dominance requires that

$$\forall f_i \in F, f_i(a) < f_i(b). \quad (6)$$

When there is a set of solutions in which none of them dominate or are dominated by the others, we say these solutions are in the *Pareto front*, i.e., they are solutions equally good, in a way that one cannot say *à priori* which one of them is the best one without making any other assumption.

This usage adds the same questions generated by populational metaheuristics, such as "Which solution from the other process should I use in the *joint evaluation function*?" or "How can I calculate the *best actual* solution in this given set of solutions if there will be some 'equally good' ones?".

The answers related to the multi-objective approaches seem to be similar to the ones given to populational metaheuristics but they weren't tested yet. Given that it is needed to evaluate all the subsets of features (and vice-versa), the algorithm can use any of the subsets of instances from the last iteration which are in the *Pareto front* since all of them are equally good. A reasonable solution would be to pick a random solution from the *Pareto front* of the instance selection process every time the algorithm has to evaluate a subset of features. This approach increases *diversification* because several different solutions are used to guide the search and there is no loss in *intensification* as only good solutions are used in this process.

In the end of the search processes there will be two *Pareto fronts*: one of features and one of instances. At this moment the user have several alternatives like choosing one solution of each search or generating all combinations of solutions and picking the one which is the best. How to deal with these two *Pareto fronts* is an open question so far.

6. References

- Almuallim, H. & Dietterich, T. (1991). Learning with many irrelevant features. *Proceedings of the Ninth National Conference on Artificial Intelligence* (AAAI'91), pp. 547-552, AAAI Press, Anaheim
- Asuncion, A. & Newman, D. J. (2007). *UCI Machine Learning Repository* [<http://www.ics.uci.edu/~mlearn/MLRepository.html>], University of California, School of Information and Computer Science, Irvine, CA
- Beritelli, F.; Casale, S.; Russo, A. & Serrano, S. (2005). A genetic algorithm feature selection approach to robust classification between "positive" and "negative" emotional states in speakers. *Thirty-Ninth Asilomar Conference on Signals, Systems and Computers*, pp. 550-553
- Blum, A. & Langley, P. (1997). Selection of Relevant Features and Examples in Machine Learning. *Artificial Intelligence*, 97, 1-2, December 1997, 245-271, 0004-3702
- Cano, J.; Herrera, F. & Lozano, M. (2003). Using evolutionary algorithms as instance selection for data reduction in kdd: an experimental study. *IEEE Transactions on Evolutionary Computation*, 7, 6, December 2003, 561-575, 1089-778X

- Chen, J-H.; Chen, H-M. & Ho S-Y. (2005). Design of nearest neighbor classifiers: multi-objective approach, *International Journal of Approximate Reasoning*, 40, 1, July 2005, 3-22, 0888-613X
- Dash, M. & Liu, H. (1997). Feature selection for classification. *Intelligent Data Analysis – An International Journal*, 1, 131-156
- Durillo, J. J.; Nebro, A. J.; Luna, F.; Dorronsoro, B. & Alba, E. (2006). *TechRep jMetal: a Java Framework for Developing Multi-Objective Optimization Metaheuristics.*, Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, 2006
- Fayyad, U.; Piatetsky-Shapiro, G. & Smyth, P. (1996). From data mining to knowledge discovery in databases. *Ai Magazine*, 17, 3, 37-54, 0738-4602
- Filippone, M.; Masulli, F.; Rovetta, S. & Constantinescu, S. P. (2006). Input selection with mixed data sets: A simulated annealing wrapper approach. *Conferenza Italiana Sistemi Intelligenti (CISI' 06)*, Ancona
- Fragoudis, D.; Meretakis, D. & Likothanassis, S. (2002). Integrating feature and instance selection for text classification. *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 501-506, 1-58113-567-X, ACM, New York
- Gates, G. (1972). The reduced nearest neighbor rule (corresp.). *IEEE Transactions on Information Theory*, 18, 3, May 1972, 431-433, 0018-9448
- Hart, P. (1968). The condensed nearest neighbor rule, *IEEE Transactions on Information Theory*, 14, 3, May 1968, 515-516, 0018-9448
- Ishibuchi, H. & Nakashima, T. (2000). Multi-objective pattern and feature selection by a genetic algorithm, *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1069-1076, Morgan Kaufmann
- Jankowski, N. & Grochowski, M. (2004). Comparison of instances selection algorithms I. Algorithms Survey, In: *Artificial Intelligence and Soft Computing - ICAISC 2004*, Springer, 978-3-540-22123-4
- John, G.; Kohavi, R. & Pfleger, K. (1994). Irrelevant features and the subset selection problem. *Proceedings of the Eleventh International Conference on Machine Learning (ICML'94)*, 121-129
- Kira, K. & Rendell, L. (1992). The feature selection problem: Traditional methods and new algorithm, *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 129-134, MIT Press
- Kirkpatrick, S.; Gelatt, C. D. & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220, 4598, May 1983, 671-680, 00368075
- Kuncheva, L. I. & Jain, L. C. (1999). Nearest neighbor classifier: simultaneous editing and feature selection, *Pattern Recognition Letters*, 20, 11, November 1999, 1149-1156, 0167-8655
- Liu, H. & Motoda, H. (2002). On issues of instance selection. *Data Mining and Knowledge Discovery*, 6, 2, 115-130, 1384-5810
- Liu, H. & Setiono, R. (1996). A probabilistic approach to feature selection – A filter solution. *Proceedings of the Thirteenth International Conference on Machine Learning (ICML'96)*, pp. 319-327, MIT Press
- Mladenović, N. & Hansen, P. (1997). Variable neighborhood search. *Computers and Operations Research*, 24, 11, November 1997, 1097-1100, 0305-0548

- Ramirez-Cruz, J. F. ; Fuentes, O.; Alarcon-Aquino, V. & Garcia-Banuelos, L. (2006). Instance selection and feature weighting using evolutionary algorithms. *15th International Conference on Computing (CIC '06)*, 73-79, 0-7695-2708-6
- Ritter, G. ; Woodruff, H.; Lowry, S. & Isenhour, T. (1975). An algorithm for a selective nearest neighbor decision rule (corresp.). *IEEE Transactions on Information Theory*, , 21, 6, November 1975, 665-669, 0018-9448
- Ros, F. ; Guillaume, S. ; Pintore, M. & Chrétien, J. R. (2007). Hybrid genetic algorithm for dual selection, *Pattern Analysis & Applications*, 11, 2, June 2008, 179-198, 1433-755X
- Sierra, B. ; Lazkano, E. ; Inza, I. ; Merino, M. ; Larrañaga, P. & Quiroga, J. (2001), *Lecture Notes in Computer Science*, 2101/2001, 20-29, 978-3-540-42294-5
- Souza, J. T. (2004). Feature selection with a general hybrid algorithm. *Doctoral dissertation, University of Ottawa, School of Information Technology and Engineering (SITE)*, Ottawa
- Souza, J. T. ; Carmo, R. A. F. & Campos, G. A. L. (2008). A novel approach for integrating feature and instance selection. *Proceedings of the International Conference on Machine Learning and Cybernetics*, pp. 374-379, 978-1-4244-2095-7, July 2008
- Sun, Z.; Bebis, G.; Yuan, X. & Louis, S. J. (2002). Genetic feature subset selection for gender classification: A comparison study. *Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision (WACV'02)*, IEEE Computer Society, Washington
- Tahir, M.; Bouridane, A.; Kurugollu, F. & Amira, A. (2004). Feature selection using tabu search for improving the classification rate prostate needle biopsies. *Pattern recognition*, 2, 335-338, 0031-3203
- Vafaie, H. & De Jong, K. (1992). Genetic algorithms as a tool for feature selection in machine learning. *Proceedings of the Fourth International Conference on Tools with Artificial Intelligence*, pp. 200-204, Arlington
- Witten, I. H. & Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*, Morgan Kaufmann, San Francisco, 2005

Fuzzy System with Positive and Negative Rules

Thanh Minh Nguyen and Q. M. Jonathan Wu

*Department of Electrical and Computer Engineering, University of Windsor
Canada*

1. Introduction

A typical rule in the rule base of a traditional fuzzy system contains only positive rules (weight is positive). In this case, mining algorithms only search for positive associations like "IF A Then do B", while negative associations such as "IF A Then do not do B" are ignored. The concept of fuzzy sets was introduced by Zadeh in 1965 as a mathematical tool able to model the partial memberships. Since then, fuzzy set theory (Zadeh, 1973) has found a promising field of application in the domain of image processing, as fuzziness is an intrinsic property of images and the natural outcome of many image processing techniques. The interest in using fuzzy rule-based models arises from the fact that they provide a good platform to deal with noisy, imprecise or incomplete information which is often handled exquisitely by the human-cognition system.

In a fuzzy system, we can generate fuzzy rule-bases of one of the following three types:

(a) Fuzzy rules with a class in the consequent (Abe & Thawonmas, 1997; Gonzalez & Perez, 1998). This kind of rule has the following structure:

$$\text{Rule } r : \text{IF } x_1 \text{ is } A_{r1} \text{ and } \dots \text{ and } x_N \text{ is } A_{rN} \text{ Then } y \text{ is class } C_m \quad (1)$$

Where, $\mathbf{x} = (x_1, \dots, x_N)$ is an N -dimensional pattern. A_m , $n = (1, 2, \dots, N)$, is an antecedent fuzzy set, and y is the class C_m to which the pattern belongs.

(b) Fuzzy rules with a class and a rule weight in the consequent (Ishibuchi et al., 1992; Ishibuchi & Nakashima, 2001):

$$\text{Rule } r : \text{IF } x_1 \text{ is } A_{r1} \text{ and } \dots \text{ and } x_N \text{ is } A_{rN} \text{ Then } y \text{ is class } C_m \text{ with } W_r \quad (2)$$

Where, W_r is the rule weight which is a real number in the unit interval $[0, 1]$.

(c) Fuzzy rules with rule weight for all classes in the consequent (Pal & Mandai, 1992; Mandai & Murthy, 1992; Ishibuchi & Yamamoto, 2005):

$$\begin{aligned} \text{Rule } r : \text{IF } x_1 \text{ is } A_{r1} \text{ and } \dots \text{ and } x_N \text{ is } A_{rN} \\ \text{Then } y \text{ is class } C_1 \text{ with } W_{r1} \text{ and } \dots \text{ and } y \text{ is class } C_M \text{ with } W_{rM} \end{aligned} \quad (3)$$

Where, W_{rm} , $m = (1, 2, \dots, M)$, is a rule weight for class C_m .

From Eq.(1), Eq.(2) and Eq.(3), we can see that a typical rule in the rule-base of a fuzzy system contains only positive rules (weight is positive). This is one of the limitations of a

traditional association mining algorithm (Han, 2006). In this case, mining algorithms only search for positive associations like "IF A Then do *B*", while negative associations such as "IF A Then do not do *B*" are ignored. In addition to the positive rules, negative rules (weight is negative) can provide valuable information. For example, the negative rule can guide the system away from situations to be avoided, and after avoiding these areas, the positive rules once again take over and direct the process. Interestingly, very few papers have focused on negative association rules due to the difficulty in discovering these rules. Although some researchers point out the importance of negative associations (Brin & Silverstein, 1997), only few groups (Savasere et al., 1998; Wu et al., 2002; Teng et al., 2002) have proposed a system to mine these types of associations. This not only indicates the novelty in the usage of negative association rules, but also the challenges in discovering them.

In this chapter, we propose a new fuzzy rule-based system for application in image classification problems. A significant advantage of the proposed system is that each fuzzy rule can be represented by more than one class. Moreover, while traditional fuzzy systems consider positive fuzzy rules only, in this chapter, we focus on combining negative fuzzy rules with traditional positive ones, leading to fuzzy inference systems. This new approach has been tested on image classification problems with promising results.

2. Positive and Negative Association Rules

Fuzzy systems can be broadly categorized into two families. The first includes linguistic models based on a collection of fuzzy rules, whose antecedents and consequents utilize fuzzy values. The Mamdani model (Mamdani et al., 1975) falls into this group. The second category, based on Sugeno-type systems (Takagi & Sugeno, 1985), uses a rule structure that has fuzzy antecedents and functional consequent parts. A typical rule in the rule-base of a fuzzy system is of the "IF-Then" type, i.e., "IF *A* then do *B*", where *A* is the premise of the rule and *B* is the consequent of the rule. This type of rule is called *positive rule* (weight is positive) because the consequent prescribes something that should be done, or an action to be taken. Another type of reasoning that has not been exploited much, involves *negative rules* (weight is negative), which prescribe actions to be avoided. Thus, in addition to the positive rules, it is possible to augment the rule-base with rules of the form, "IF *A*, Then do not do *B*". Let us consider the following two fuzzy IF-Then rules:

$$\begin{aligned} \text{Rule 1 : IF } &\textit{customer} \text{ is a } \underline{\textit{child}} \text{ Then } \underline{\textit{he buys Coke}} \text{ and } \underline{\textit{he does not buy bottled water}} \\ \text{Rule 2 : IF } &\textit{customer} \text{ is an } \underline{\textit{adult}} \text{ Then } \underline{\textit{he buys Coke}} \text{ and } \underline{\textit{he buys bottled water}} \end{aligned} \quad (4)$$

In the example above, the negative rule (rule 1) guides the system away from situations to be avoided, after which, the positive rules (rule 2) take over and direct the process. Depending on the probability of such an association, marketing personnel can develop better planning of the shelf space in the store, or can base their discount strategies on correlations that can be found in the data itself. In some situations (Branson & Lilly, 1999; Branson & Lilly, 2001; Lilly, 2007), a combination of positive and negative rules can form a more efficient fuzzy system.

One of the limitations of fuzzy IF-Then rules in Eq.(4) is that the two classes (Coke, bottled water) appearing in the consequent parts of the above rules have the same degree of importance. Clearly, to help the marketing personnel develop better planning of different

products (Coke, bottled water) for different customers (child, adult), we should assign different weights to different classes appearing in the consequent parts of the rules.

Based on these considerations, we propose a new adaptive fuzzy system that applies to the image classification problem (Thanh & Jonathan, 2008). The main advantage of this fuzzy model is that every fuzzy rule in its rule-base can describe more than one class. Moreover, it combines both positive and negative rules in its structure. This approach is expressed by:

$$\begin{aligned} \text{Rule } r : \text{ IF } x_{1k} \text{ is } A_{r1} \text{ and } \dots \text{ and } x_{Nk} \text{ is } A_{rN} \\ \text{Then } y_{k1} \text{ is class } C_1 \text{ with } W_{r1} \text{ and } \dots \text{ and } y_{kM} \text{ is class } C_M \text{ with } W_{rM} \end{aligned} \quad (5)$$

Where, W_{rm} , $r=(1,2,\dots,R)$, $m=(1,2,\dots,M)$ is the weight of each class belonging to the rule r . We use the rule weight of the form below:

$$W_{rm} = w_{rm0} + w_{rm1}x_{1k} + \dots + w_{rmN}x_{Nk} \quad (6)$$

Where, parameters w_{rml} , $l=(0,1,\dots,N)$ are determined by the least squares estimator, which is discussed in detail, in the following section. R , M , K , and N denote the number of fuzzy rules, number of classes, number of patterns and dimension of patterns, respectively. Classes are denoted by C_1, C_2, \dots, C_M , and the N -dimensional pattern is denoted by $\mathbf{x}_k = (x_{1k}, x_{2k}, \dots, x_{Nk})$, $k=(1,2,\dots,K)$.

Consider a multiple-input, multiple-output (MIMO) fuzzy system in Eq.(5), similar to Takagi-Sugeno fuzzy models (Takagi & Sugeno, 1985; Purwar et al., 2005). The m -th output of the MIMO with product inference, centroid defuzzifier and Bell membership functions is given by:

$$y_{km} = \frac{\sum_{r=1}^R \prod_{n=1}^N A_{rn}(x_{nk}) W_{rm}}{\sum_{r=1}^R \prod_{n=1}^N A_{rn}(x_{nk})} = \frac{\sum_{r=1}^R \beta_r(\mathbf{x}_k) W_{rm}}{\sum_{r=1}^R \beta_r(\mathbf{x}_k)} = \frac{\sum_{r=1}^R \beta_r(\mathbf{x}_k) W_{rm}}{\sum_{r=1}^R \beta_r(\mathbf{x}_k)} ; m = 1, \dots, M \quad (7)$$

Where the normalization degree of activation of the r -th rule $\bar{\beta}_r(\mathbf{x}_k)$ is expressed by:

$$\bar{\beta}_r(\mathbf{x}_k) = \frac{\beta_r(\mathbf{x}_k)}{\sum_{r=1}^R \beta_r(\mathbf{x}_k)} ; \quad \beta_r(\mathbf{x}_k) = \prod_{n=1}^N A_{rn}(x_{nk}) \quad (8)$$

The fuzzy set $A_{rn}(x_{nk})$ and the corresponding rule weight W_{rm} is discussed in detail in the following section. The output of the classifier is determined by the winner-take-all strategy shown in Eq.(9), whereby " \mathbf{x}_k will belong to the class with the highest activation".

$$y_k = C_m^* ; \quad m^* = \max_{1 \leq m \leq M} (y_{km}) \quad (9)$$

3. Structure of the proposed fuzzy system

So far, our discussion has focused on class estimation in Eq.(9) to which class the pattern \mathbf{x}_k should be assigned. In this section, we suggest a new adaptive fuzzy system that can automatically adjust the values of fuzzy set $A_{rn}(x_{nk})$ and rule weight W_{rm} . After training the fuzzy system, we can determine which class the pattern \mathbf{x}_k should be assigned to.

The proposed structure consists of two visible layers (input and output layer) and three hidden layers as shown in Fig. 1. This fuzzy system can be expressed as a directed graph corresponding to Eq.(7).

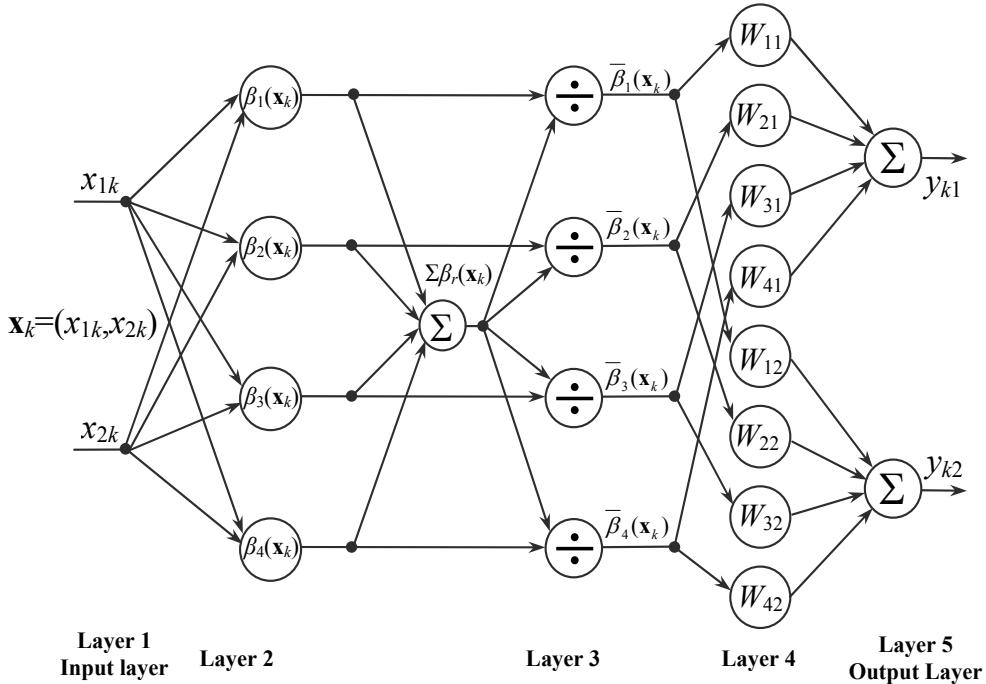


Fig. 1. Proposed fuzzy system with 2 inputs ($N=2$), 2 classes ($M=2$) and 4 rules ($R=4$).

Layer 1 (Input layer): each node in this layer only transmits input x_{nk} , $n=(1,2,\dots,N)$, $k=(1,2,\dots,K)$ directly to the next layer. No computation is performed in this layer. There are a total of N nodes in this layer, where the output of each node is $O_{1n} = x_{nk}$.

Layer 2: The number of nodes in this layer is equal to the number of fuzzy rules. Each node in this layer has N inputs from N nodes of the input layer, and feeds its output to the corresponding node of layer 3.

One of the major disadvantages of Anfis (Jang et al., 1997) model is, that an explosion in the number of inference rules limits the number of possible inputs. Thus, grid partitioning is not advised when the input dimension is more than six (Nayak et al., 2004). To overcome this problem, a fuzzy scatter partition is used in this layer. Therefore, our system can work well, even when the dimension of pattern (N) is high.

We use the bell type distribution defined over an N -dimensional pattern \mathbf{x}_k for each node in this layer. The degree of activation of the r -th rule $\beta_r(\mathbf{x}_k)$ with the antecedent part $\mathbf{A}_r = (A_{r1}, \dots, A_{rN})$ is expressed as follows:

$$\beta_r(\mathbf{x}_k) = \prod_{n=1}^N A_{rn}(x_{nk}) = \prod_{n=1}^N \frac{1}{1 + \left(\frac{x_{nk} - c_{rn}}{a_{rn}} \right)^{2b_{rn}}} \quad (10)$$

Where, parameters a_{rn} , b_{rn} , c_{rn} , $r=(1,2,\dots,R)$, $n=(1,2,\dots,N)$ are constants that characterize the value of $\beta_r(\mathbf{x}_k)$. The optimal values of these parameters are determined by training, which is discussed in the next section. There are R distribution nodes in this layer, where each node has $3 \times N$ parameters. The output of each node in this layer is $O_{2r} = \beta_r(\mathbf{x}_k)$.

Layer 3: This layer performs the normalization operation. The output of each node in this layer is represented by:

$$O_{3r} = \bar{\beta}_r(\mathbf{x}_k) = \frac{\beta_r(\mathbf{x}_k)}{\sum_{r=1}^R \beta_r(\mathbf{x}_k)} \quad (11)$$

Layer 4: Each node of this layer represents the rule weight in Eq.(6), $W_{rm} = w_{rm0} + w_{rm1}x_{1k} + \dots + w_{rmNxNk}$. Where, parameters w_{rml} , $r=(1,2,\dots,R)$, $m=(1,2,\dots,M)$, $l=(0,1,\dots,N)$ are determined by least squares estimator, which is discussed in the next section.

In the proposed model, for pattern \mathbf{x}_k , the output of the classifier is determined by the winner-take-all strategy. Therefore, when the rule weight W_{rm} has a negative value, it will narrow the choices for class C_m (the higher the negative value of W_{rm} , the smaller the value of y_{km} in Eq.(13)). In other words, negative rule weight prescribes actions to be avoided rather than performed. The output of each node in this layer is:

$$O_{4rm} = W_{rm} O_{3r} = W_{rm} \frac{\beta_r(\mathbf{x}_k)}{\sum_{r=1}^R \beta_r(\mathbf{x}_k)} \quad (12)$$

There are $M \times R$ nodes in this layer, where each node has $(1+N)$ parameters.

Layer 5 (Output layer): Each node in the output layer determines the value of y_{km} in Eq.(7).

$$O_{5m} = y_{km} = \sum_{r=1}^R O_{4rm} = \sum_{r=1}^R \frac{W_{rm} \beta_r(\mathbf{x}_k)}{\sum_{r=1}^R \beta_r(\mathbf{x}_k)} = \sum_{r=1}^R \bar{\beta}_r(\mathbf{x}_k) W_{rm} \quad (13)$$

There are M nodes in the output layer.

4. Parameter Learning

The goal of the work presented here is perform the parameterized learning to minimize the sum-squared error with respect to the parameters $\Theta = [a_{rn}, b_{rn}, c_{rn}, w_{rml}]$. The objective function $E(\Theta)$ for all the training data-sets is defined as:

$$E(\Theta) = \frac{1}{2KM} \sum_{k=1}^K \sum_{m=1}^M (y_{km} - y_{dkm})^2 \quad (14)$$

Where, y_{km} is the output of class m obtained from Eq.(7).

For a training data pair, $\{\mathbf{x}_k, \mathbf{y}_{dk}\}$, the input is $\mathbf{x}_k = (x_{1k}, x_{2k}, \dots, x_{Nk})$, $k = (1, 2, \dots, K)$, and the desired output \mathbf{y}_{dk} is of the form:

$$\mathbf{y}_{dk} = (y_{dk1}, y_{dk2}, \dots, y_{dKM})^T = \begin{cases} (1, 0, \dots, 0)^T, & \text{if } \mathbf{x}_k \in \text{class } C_1 \\ (0, 1, \dots, 0)^T, & \text{if } \mathbf{x}_k \in \text{class } C_2 \\ \dots \\ (0, 0, \dots, 1)^T, & \text{if } \mathbf{x}_k \in \text{class } C_M \end{cases} \quad (15)$$

When the initial structure has been identified with N inputs, R rules and M classes, the fuzzy system then performs the parameter identification to tune the parameters of the existing structure. To minimize the sum-squared error $E(\Theta)$, a two-phased hybrid parameter learning algorithm (Jang et al., 1997; Wang et al., 1999; Wang & George Lee, 2002; Lee & Lin, 2004) is applied with a given network structure. In hybrid learning, each iteration is composed of a forward and backward pass. In the forward pass, after the input pattern is presented, we calculate the node outputs in the network layers. In this step, the parameters a_{rn} , b_{rn} , and c_{rn} in layer 2 are fixed. The parameters w_{rml} in layer 4 are identified by least squares estimator. In the backward pass, the error signal propagates from the output towards the input nodes. In this step, the w_{rml} are fixed, and the error signals are propagated backward to update the a_{rn} , b_{rn} and c_{rn} by steepest descent method. This process is repeated many times until the system converges.

Next, optimization of the parameters w_{rml} in layer 4 is performed using least-squares algorithm in the forward step. To minimize the error $E(\Theta)$ in Eq.(14), we have to minimize each output-error (m -th output):

$$E_m = \sum_{k=1}^K (y_{km} - y_{dkm})^2 \quad (16)$$

When the training pattern \mathbf{x}_k is fed into the fuzzy system, Eq.(13) can be written as:

$$\begin{aligned} y_{km} = & \bar{\beta}_1(\mathbf{x}_k)w_{1m0} + \bar{\beta}_1(\mathbf{x}_k)w_{1m1}x_{1k} + \dots + \bar{\beta}_1(\mathbf{x}_k)w_{1mN}x_{Nk} + \\ & \bar{\beta}_2(\mathbf{x}_k)w_{2m0} + \bar{\beta}_2(\mathbf{x}_k)w_{2m1}x_{1k} + \dots + \bar{\beta}_2(\mathbf{x}_k)w_{2mN}x_{Nk} + \\ & \dots \\ & \bar{\beta}_R(\mathbf{x}_k)w_{Rm0} + \bar{\beta}_R(\mathbf{x}_k)w_{Rm1}x_{1k} + \dots + \bar{\beta}_R(\mathbf{x}_k)w_{RmN}x_{Nk} \end{aligned} \quad (17)$$

For all training patterns, we have K equations of Eq.(17). Thus, Eq.(16) can be expressed:

$$E_m = \| \mathbf{A} \mathbf{W}_m - \mathbf{Y}_m \| \quad (18)$$

Where, \mathbf{W}_m , \mathbf{Y}_m , and \mathbf{A} are matrices of $((N+1)*R) \times 1$, $K \times 1$, and $K \times ((N+1)*R)$ respectively.

$$\mathbf{W}_m = [w_{1m0}, w_{1m1}, \dots, w_{1mN}, \dots, w_{Rm0}, w_{Rm1}, \dots, w_{RmN}]^T \quad (19)$$

$$\mathbf{Y}_m = [y_{d1m} \ y_{d2m} \ \dots \ y_{dKm}]^T \quad (20)$$

$$\mathbf{A} = \begin{bmatrix} \bar{\beta}_1(\mathbf{x}_k) & \bar{\beta}_1(\mathbf{x}_k)x_{11} & \dots & \bar{\beta}_1(\mathbf{x}_k)x_{N1} & \dots & \bar{\beta}_R(\mathbf{x}_k) & \bar{\beta}_R(\mathbf{x}_k)x_{11} & \dots & \bar{\beta}_R(\mathbf{x}_k)x_{N1} \\ \bar{\beta}_1(\mathbf{x}_k) & \bar{\beta}_1(\mathbf{x}_k)x_{12} & \dots & \bar{\beta}_1(\mathbf{x}_k)x_{N2} & \dots & \bar{\beta}_R(\mathbf{x}_k) & \bar{\beta}_R(\mathbf{x}_k)x_{12} & \dots & \bar{\beta}_R(\mathbf{x}_k)x_{N2} \\ \dots & \dots \\ \bar{\beta}_1(\mathbf{x}_k) & \bar{\beta}_1(\mathbf{x}_k)x_{1K} & \dots & \bar{\beta}_1(\mathbf{x}_k)x_{NK} & \dots & \bar{\beta}_R(\mathbf{x}_k) & \bar{\beta}_R(\mathbf{x}_k)x_{1K} & \dots & \bar{\beta}_R(\mathbf{x}_k)x_{NK} \end{bmatrix} \quad (21)$$

Next, we apply linear least-squares algorithm (Jang et al., 1997) for each output (m -th output) to tune the parameters w_{rml} .

$$\mathbf{W}_m = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{Y}_m \quad (22)$$

After the forward pass in the learning, error signals are propagated backward to update the premise parameters a_{rn} , b_{rn} and c_{rn} by gradient decent with the error function $E(\Theta)$ in Eq.(14). The learning rule is given by:

$$a_{rn}^{new} = a_{rn}^{old} - \eta \frac{\partial E(\Theta)}{\partial a_{rn}}; b_{rn}^{new} = b_{rn}^{old} - \eta \frac{\partial E(\Theta)}{\partial b_{rn}}; c_{rn}^{new} = c_{rn}^{old} - \eta \frac{\partial E(\Theta)}{\partial c_{rn}} \quad (23)$$

Where, η is the learning rate. The formulae used to update the parameters a_{rn} , b_{rn} and c_{rn} are given in the Appendix.

5. Simulation Results

In the first set of simulations, the proposed method is compared with Fuzzy C-Means (Hppner et al., 1999), K-Means algorithm (Dubes, 1993), Feedforward Backpropagation Network (Schalkoff & Robert, 1997; Russell et al., 2003) and Anfis methods (Jang, 1991; Jang, 1993; Russell et al., 1997). The performance of our classifier system is demonstrated for SAR Image and a natural image.

To test the effectiveness of our proposed method, in the next set of simulations, fuzzy system is used to detect the edges of the image when it is significantly degraded by high noise. The proposed system is compared with other edge-detection methods: Prewitt (Prewitt, 1970), Roberts (Roberts, 1965), LoG (Marr & Hildreth, 1980), Sobel (Sobel, 1970), and Canny (Canny, 1986).

5.1. SAR Image Classification

The JPL L-band polarimetric SAR image (size: 1024x900 pixels) of San Francisco Bay (Tzeng & Chen, 1998; Khan & Yang, 2005; Khan et al., 2007) as shown in Fig. 2(a) is used for this simulation. The goal is to train the fuzzy system to classify three different terrains in this image, namely *water*, *park* and *urban* areas.

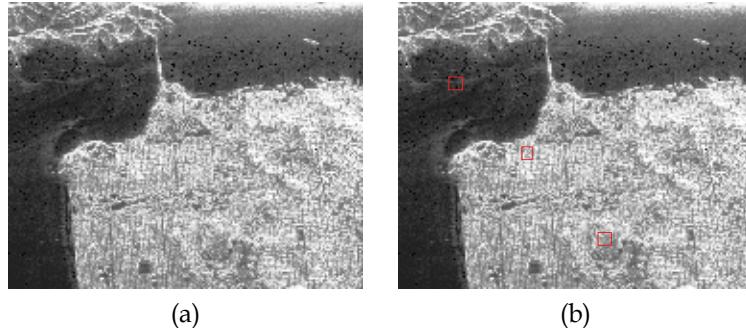


Fig. 2. SAR Image Classification, (a): original Image, (b): training data with 3 classes

The training patterns are shown enclosed in red boxes in Fig. 2(b). The proposed system was trained using these features to estimate the parameters. The algorithm was run with 100 training iterations.

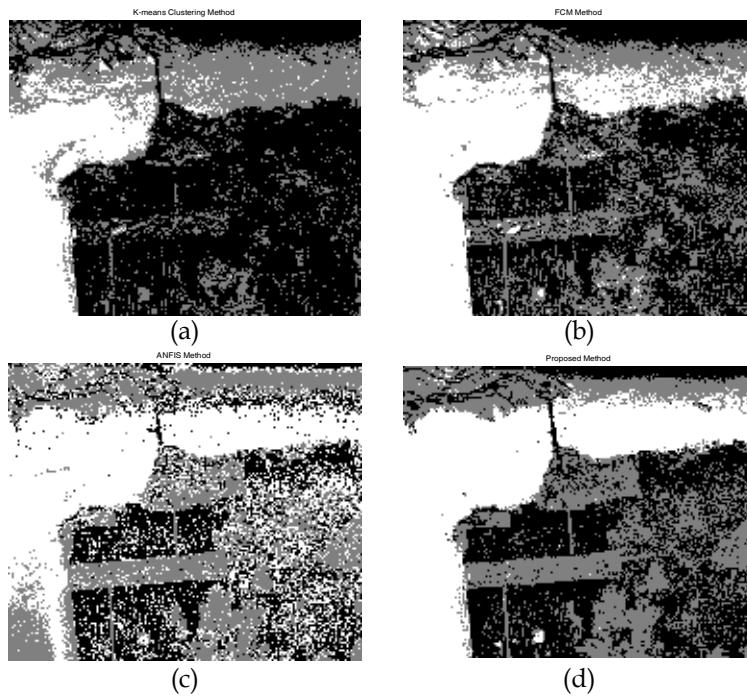


Fig. 3. SAR image classification results, (a): K-Means clustering method, (b): Fuzzy C-Means methods, (c): Anfis method, (d): proposed method.

In this example, the proposed system was used to indicate three distinct classes ($M=3$), with 3 inputs corresponding to 3 polarimetric channels: hh , vv , and vh (Tan et al., 2007), 4 rules

($R=4$). The desired outputs for *urban*, *park* and *water* classes were chosen to be [0 0 1], [0 1 0], and [1 0 0], respectively. After training with the patterns, the system was used to classify the whole image. Fig. 3(d) shows the classification results of the proposed method. A comparison of the proposed classifier with the K-Means classifier and Fuzzy C-Means classifier is shown in Fig. 3(a) and Fig. 3(b), respectively. These two methods were executed using MATLAB with the same 3 inputs (*hh*, *vv*, and *vh*), 3 outputs and default values for auxiliary parameters. As can be seen from Fig. 3, the classification accuracy of K-Means and Fuzzy C-Means methods was lower in water and park regions, as compared to the proposed method.

Fig. 3(c) shows the simulation result of Anfis. In this example, the same training areas in red boxes as shown in Fig. 2(b) were used to train the Anfis system. Anfis system with 3 inputs and 8 rules was run for 100 training iterations. The desired outputs for *urban*, *park* and *water* classes were chosen to be 1, 2, and 3, respectively. Compared with the Anfis method, clearly, our classifier accuracy is higher and the effect of noise on the performance of the detector is much less.

5.2. Natural Image Classification

In this experiment, the proposed system is compared to other classification algorithms by testing them on natural image taken from the Berkeley Dataset (Berkeley Dataset, 2001), as shown in Fig. 4.



Fig. 4. Natural Image Classification.

Fig. 5(a) shows the image corrupted by Gaussian noise (0 mean, 0.1 variance) that we want to segment into 3 classes (*snow*, *wolf*, and *tree*). This input image is scanned left-to-right by taking a square window of size 5x5 pixels around a centre pixel, which is then feed into the trained fuzzy system for classification into *snow*, *wolf* or *tree*.

To train our proposed system, the training patterns are generated as shown by red boxes in Fig. 5(a). For this experiment, we have chosen a fuzzy system with 25 inputs (corresponding to the 5x5 window), 8 rules ($R=8$) and 3 distinct classes ($M=3$) with the desired outputs for *snow*, *wolf* and *tree* classes as [0 0 1], [0 1 0], and [1 0 0], respectively. Fig. 5(b) shows the clustering results of Fuzzy C-Means classifier with 25 inputs, and 3 outputs.

The image shown in Fig. 5(c) is the result obtained using Feedforward Backpropagation networks. In this example, the networks is established with the structure of 25-8-8-8-3, five layer network with 3 hidden layers, 8 neurons in each hidden layer and 3 neurons in the output layer. We use *tansig* for hidden layers and *purelin* for the output layer. Both Fuzzy C-Means and Feedforward Backpropagation networks in this example were executed using

MATLAB with default values for auxiliary parameters. As can be seen, compared to other methods, the proposed system as shown in Fig. 5(d) could not only successfully segment the image when it is significantly degraded by high noise, but also reduces the effect of noise on the final segmented image.

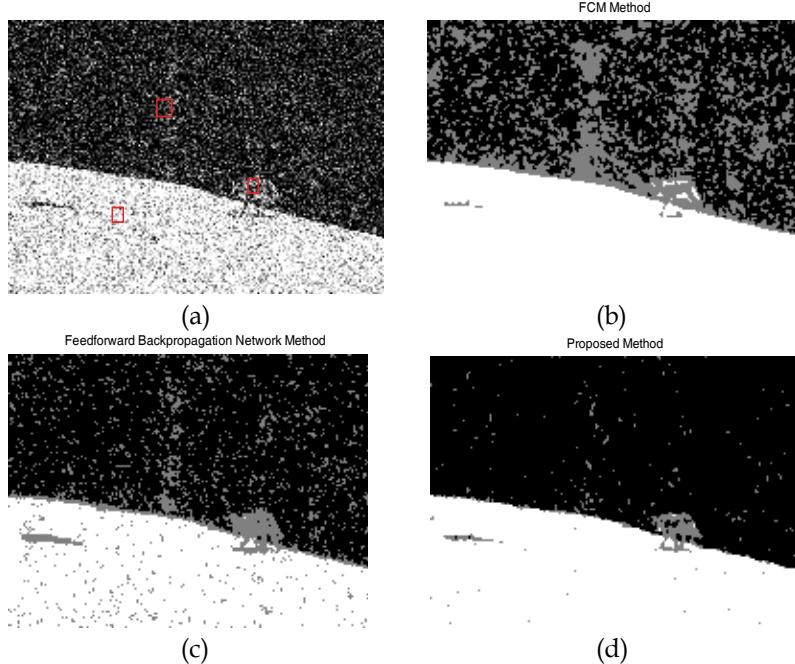


Fig. 5. Natural image classification results, (a): Noisy image, (b): Fuzzy C-Means methods, (c): Feedforward Backpropagation Network, (d): proposed method.

5.3. Edge Detection in Noisy Images

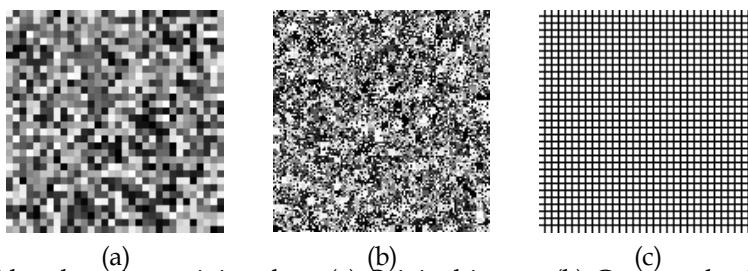


Fig. 6. Edge detection training data, (a) Original image, (b) Corrupted original image with 40% salt and pepper noise, (c) Target image.

In principle, edge detection is a two-class image classification problem where each pixel in the image is classified as either a part of the *background* or an *edge*. For this reason, a fuzzy system consisting of 2 output nodes corresponding to the 2 classes (*edge*, *background*) is chosen. In this experiment, a window of size 3x3 is scanned left-to-right across an image taken from the training set, and a determination is made as to whether the centre pixel

within the square neighbourhood window belongs to an *edge* (desired output classification [0,1]) or the *background* (desired output classification [1,0]). The the fuzzy model is structured with 9 inputs ($N=9$) corresponding to the 3×3 window, 16 rules ($R=16$), and 300 training epochs, to predict the binary decision class.

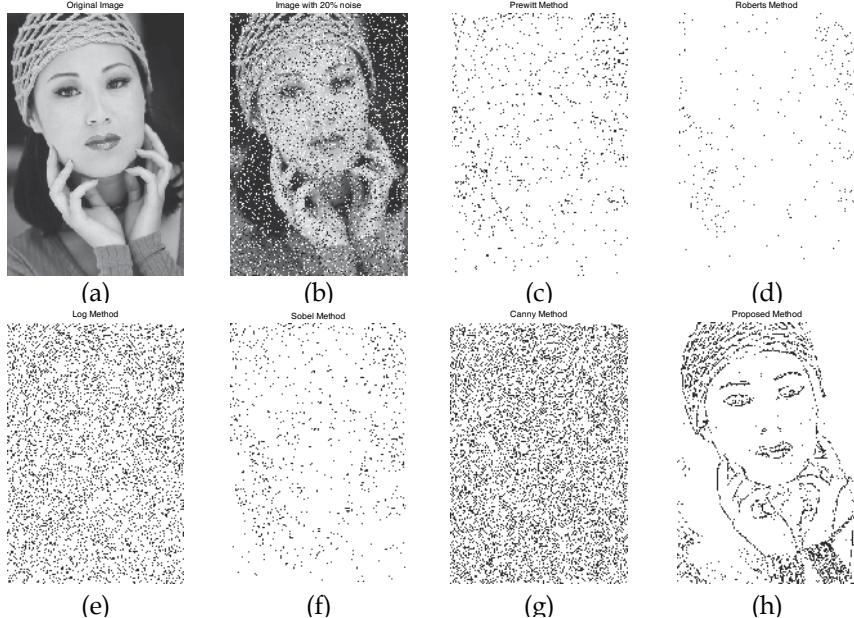


Fig. 7. The first natural image for checking (a) Original image, (b) Corrupted with 20% salt-and-pepper noise, (c) Prewitt method, (d) Roberts method, (e) LoG method, (f) Sobel method, (g) Canny method, (h) proposed method.

To train the proposed system, simple images (see Fig. 6) of size 128x128 pixels are utilized (Yksel, 2007). Fig. 6(a) shows the original image, where each square box of size 4×4 pixels has the same random luminance value. The input to the fuzzy system consists of the corrupted original image with 40% salt and pepper noise, as shown in Fig. 6(b). The target image shown in Fig. 6(c) is a black and white image, with black pixels indicating the locations of true edges in the input training image.

Once trained, the model is tested by applying it to a set of natural images taken from the Berkeley Dataset (Berkeley Dataset, 2001) as shown in Fig. 7(a). Images are corrupted with 20% of "salt" (with value 1) and "pepper" (with value 0) noise with equal probability, as shown in Fig. 7(b). The proposed detector is then compared to the existing methods - Prewitt, Roberts, LoG, Sobel and Canny detector. It is not an easy task to select good threshold values for these methods. In this case, all these methods are executed using MATLAB and with default values for auxiliary parameters. It can be easily seen that most of the edge structures of the noisy image cannot be detected by Prewitt in Fig. 7(c), Roberts in Fig. 7(d), LoG in Fig. 7(e), Sobel in Fig. 7(f) and Canny in Fig. 7(g). Besides, the effect of noise is still clearly visible as real edges are significantly distorted by the noise, and many noise pixels are incorrectly detected as edges. Comparing the results with these operators, the proposed method's classification accuracy as shown in Fig. 7(h) is quite high, the effect of

noise on the performance of the detector is much less, and the edges in the input images are successfully classified. These results indicate that the proposed system performs well when the even when image quality is significantly degraded by high noise.

Error! Reference source not found. shows the edge images which have been detected by our proposed system with different percentages of salt and pepper noise as applied to various natural images. The proposed fuzzy model consists of 16 rules ($R=16$) and 250 training epochs. The 1-st, 2-nd and 4-th column show the original images, images corrupted by 10%, and 20% salt and pepper noise, respectively. The final edge images corresponding to these noisy images as detected by the proposed system have been shown in 3-rd and 5-th columns. It can be easily seen that the proposed fuzzy system is highly robust with respect to noise in the natural images.

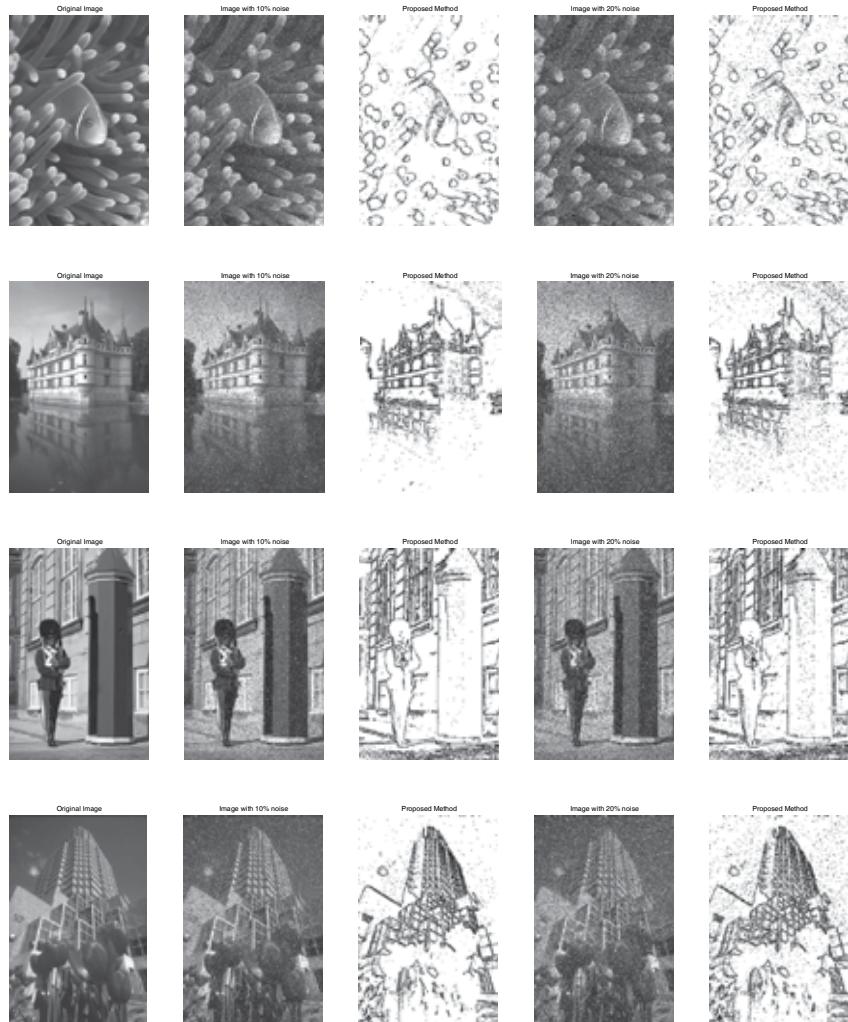


Fig. 8. The edge images which have been detected by proposed system with difference salt and pepper noise of difference natural images.

6. Conclusions

In this chapter, we have introduced a fuzzy rule-based system that combines both positive and negative association rules in its structure. A major advantage of this system is that each rule can represent more than one class. Through experimental tests and comparisons with existing algorithms on a number of natural images, it is found that the proposed system is a powerful tool for image classification.

7. Acknowledgement

This research has been supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

8. References

- Abe, S. & Thawonmas, R. (1997). A fuzzy classifier with ellipsoidal regions, *IEEE Transactions on Fuzzy Systems* 5 (3), page(s): 358-368.
- Berkeley Dataset, (2001): <http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench>.
- Branson, J.S. & Lilly, J.H. (1999). Incorporation of negative rules into fuzzy inference systems, *Decision and Control, Proceedings of the 38th IEEE Conference on*, Volume: 5, page(s): 5283-5288.
- Branson, J.S. & Lilly, J.H. (2001). Incorporation, characterization, and conversion of negative rules into fuzzy inference systems, *IEEE Trans. Fuzzy Syst.*, vol. 9, page(s): 253-268.
- Brin, S.; Motwani, R. & Silverstein, C. (1997). Beyond market basket: Generalizing association rules to correlations, *Proc. of SIGMOD*, page(s): 265-276.
- Canny, J. (1986). A Computational Approach to Edge Detection, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, page(s): 679-698.
- Dubes, R.C. (1993). Cluster analysis and related issues, *Handbook of Pattern Recognition and Computer Vision*, World Scientific Publishing Co., Inc., River Edge, NJ, page: 332.
- Gonzalez, A. & Perez, R. (1998). Completeness and consistency conditions for learning fuzzy rules, *Fuzzy Sets and Systems* 96, page(s): 37-51.
- Han, J. (2006). Learning Fuzzy Association Rules and Associative Classification Rules, *Fuzzy Systems, IEEE International Conference on*, page(s): 1454-1459.
- Hppner, F.; Klawonn, F.; Kruse, R & Runkler, T. (1999). *Fuzzy Cluster Analysis*, Wiley.
- Ishibuchi, H. & Nakashima, T. (2001). Effect of rule weights in fuzzy rule-based classification systems, *IEEE Trans. on Fuzzy Systems*, vol. 9, no. 4, page(s): 506-515.
- Ishibuchi, H. & Yamamoto, T. (2005). Rule weight specification in fuzzy rule-based classification systems, *IEEE Trans. on Fuzzy Systems*, vol. 13, no. 4, page(s): 428-435.
- Ishibuchi, H.; Nozaki, K. & Tanaka, H. (1992). Distributed representation of fuzzy rules and its application to pattern classification, *Fuzzy Sets and Systems* 52, page(s): 21-32.
- Jang, J.R.; Sun, C. & Mizutani, E. (1997). *Neuro-Fuzzy and Soft Computing*, Prentice-Hall, Englewood Cliffs, NJ, page(s): 113-115.
- Jang, J.S.R. (1991). Fuzzy Modeling Using Generalized Neural Networks and Kalman Filter Algorithms, *Proc. of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, page(s): 762-767.

- Jang, J.S.R. (1993). ANFIS: Adaptive-Network-based Fuzzy Inference Systems, *IEEE Trans. on Systems, Man and Cybernetics* 23 (3), page(s): 665–685.
- Khan, K. & Yang J. (2005). Novel features for polarimetric SAR image classification by neural network, *International Conference on Neural Networks and Brain*, page(s): 165–170.
- Khan, K.U.; Yang J. & Zhang W. (2007). Unsupervised Classification of Polarimetric SAR Images by EM Algorithm, *IEICE Transactions 90-B(12)*, page(s): 3632-3642.
- Lee, C.H. & Lin, Y.C. (2004). Hybrid learning algorithm for fuzzy neuro systems, Fuzzy Systems Proceedings, *IEEE International Conference on*, Volume 2, 25-29 July 2004, page(s): 691-696.
- Lilly, J.H. (2007). Evolution of a negative-rule fuzzy obstacle avoidance controller for an autonomous vehicle, *Fuzzy Systems, IEEE Transactions on*, 15(4), page(s): 718–728.
- Mamdani, E.H. & Assilian S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller, *Int. J. Man-Mach. Stud.*, vol. 7, page(s): 1-13.
- Mandai, D.P.; Murthy, C.A. & Pal S.K. (1992). Formulation of a multivalued recognition system, *IEEE Transactions on Systems, Man, and Cybernetics*, 22 (4), page(s): 607-620.
- Marr, D. & Hildreth E. (1980). Theory of edge detection, *Proc. of Royal Society London*, page(s): 187-217.
- Nayak, P.C.; Sudheer, K.P.; Ragan, D.M. & Ramasastri, K.S. (2004). A neuro fuzzy computing technique for modeling hydrological time series, *Journal of Hydrology* 29, vol 291, Issues 1-2, page(s): 52-66.
- Pal, S. & Mandai, D.P. (1992). Linguistic recognition system based on approximate reasoning, *Information Sciences* 61, page(s): 135-161.
- Prewitt, L.G. (1970). *Object Enhancements and Extraction in Picture Processing and Psychopictorics*, Academic Press, New York, NY, page(s): 75-149.
- Purwar, S.; Kar I.N. & Jha A.N. (2005). Adaptive control of robot manipulators using fuzzy logic systems under actuator constraints, *Fuzzy Sets and Systems* 152, page(s): 651–664.
- Roberts, L.G. (1965). *Machine Perception of Three Dimensional Solids*, in *Optical and Electrooptical Information Processing*, MIT Press, Cambridge, MA, page(s): 159-197.
- Russell; Stuart; Norvig & Peter (2003). *Artificial Intelligence: A Modern Approach*, 2nd Edition, Prentice Hall.
- Savasere, A.; Omiecinski, E. & Navathe, S. (1998). Mining for strong negative associations in a large database of customer transactions, *Proc. of ICDE*, page(s): 494–502.
- Schalkoff & Robert, J. (1997). *Artificial Neural Networks*, International Editions. McGraw-Hill.
- Sobel, I. E. (1970). *Camera Models and Machine Perception*, Ph.D. Thesis, Electrical Engineering Department, Stanford University, Stanford, CA.
- Takagi T. & Sugeno M. (1985). Fuzzy identification of systems and its application to modeling and control, *IEEE Trans. Syst., Man, Cybren.*, vol. 15, page(s): 116–132.
- Tan, C.P.; Lim, K.S. & Ewe, H.T. (2007). Image Processing in Polarimetric SAR Images Using a Hybrid Entropy Decomposition and Maximum Likelihood (EDML), *5th International Symposium on*, Sept. 2007, page(s): 418 – 422.
- Teng, W.; Hsieh, M. & Chen, M. (2002). On the mining of substitution rules for statistically dependent items, *Proc. of ICDM*, page(s): 442–449.

- Thanh, M.N. & Jonathan, W.Q.M. (2008). A Combination of Positive and Negative Fuzzy Rules for Image Classification Problem, *Proceedings of the 2008 Seventh International Conference on Machine Learning and Applications*, page(s): 741-746.
- Tzeng, Y.C. & Chen, K.S. (1998). A fuzzy neural network to SAR image classification, *IEEE Trans. Geosci. Remote. Sensing*, vol. 36, page(s): 301-307.
- Wang, J.S. & George Lee, C.S. (2002). Self-adaptive neuro-fuzzy inference systems for classification applications, *IEEE Trans. Fuzzy Syst*, vol. 10, page(s): 790-802.
- Wang, J.S.; Lee, C.S.G. & Juang C.H. (1999). Structure and Learning in Self-Adaptive Neural Fuzzy Inference Systems, *Proc. of the Eighth Intl Fuzzy Syst. Association World Conf.*, Taipei, Taiwan, page(s): 975- 980.
- Wu, X.; Zhang, C. & Zhang, S. (2002). Mining both positive and negative association rules, *Proc. of ICML*, page(s): 658-665.
- Yksel, M. E. (2007). Edge detection in noisy images by neuro-fuzzy processing, *International Journal of Electronics and Communications*, vol 61, Issue 2, page(s): 82-89.
- Zadeh, L.A. (1973). Outline of a new approach to the analysis of complex systems and decision processes, *IEEE Trans. Syst. Man. Cybern.*, vol. SMC-3, no. 1, page(s): 28-44.

APPENDIX:

We apply the gradient descent technique to modify the parameters a_{rn} , b_{rn} , and c_{rn} . Parameter update formula for k -th data set of a_{rn} is represented in Eq.(24). Similarly, the update rule of c_{rn} is derived in Eq.(25). The update rule of b_{rn} is derived in Eq.(26)

$$\frac{\partial E}{\partial a_{rn}} = \frac{\partial E(\Theta)}{\partial y_{km}} \frac{\partial y_{km}}{\partial \bar{\beta}_r(\mathbf{x}_k)} \frac{\partial \bar{\beta}_r(\mathbf{x}_k)}{\partial \beta_r(\mathbf{x}_k)} \frac{\partial \beta_r(\mathbf{x}_k)}{\partial A_{rn}(\mathbf{x}_k)} \frac{\partial A_{rn}(\mathbf{x}_k)}{\partial a_{rn}} \quad (24)$$

$$\frac{\partial E(\Theta)}{\partial c_{rn}} = \frac{\partial E(\Theta)}{\partial y_{km}} \frac{\partial y_{km}}{\partial \bar{\beta}_r(\mathbf{x}_k)} \frac{\partial \bar{\beta}_r(\mathbf{x}_k)}{\partial \beta_r(\mathbf{x}_k)} \frac{\partial \beta_r(\mathbf{x}_k)}{\partial A_{rn}(\mathbf{x}_k)} \frac{\partial A_{rn}(\mathbf{x}_k)}{\partial c_{rn}} \quad (25)$$

$$\frac{\partial E(\Theta)}{\partial b_{rn}} = \frac{\partial E(\Theta)}{\partial y_{km}} \frac{\partial y_{km}}{\partial \bar{\beta}_r(\mathbf{x}_k)} \frac{\partial \bar{\beta}_r(\mathbf{x}_k)}{\partial \beta_r(\mathbf{x}_k)} \frac{\partial \beta_r(\mathbf{x}_k)}{\partial A_{rn}(\mathbf{x}_k)} \frac{\partial A_{rn}(\mathbf{x}_k)}{\partial b_{rn}} \quad (26)$$

Moreover, the partial derivatives in Eq.(24) to Eq.(26) are as follows:

$$\frac{\partial E(\Theta)}{\partial y_{km}} = \frac{1}{KM} \sum_{k=1}^K \sum_{m=1}^M (y_{km} - y_{dkm}) \quad (27)$$

$$\frac{\partial y_{km}}{\partial a_{rn}} = W_{rm} \quad (28)$$

$$\frac{\partial \bar{\beta}_r(\mathbf{x}_k)}{\partial a_{rn}} = \frac{\bar{\beta}_r(\mathbf{x}_k)}{\beta_r(\mathbf{x}_k)} [1 - \bar{\beta}_r(\mathbf{x}_k)] \quad (29)$$

$$\frac{\partial \beta_r(\mathbf{x}_k)}{\partial a_{rn}} = \frac{\beta_r(\mathbf{x}_k)}{1 + \left(\frac{x_{nk} - c_{rn}}{a_{rn}} \right)^{2b_{rn}}} \quad (30)$$

$$\frac{\partial A_{rn}}{\partial a_{rn}} = \frac{2b_{rn}}{a_{rn}} \frac{\left(\frac{x_{nk} - c_{rn}}{a_{rn}} \right)^{2b_{rn}}}{\left(1 + \left(\frac{x_{nk} - c_{rn}}{a_{rn}} \right)^{2b_{rn}} \right)^2} \quad (31)$$

$$\frac{\partial A_{rn}}{\partial c_{rn}} = \frac{2b_{rn}}{x_{nk} - c_{rn}} \frac{\left(\frac{x_{nk} - c_{rn}}{a_{rn}} \right)^{2b_{rn}}}{\left(1 + \left(\frac{x_{nk} - c_{rn}}{a_{rn}} \right)^{2b_{rn}} \right)^2} \quad (32)$$

$$\frac{\partial A_{rn}}{\partial b_{rn}} = \frac{\left(\frac{x_{nk} - c_{rn}}{a_{rn}} \right)^{2b_{rn}}}{\left(1 + \left(\frac{x_{nk} - c_{rn}}{a_{rn}} \right)^{2b_{rn}} \right)^2} \left(-\log \left(\frac{(x_{nk} - c_{rn})^{2b_{rn}}}{(a_{rn})^{2b_{rn}}} \right) \right) \quad (33)$$

Automatic Construction of Knowledge-Based System using Knowware System

Sio-Long Lo and Liya Ding

Macau University of Science and Technology

Macau SAR

China

1. Introduction

Knowledge-based system (KBS) is a problem solving approach that makes use of human knowledge in possible ways. Usually, the knowledge used in KBS may be obtained directly from domain expert or through some kind of machine learning based on available data. The quality of knowledge used has an important impact on the performance of KBS. The success of development and application of an intelligent system requires the availability of two groups of people: AI experts who hold the techniques and tools for problem solving, and domain experts who know well the problem to be solved and hold domain knowledge leading to a necessity of the development of intelligent system. However, in reality, it is often a challenge to get the both groups working together to derive the inherent synergies.

Knowware System (KWS) is a framework proposed as development tool for design and development of KBS. KWS offers classes of knowledge-based processing unit to support developer in modelling their KBS, and generates the target KBS based on the definition from developer. A typical KBS generated by KWS is a hybrid intelligent system that contains a knowledge hierarchy and an inference engine. The knowledge hierarchy consisting of multiple components forms a static inference structure in KBS while the inference engine controls the dynamic inference flow through managing execution of components.

The inference in a hybrid KBS constructed by KWS is a truth value flow inference, with knowledge-based processing handled locally in each individual components and a truth value flow throughout the entire KBS. As a uniformed format, interval-valued confidence defined as fuzzy number has been proposed to represent the imprecision and uncertainty during inference. The KWS inference engine realizes control of inference through three aspects: the management of protocol between components, the control of execution order of components, and the confidence transfer.

2. Knowware System

The Knowware System (KWS) has been proposed for the development of knowledge-based systems. It can accept from user knowledge sources represented in varied formats and select appropriate intelligent techniques to construct desired knowledge-based processing units of

hybrid KBS, therefore allow the KBS developer more easily and conveniently model and develop a customized intelligent system.

2.1. Hierarchical Modeling of KBS

In a typical application, the mapping relation between inputs and output of the problem may be complex, and description of such a mapping relation using a global knowledge can be difficult and incapacity. A possible strategy is to divide the complex mapping relation to multiple units, with each of the units described by a corresponding local knowledge base, and the type of knowledge and the inference mechanism in each of the units varied upon the specific problem solving and the availability of knowledge. Following this spirit, hierarchical problem representation represents a domain problem with a hierarchy and uses multiple AI techniques for problem solving.

2.2. Construction of KBS using KWS

The hierarchical representation for KBS was introduced by (L. Ding and H.C. Lui, 1999; L. Ding, 2007a). The key idea of hierarchical representation for KBS is hybrid KBS, which consists of multiple sub-KBS constructed in a hierarchical structure (Figure 1). The KWS not only allows developers to easily design their system, but also realizes an automatic construction of the target KBS based on the developers' design.

As a typical development process, KWS receives the description of KBS from developer and then automatically constructs the target KBS. Therefore a Knowledge Description Language (KDL for short) is essential, which will be introduced in Section 2.2.3. Developers can use the KDL text to describe their system, and the text is a kind of input to KWS with a KBS constructed by KWS as the corresponding output. The KBS constructed by KWS is a stand-alone application, so the end-user can use the KBS easily without the care about the details of implementation.

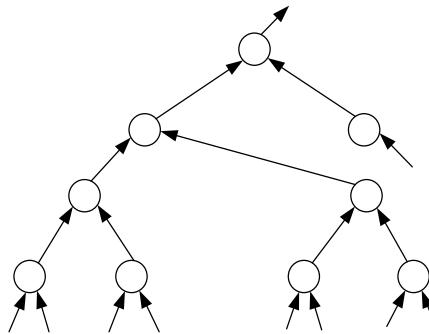


Fig. 1. Structure of KBS constructed by KWS

2.2.1. Sub-Systems of KWS for KBS Construction

There are three subsystems of KWS supporting the automatic construction of customized KBS.

Intelligent Editor – It provides a friendly GUI for the developer to design a KBS. Developers use the graphic description to describe their KBS. Editor also does error checking for the

process of developing KBS. Once design is confirmed, Editor will construct the internal inference structure of target KBS based on the graphic description, and generate the corresponding KDL text.

KDL Processor – It receives the KDL description of a target KBS, and compiles it to a corresponding knowledge hierarchy as the internal inference structure, using suitable intelligent components stored in the warehouse with possible customization. The KDL text can be either from the interactive editor or user's input. In the latter case, it also checks the syntax of KDL text inputted.

Installer – It saves the internal knowledge hierarchy in a suitable data format when a user's definition of target KBS is confirmed. At the last stage, it packs the knowledge hierarchy with the KWS inference engine as well as the installer itself to a stand-alone target application. The embedded installer will be responsible to reload the saved KBS upon user's calling of the application.

2.2.2. Work Flow of KWS

In order to develop a desired intelligent system, the developer can choose any of the knowware that fits into his/her need, via two possible ways. One is to define his/her target system in KDL text and then call the KDL processor for compilation to generate the internal inference structure. The other alternative is to use the intelligent editor to design the target system step-by-step and get the target knowledge hierarchy constructed after confirmation. In the latter case, the editor also generates a corresponding KDL text so the developer can make modification conveniently later on. For a KBS successfully constructed, the installer will save the internal inference structure to a suitable format and reconstruct it later upon request. Figure 2 shows the work flow of KWS.

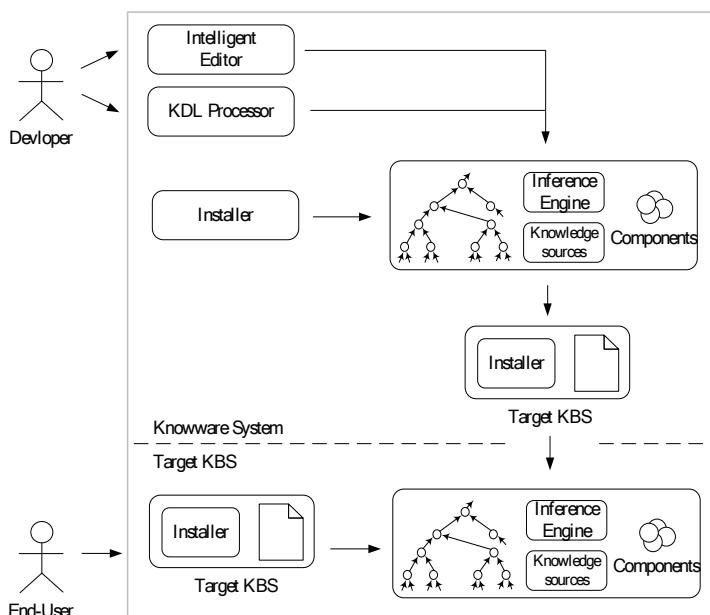


Fig. 2. Work flow of KWS

2.2.3. Knowledge Description Language

The Knowledge Description Language makes it possible for developers to describe their target KBS in a text format. The knowledge-based processing units offered by KWS will be used as building blocks to make up the KBS. The input/output of each intelligent component (IC) called field must be specified, this information indicates the linking between ICs and a pipeline for data connection with ICs. A KDL text consists of the following parts: 1) declaration of fields, each including name and data type; 2) declaration of intelligent components, each including name, component class and type, knowledge source, fields of input(s) and output linked with. The details of intelligent component and field will be presented in Sections 2.3. An example of KDL text is shown in Figure 3.

```

Support-Field-Name = ( Field1 ), Support-Field-Data { Char ( 2 ) }
Result-Field-Name = ( Field2 ), Result-Field-Data { Char ( 2 ) }
InCom-Name = ( Filter-01 ), InCom-Body {
    Filter Dictionary
    NoCondition
    Standard { Program = ( Standard Dictionary ),
        Knowledge-Source = ( Filter01_Knowledge ) }
    Input = ( Field = ( Field1 ) ), Output = ( Field = ( Field2 ) )
}

```

Fig. 3. An example of KDL

2.2.4. Generation of Target KBS

The last process of developing KBS using KWS is the generation of target KBS. The knowledge hierarchy will be recorded in a data-file. By packing the target KBS with hierarchy record, corresponding components, knowledge sources, inference engine, and installer, the KBS for the end-user is obtained as a stand-alone system. The task for packing and reloading of KBS is done by the Installer which also provides a GUI to the end-user based on the input/output of target KBS.

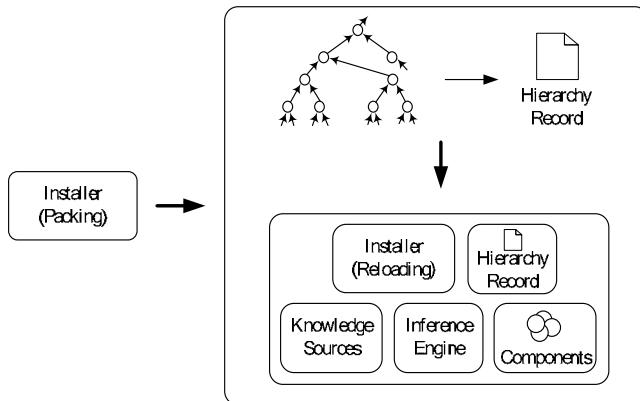


Fig. 4. Packing the KBS using Installer

Upon call to the target KBS received the installer will be started first to reload the KBS with all the necessary components, knowledge sources, and inference engine.

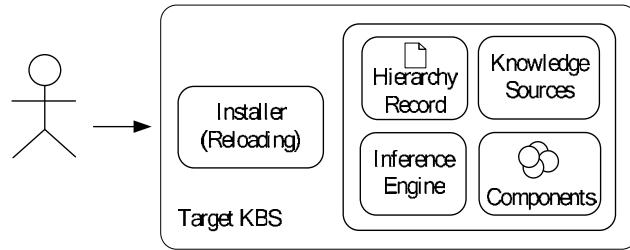


Fig. 5. Reloading the KBS using Installer

2.3. Components and Fields

The KWS warehouse stores pre-defined knowledge-based processing units that are the basic building blocks of KBS. Intelligent components are further classified by the nature of processing, in terms of the corresponding input and output. A KWS offers a set of k classes of intelligent components defined as

$$COM = \{com_1, \dots, com_k\},$$

and

$$com_i = <cl_i, t_i, s_i, c_i>,$$

where $i = 1, \dots, k$, $cl_i \in CL = \{cl_1, \dots, cl_k\}$, the set of class names of intelligent components; $t_i \in T_{cl_i}$, the *type* set under the class cl_i ; $s_i \in S_{cl_i}$, the *source and strategy* set under the class cl_i ; and $c_i \in C_{cl_i}$, the *connection* set under the class cl_i . At an *abstract* level, for any class cl defined, there is a mapping function f_{CL} :

$$f_{CL}: I_{CL} \xrightarrow{K} O_{CL}$$

where, I_{CL} is the input of the intelligent component of class cl , O_{CL} is the output, and K represents the corresponding knowledge-based processing. The features and properties of intelligent components under different classes are determined by their mapping function f_{CL} . It is an important feature of the KWS that an intelligent component under certain class always follows the same syntax for the interface with other intelligent components no matter which specific intelligent approach is adopted for the knowledge-based processing inside it. At the same time, intelligent components under the same class may behave differently when different approaches of knowledge-based processing are applied in problem solving. For instances, a decision-making may be done by applying traditional rule-based approach, or soft computing approaches, such as fuzzy logic inference, or neural networks; a knowledge discovery may be achieved by data mining applying different approaches; a prediction may be made by statistical methods or by using neural networks. When an intelligent component is defined as 'conditional component', it chooses suitable knowledge source to be applied among the alternatives provided according to run-time conditions detected. We have designed ten classes of intelligent components under two different categories: *processing components* and *learning components*, with each category including several classes based on the nature of function.

KWS also provides a possibility for the developers to include their own mathematical formulas or algorithms as user-defined procedures and make them intelligent components. Once such a procedure is defined, it becomes a special knowledge-based processing unit for possible use in other intelligent components in the same KBS under development.

Each of the input(s) and output of component is linked with a *Field*; fields are the basic data units indicated for input and output of processing intelligent components. They provide a pipeline for the data flow between components. An intelligent component can have multiple inputs, but only one output.

There are seven classes of processing component and three classes of learning component supported by KWS, as listed in Table 1.

Processing Component
<p><i>Filter</i> class applies its knowledge to check the input candidate list and filter out those "illegal" or "bad" members.</p> $f_{\text{Filtering}} : I_{\text{Filtering}} \xrightarrow{K} O_{\text{Filtering}}$ <ol style="list-style-type: none"> 1. Where $I_{\text{Filtering}}$ and $O_{\text{Filtering}}$ are the input date set and output data set respectively, and $I_{\text{Filtering}} \subseteq O_{\text{Filtering}}$; 2. The input and output share the same type of data structure; 3. The length of output should not be longer than that of input;
<p><i>Recognition</i> class applies its knowledge to "read out" the meaning of a single input pattern.</p> $f_{\text{Recognition}} : p \xrightarrow{K} L$ <ol style="list-style-type: none"> 1. Where p is a single pattern, and $L = \{l_1, \dots, l_k\}$ is a set of labels as possible recognition result, each of $l_i (1 \leq i \leq k)$ may be associated with a confidence value; 2. The input and output usually have different types of data structure; 3. The processing establishes one-to-one relation between an input pattern and an output label.
<p><i>Summarization</i> class contains the <i>Recognition</i> class as a special case, where the summary is a label or a highly summarized meaning.</p> $f_{\text{Summarization}} : p \xrightarrow{K} P$ <ol style="list-style-type: none"> 1. Where p is a single input pattern, $P = \{p'_1, \dots, p'_k\}$ is a set of patterns as possible summarization for the p, and each of $p'_i (1 \leq i \leq k)$ may be associated with a confidence value; 2. The input and output is equivalent or approximate in some degree, in terms of their meaning or explanation; 3. The degree or the level of abstraction of the output is determined by the knowledge applied and the inference mechanism adopted.
<p><i>Confirmation</i> checks the input, and gives "Yes/No" to each of the candidates.</p> $f_{\text{Confirmation}} : D_c \xrightarrow{K} YN$ <ol style="list-style-type: none"> 1. Where $D_c = \{d_1, \dots, d_k\}$ is a data set and $1 \leq k$, $YN = \{t_1, \dots, t_k\}$ is the corresponding truth list and $t_i (1 \leq i \leq k) \in \{\text{Yes}, \text{No}\}$; 2. It can be used as a conditional checker to support other intelligent components; 3. Fuzzy logic approaches may be introduced when a clear Yes/No cannot be simply decided.
<p><i>Judgement</i> is a more general class than <i>Confirmation</i> in the sense that the output can be defined as linguistic terms or values, such as high, expensive, going-up, or so.</p> $f_{\text{Judgement}} : D_{JP} \xrightarrow{K} JP$ <ol style="list-style-type: none"> 1. Where $D_{JP} = \{d_1, \dots, d_k\}$ is a data set and $1 \leq k$; $JP = \{\text{term}_1, \dots, \text{term}_k\}$ is a corresponding term list with possible confidence value associated, and $\text{term}_i (1 \leq i \leq k) \in LT$, the set of pre-defined linguistic terms; 2. Conceptually, it contains the <i>Confirmation</i> class as a special case where the judgement is simply represented as Yes/No; 3. Changing the LT may change the behaviour of intelligent component.
<p><i>Projection</i> projects an input data set with k features to an output data set with $j \leq k$ features.</p>

$f_{\text{Projection}} : D_k \xrightarrow{K} D_j$
<ol style="list-style-type: none"> 1. Where $D_k = \{<d^{(1)}_1, d^{(1)}_2, \dots, d^{(1)}_k>, \dots, <d^{(n)}_1, d^{(n)}_2, \dots, d^{(n)}_k>\}$ is an n-entry data set with k features, $D_j = \{<d^{(1)'_1}, d^{(1)'_2}, \dots, d^{(1)'_j}>, \dots, <d^{(n)'_1}, d^{(n)'_2}, \dots, d^{(n)'_j}>\}$ is an n-entry data set with j ($j \leq k$) features, and for any $1 \leq i \leq n$, the entry $<d^{(i)'_1}, d^{(i)'_2}, \dots, d^{(i)'_j}> \in D_j$ is an image of the entry $<d^{(i)}_1, d^{(i)}_2, \dots, d^{(i)}_k> \in D_k$ under the projection defined; 2. The process does not remove any data entry, but “remove” some of its features; 3. After projection, the data set will remain the entries but each of them appears in a space of probably lower dimension.
<i>Decision</i> checks the input as a situation and recommends a possible decision.
$f_{\text{Decision}} : s \xrightarrow{K} AD$
<ol style="list-style-type: none"> 1. Where s is a single situation, and $AD = \{ad_1, \dots, ad_k\}$ is a list of recommended action or decision with possible confidence value associated; 2. This class of intelligent components is usually used at a late or final stage of intelligent systems, but not at the beginning; 3. For a complicated problem, multiple techniques and approaches may be required to form the inference strategy used in the component.
Learning Component
<i>Discovery</i> not only makes use of knowledge but also produces knowledge. It has relevant domain data for input and gives output as the knowledge discovered.
$f_{\text{Discovery}} : D_D \xrightarrow{K} K_D$
<ol style="list-style-type: none"> 1. Where $D_D = \{d_1, \dots, d_k\}$ is a data set and $1 \leq k$; and K_D is a set of discovered knowledge of selected form, such as rules, relations, or other types; 2. Its output result can be applied as knowledge source to support other intelligent components; 3. It may use <i>Filtering</i> (a post-processing component's type) for its pre-processing or post-processing.
<i>Training</i> can train some rule on it based on the user input's data.
<i>Post-Processing</i> support <i>Learning Component</i> for post-processing.

Table 1. Intelligent component

2.4. KWS Inference Engine

The inference structure of a KBS constructed by KWS is represented as a knowledge hierarchy with multiple intelligent components connected. The task of construction can be done either by the intelligent editor or the KDL processor.

The knowledge hierarchy forms a static inference structure of the target KBS. A single intelligent component realizes the mapping from its input to its output with the support of its local knowledge base. The entire mapping of the KBS is achieved through the integration of intelligent components. There is no direct mapping relation from the input to the output of the KBS, but each intelligent component contributes to part of the mapping. Truth/confidence value is used to indicate uncertainty or imprecision that may occur in individual intelligent components, and also to connect the inference of individual components to the inference flow of the entire KBS.

One of the main challenges facing KWS for the construction of intelligent system is the complexity associated to inference mechanism having multi-level, and multi-modal knowledge integration. Each single intelligent component is actually a smaller KBS for a sub-problem of the target application, and its input and output can be directly linked to problem domain or the result from different stages of processing. How to assemble intelligent components to get a meaningful and unified data/information flow in the entire intelligent system constitutes a key task. Inference engine is necessary to control the execution which is realized through three aspects: 1) The management of protocol between

components; 2) The control of execution order of components; and 3) The confidence transfer.

3. Truth Value Flow Inference

The concept of truth value flow inference (TVFI) was first put forward by Wang et al (P.Z. Wang and H.M. Zhang, 1993). It offers a conceptual mechanism of fuzzy inference in a network structure (L. Ding et al, 1996) and finds rationality in connection to the description of fuzzy propositions. Figure 6 shows a conceptual illustration of implication $P \rightarrow Q$ with TVFI, where C_P and C_Q are the confidence of P and Q respectively; w is the weight of rule (L. Ding and Z. Shen, 1994) controlling the channel of transferring the truth of P to the truth of Q .

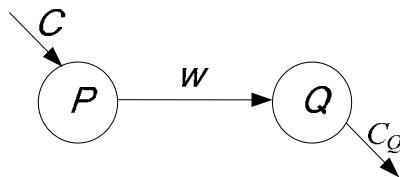


Fig. 6. Truth value flow inference

Based on the concepts of truth value flow inference and symbolical-numerical duality, we can construct a fuzzy inference with a static structure of nodes representing the relationship between propositions symbolically, and with a dynamic flow implementing the truth (or confidence) transfer among the individual nodes. This idea can be extended to KBS represented in network structure, with each intelligent component be treated as an extended node realizing a mapping relation between its input and output, and the entire KBS be an inference network (L. Ding and H.C. Lui, 1999).

3.1. Data Flow and Truth Value Flow in a Component

In order to have a unified inference flow in hybrid KBS, a possible solution is to separate the inference into a content level as well as a truth (confidence) level and handle them simultaneously. In this way, the content level of inference relies only on the knowledge sources stored "locally" in individual intelligent components whereas the truth (confidence) level of inference contributes to the flow of truth (confidence) throughout the entire system. So the inference in a hybrid KBS constructed by KWS is a truth value flow inference on the knowledge hierarchy.

In each intelligent component, two kinds of processing will be executed when receiving data in its inputs: content (or data) processing, and truth value (or confidence value) processing. These two types of processing are handled in intelligent component simultaneously to obtain the final result of the component that is the content of processing result associated with its corresponding confidence value. This concept can be shown as in Figure 7.

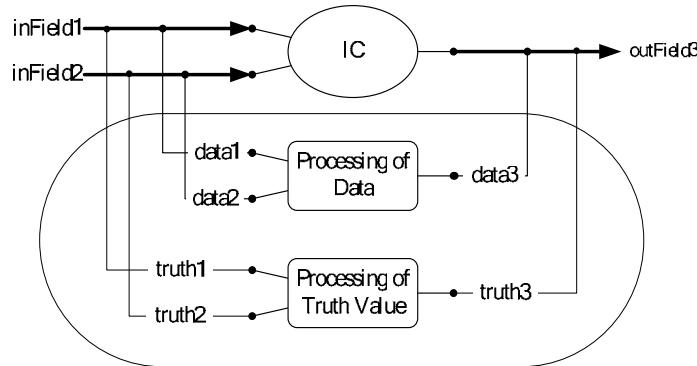


Fig. 7. An example of separated inputs in two levels

3.2. Interval-Valued Confidence

We adopted interval-valued confidence to represent the truth of fact and knowledge, and the confidence of inference. Here, the term “interval” is used in some different way from its usual meaning. Our motivation comes from several points.

- 1) A given truth value $t \in [0, 1]$ with some possible uncertainty or imprecision can always be understood as a fuzzy number defined on the closed interval $[0, 1]$. In an extreme case, t can be represented as a special fuzzy number with left, middle and right parameters at 0, t and 1 respectively, when its uncertainty reaches the maximum.
- 2) With more accurate information available, the range between the left and right parameters of such a fuzzy truth can be reduced. In another extreme case, we may have all the three parameters be t , if no uncertainty is considered, and it then comes back to usual case of single point of truth.
- 3) The acceptance of possible uncertainty associated with fuzzy truth is expected to allow more tolerance of imprecision in inference in terms of truth (confidence) calculation.

Three-parametric triangular truth value has some good features of easy representation and processing, intuitive interpretation consistent with common sense, convenient conversion from/to single-valued fuzzy truth, linguistic fuzzy truth, fuzzy numbers and fuzzy sets.

Definition 1 (general definition): A confidence value C of inference result is represented in the following general format:

$$C = (a, m, b), \quad (1)$$

where $0 \leq a \leq m \leq b \leq 1$, a is called the *left point*, b the *right point*, and m the *middle point*. It is a fuzzy subset defined on the universal set U which is the closed interval $[0, 1]$, i.e., $C \subseteq U = [0, 1]$ (Figure 8-a).

Definition 2 (conversion of single-valued truth): A single-valued truth $t \in [0, 1]$ of a fact inputted from user is represented as: $T = (a, m, b)$ with the left point $a = t$, the right point $b = t$, and the middle point $m = t$ (Figure 8-b). It looks like a fuzzy singleton, but should be understood as a special case (t, t, t) of Definition 1.

Definition 3 (conversion of interval-valued truth): An interval-valued truth $[a, b]$, $0 \leq a \leq b \leq 1$, of a fact inputted from user is represented as: $T = (a, m, b)$ with the middle point (Figure 8-c):

$$m = (a + b) / 2, \quad (2)$$

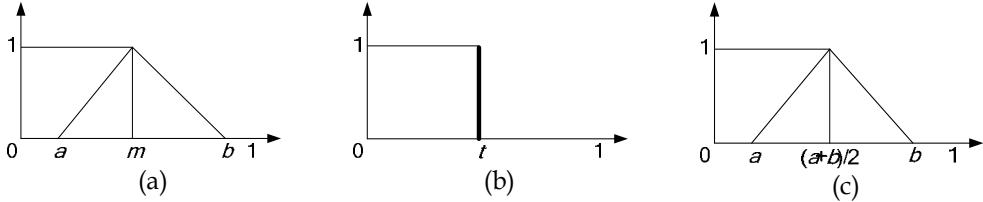


Fig. 8. Interval-Valued Confidence

Figure 9 shows the linguistic truth value true and false first given by L.A. Zadeh (L.A. Zadeh, 1975). Using the IVC defined in (1), we can represent them as: *true* = $(0, 1, 1)$, and *false* = $(0, 0, 1)$.

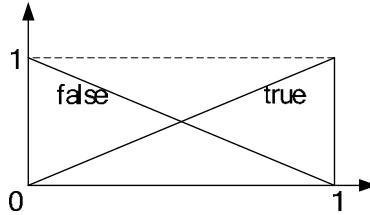


Fig. 9. Linguistic truth value *true* and *false*

Definition 4 (AND operation): The operation of AND on two interval-valued confidences $C_1 = (a_1, m_1, b_1)$ and $C_2 = (a_2, m_2, b_2)$, represented as in (1) is defined as

$$\begin{aligned} \text{AND}_{\text{IVC}}(C_1, C_2) &= (C_a, C_m, C_b) \\ &= [\min(a_1, a_2), \min(m_1, m_2), \max(m_1, m_2)]. \end{aligned} \quad (3)$$

Definition 5 (OR operation): The operation of OR on two interval-valued confidences $C_1 = (a_1, m_1, b_1)$ and $C_2 = (a_2, m_2, b_2)$, represented as in (1) is defined as

$$\begin{aligned} \text{OR}_{\text{IVC}}(C_1, C_2) &= (C_a, C_m, C_b) \\ &= [\min(m_1, m_2), \max(m_1, m_2), \max(b_1, b_2)]. \end{aligned} \quad (4)$$

Definition 6 (NOT operation): The operation of NOT on an interval-valued confidence $C = (a, m, b)$, represented as in (1) is defined as

$$\begin{aligned} \text{NOT}_{\text{IVC}}(C) &= (C_a, C_m, C_b) \\ &= (1 - b, 1 - m, 1 - a). \end{aligned} \quad (5)$$

Applying the operations defined in Definitions 4, 5, and 6 on linguistic truth values *true* and *false*, we have:

$$\mathbf{OR}_{\text{IVC}}(\text{true}, \text{false}) \quad (6)$$

$$= [\min(1, 0), \max(1, 0), \max(1, 1)] = \text{true},$$

$$\mathbf{AND}_{\text{IVC}}(\text{true}, \text{false}) \quad (7)$$

$$= [\min(0, 0), \min(1, 0), \max(1, 0)] = \text{false},$$

$$\mathbf{NOT}_{\text{IVC}}(\text{true}) \quad (8)$$

$$= (1 - 1, 1 - 1, 1 - 0) = (0, 0, 1) = \text{false},$$

$$\mathbf{NOT}_{\text{IVC}}(\text{false}) \quad (9)$$

$$= (1 - 1, 1 - 1, 1 - 0) = (0, 1, 1) = \text{true}.$$

The results of (6) ~ (9) are consistent with conventional definitions. However, we can also find that our operations provide interesting results when applying OR to both *true*, or AND to both *false*:

$$\mathbf{OR}_{\text{IVC}}(\text{true}, \text{true}) \quad (10)$$

$$= [\min(1, 1), \max(1, 1), \max(1, 1)] = (1, 1, 1),$$

$$\mathbf{AND}_{\text{IVC}}(\text{false}, \text{false}) \quad (11)$$

$$= [\min(0, 0), \min(0, 0), \max(0, 0)] = (0, 0, 0).$$

We shall call the result (1, 1, 1) of (10) “*strong-true*”, because it does not have any belief for “not-true”, and the result (0, 0, 0) of (11) “*strong-false*” as it does not provide any possible room for “not-false”. These two represent the extreme cases of IVC.

In order to have a further clear view of the properties of IVC with the corresponding operations, we list in Table 2 the typical results of $\mathbf{AND}_{\text{IVC}}$, \mathbf{OR}_{IVC} , and $\mathbf{NOT}_{\text{IVC}}$ on *true*, *false*, *strong-true* and *strong-false*, represented in IVC format. The highlighted parts show that the results well meet commonsense interpretation.

Definition 7 (generalized AND): The operation of AND on k ($k > 2$) interval-valued confidences $C_1 = (a_1, m_1, b_1), \dots, C_k = (a_k, m_k, b_k)$, represented as in (1) is defined as

$$\mathbf{AND}^{(g)}_{\text{IVC}}(C_1, \dots, C_k) = (C_a, C_m, C_b), \quad (12)$$

with C_a being the smallest value among the a_1, \dots, a_k , C_m the smallest value among the m_1, \dots, m_k , and C_b the second smallest value among the m_1, \dots, m_k .

$\mathbf{AND}_{\text{IVC}}$	F	T	s-F	s-T
F	F	F	s-F	F
T	F	T	F	T
s-F	s-F	F	s-F	F
s-T	F	T	F	s-T

(a) AND

\mathbf{OR}_{IVC}	F	T	s-F	s-T
F	F	T	F	T
T	T	T	T	s-T
s-F	F	T	s-F	T
s-T	T	s-T	T	s-T

(b) OR

$\mathbf{NOT}_{\text{IVC}}$	
F	T
T	F
s-F	s-T
s-T	s-F

(C) NOT

Table 2. Logical operations on *true*, *false*, *strong-true*, *strong-false*

Definition 8 (generalized OR): The operation of OR on k ($k > 2$) interval-valued confidences $C_1 = (a_1, m_1, b_1), \dots, C_k = (a_k, m_k, b_k)$, represented as in (1) is defined as

$$\mathbf{OR}^{(g)}_{\text{IVC}}(C_1, \dots, C_k) = (C_a, C_m, C_b), \quad (13)$$

with C_a being the second largest value among the m_1, \dots, m_k , C_m the largest value among the m_1, \dots, m_k , and C_b the largest value among the b_1, \dots, b_k .

It is necessary to notice that in general the $\text{OR}^{(g)}_{\text{IVC}}$ and $\text{AND}^{(g)}_{\text{IVC}}$ operations do not satisfy connective laws as usual logic OR and AND . This can be seen from the following examples.

Example 1: Generalized $\text{OR}^{(g)}_{\text{IVC}}$ and generalized $\text{AND}^{(g)}_{\text{IVC}}$ operations do not satisfy connective laws.

Suppose $C_1 = (0.7, 0.9, 0.9)$, $C_2 = (0.6, 0.6, 1)$, and $C_3 = (0.4, 0.4, 1)$, with Definition 7 we have:

$$\text{AND}^{(g)}_{\text{IVC}} (C_1, C_2, C_3) = (0.4, 0.4, 0.6). \quad (14)$$

However, we also have

$$\text{AND}_{\text{IVC}} [\text{AND}_{\text{IVC}} (C_1, C_2), C_3] \quad (15)$$

$$= \text{AND}_{\text{IVC}} [(0.6, 0.6, 0.9), (0.4, 0.4, 1)] = (0.4, 0.4, 0.6),$$

$$\text{AND}_{\text{IVC}} [C_1, \text{AND}_{\text{IVC}} (C_2, C_3)] \quad (16)$$

$$= \text{AND}_{\text{IVC}} [(0.7, 0.9, 0.9), (0.4, 0.4, 0.6)] = (0.4, 0.4, 0.9).$$

Similarly with Definition 8, we have

$$\text{OR}^{(g)}_{\text{IVC}} (C_1, C_2, C_3) = (0.6, 0.9, 1). \quad (17)$$

$$\text{OR}_{\text{IVC}} [\text{OR}_{\text{IVC}} (C_1, C_2), C_3] \quad (18)$$

$$= \text{OR}_{\text{IVC}} [(0.6, 0.9, 1), (0.4, 0.4, 1)] = (0.4, 0.9, 1),$$

$$\text{OR}_{\text{IVC}} [C_1, \text{OR}_{\text{IVC}} (C_2, C_3)] \quad (19)$$

$$= \text{OR}_{\text{IVC}} [(0.7, 0.9, 0.9), (0.4, 0.6, 1)] = (0.6, 0.9, 1).$$

The highlighted parts show the difference between (15) and (16), and between (18) and (19). This feature can be interpreted by the truth value flow inference adopted in KWS. It is understood that the corresponding structures of TVFI for (14), (15) and (16) are different (Figure 10), and the same applies to (17), (18) and (19). Figure 10 gives three structures of TVFI. The nodes of the structures are representing intelligent components for knowledge-based processing, and handling truth value flow from the input side to the output side. So the (a), (b), and (c) actually represent different *internal* inference flows, though they have the same input interface A_1, A_2 and A_3 , and output interface B for the entire structure.

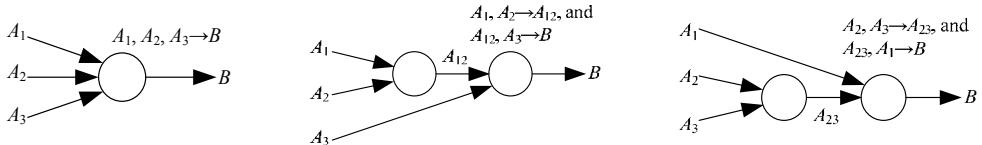


Fig. 10. TVFI structures

When a single-valued confidence of conclusion is desired, we need to consider defuzzification in the last stage of inference in hybrid KBS. As defuzzification is considered a matter of application-specific, we here propose two simplified calculations based on the

idea of conventional center of gravity approach (J.-S.R. Jang et al, 1997; R.R. Yager and D.P. Filev, 1994) with the reference to the left, middle and right points of IVC.

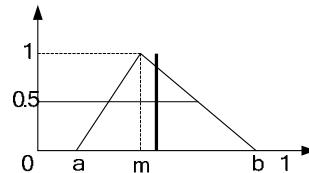
Definition 9 (compromised defuzzification): The compromised defuzzification $\text{Def}_{\text{com}}(C)$ of IVC $C = (a, m, b)$, $0 \leq a \leq m \leq b \leq 1$, is defined as the center of gravity of α -cut with $\alpha = 0.5$ (Figure 11-a).

Since the IVC is a fuzzy number defined as a piece-wise linear function with the corresponding left, middle, and right points, we have the following calculation:

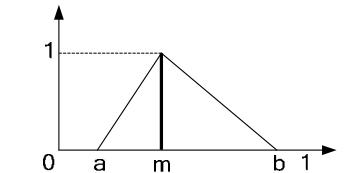
$$\text{Def}_{\text{com}}(C) = [(a + m)/2 + (m + b)/2] / 2. \quad (20)$$

Definition 10 (simple defuzzification): The simple defuzzification $\text{Def}_{\text{sim}}(C)$ of IVC $C = (a, m, b)$, $0 \leq a \leq m \leq b \leq 1$, is defined as the middle point m (Figure 11-b):

$$\text{Def}_{\text{sim}}(C) = m. \quad (21)$$



(a) Compromised defuzzification with IVC



(b) Simple defuzzification with IVC

Fig. 11. Defuzzification with IVC

Example 2: Single-valued confidence and interval-valued confidence with corresponding AND/OR operators.

A. Single-Valued Confidence using Min/Max for AND/OR operation

Consider the following rule and facts given:

- rule 1: if the topic is interesting,
and the weather is good,
then I will attend the seminar;
- fact 1: the topic is interesting;
- fact 2: the weather is good.

The most common way of handling *and* is to use *min* as *t*-norm to calculate the overall truth of the premise from the two subpremises, and when both facts are 0.5 true, for instance, we get $\text{min}(0.5, 0.5) = 0.5$. Now let's consider the fact about interesting topic:

- fact 1': the topic is interesting (0.9 true).

With *min*, we will still get the same truth 0.5 for the premise, i.e., the influence of interesting topic has been buried by the fact of weather as long as its truth is not lower than the other fact. However, we tend to agree that a more interesting topic (0.9) makes a person more willing to go to the seminar than a moderately interesting topic (0.5) given the same weather condition (0.5). The following example shows a similar problem with *or* operation.

- rule 2: if Mr. A and Mr. B are first cousin,
or second cousin,
then they have a close relationship;
fact 3: Mr. A and Mr. B are first cousin;
fact 4: Mr. A and Mr. B are second cousin.

With *max*, the most common way of calculating *t*-conorm, when either fact 3 or fact 4 is 0.5 true and the other is 0 true, we can obtain the overall truth of premise to be $\max(0.5, 0) = 0.5$. However, it is also possible that Mr. A and Mr. B have both first cousin and second cousin relationship when one's parents being first cousin. Assume both subpremises with 0.5 confidence, we will still have the same 0.5 for the truth of premise using max calculation. Conventionally, two persons that are in both first cousin and second cousin relation should more likely to have a close relationship than only being one kind of cousin having their double connections of relative. Obviously, *max* does not well reflect this situation.

From the above examples, we can see that a single-valued truth (confidence) does not provide sufficient room for the description of imprecise knowledge, especially in decision making applications, where subjective knowledge and experience play an important role and the truth of subjective knowledge is hardly to be measurable in absolute sense. It is also very often that in real applications, a single-valued truth (confidence) does not necessarily mean in the explicit way as it seems. For instance, when a user inputs 0.8 as the truth of good weather, it should not be simply treated as a precise value 0.8 but some thing *around* 0.8.

B. Interval-Valued Confidence using AND_{IVC}/OR_{IVC}

We apply the IVC and corresponding operations to the previous examples. For fact 1 and fact 2, we have

$$\begin{aligned} AND_{IVC} [(0.5, 0.5, 0.5), (0.5, 0.5, 0.5)] \\ = [\min(0.5, 0.5), \min(0.5, 0.5), \max(0.5, 0.5)] \\ = (0.5, 0.5, 0.5), \end{aligned}$$

and for fact 1' and fact 2, we have

$$\begin{aligned} AND_{IVC}[(0.9, 0.9, 0.9), (0.5, 0.5, 0.5)] (13) \\ = [\min(0.9, 0.5), \min(0.9, 0.5), \max(0.9, 0.5)] \\ = (0.5, 0.5, 0.9). \end{aligned}$$

It shows that fact 1' together with fact 2 gives more potential to have a truth higher than 0.5 (Figure 12).

For fact 3 (0.5 true) or fact 4 (0 true), we have

$$\begin{aligned} OR_{IVC}[(0.5, 0.5, 0.5), (0, 0, 0)] \\ = [\min(0.5, 0), \max(0.5, 0), \max(0.5, 0)] \\ = (0, 0.5, 0.5), \end{aligned}$$

and for fact 3 (0.5 true) or fact 4' (0.5 true), we have

$$\begin{aligned}
 \text{OR}_{\text{IVC}}[(0.5, 0.5, 0.5), (0.5, 0.5, 0.5)] \\
 &= [\min(0.5, 0.5), \max(0.5, 0.5), \max(0.5, 0.5)] \\
 &= (0.5, 0.5, 0.5).
 \end{aligned}$$

It shows that fact 3 together with fact 4' has a stronger belief for truth 0.5 (Figure 13).

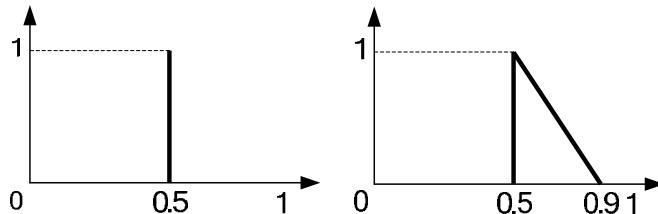


Fig. 12. An example of AND_{IVC}

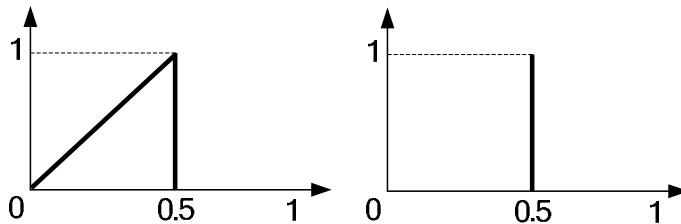


Fig. 13. An example of OR_{IVC}

From above discussion, we can see that using IVC and the corresponding operations defined for confidence calculation, partial conclusion with a relatively stronger confidence about true or false will not easily make the influence of other parts be totally ignored in inference.

3.3. Confidence Transfer and Interpretability

The processing in intelligent component can be further classified in several categories according to the description of mapping relation.

3.3.1. Component with Interpretable Mapping Relation

When mapping relation between input and output of intelligent components can be interpreted by a mathematic formula or an algorithm described by procedure. The mapping relation is considered interpretable, and precise in the sense that the processing does not affect uncertainty and imprecision. In this type of intelligent components, the confidence of output remains the same as that of input.

Example 3: A component performs some simple data processing, such as sorting. The output of component is the sorted result based on certain condition specified with knowledge source. In this case, the mapping relation between input and output can be determined by algorithm, and the corresponding output truth value remains the same as the input truth value.

If a component uses rule-based knowledge (fuzzy or precise) that can be approximately interpreted by **AND**, **OR**, and **NOT** relations, the mapping relation is also a kind of interpretable, but the truth of output may be affected by the knowledge-based processing. The corresponding truth value of output can be determined by the logic relations used in the rules.

3.3.2. Component with Less Interpretable Mapping Relation

When an IC uses less interpretable knowledge representation, e.g., neural networks, or case-based reasoning, the mapping relationship realized by the IC may not be interpreted in composition of logic operations and therefore the input confidence of the IC cannot be simply transferred to its output side to obtain the output confidence through its internal inference structure (L. Ding and S.L. Lo, 2008). A further extension of the framework of truth value flow inference using IVC (L. Ding, 2008) is needed to cope with this problem. It is achieved by two steps:

- 1) First carry out the internal inference of such an IC by assuming that the input is completely true (i.e. with full confidence);
- 2) Combine the input confidence with the result confidence as one unified output confidence at the output side of an IC after its processing.

We adopt the concept of truth base introduced with the exponential form of fuzzy logic (Z. Shen and L. Ding, 1994) for the interpretation of confidence transfer.

A. Truth base and confidence representation

The exponential form of fuzzy logic (EF) was proposed for confidence comparison and high order fuzziness simplification (Z. Shen and L. Ding, 1994). It provides a possible way for confidence transfer in intelligent components that use less interpretable representation of knowledge. An important concept introduced with EF is the truth base. For instance, saying “*P* is 0.8 true” may be understood in two ways: “*P* has *complete truth* (1) with 0.8 confidence”, or “*P* has 0.8 truth with *full confidence* (1)”. The difference is from the use of different truth bases: in the former, we put our truth base at 1, whereas in the latter, we put our truth base at 0.8. Obviously, it is reasonable to make these two ways of understanding be exchangeable from one to the other equivalently.

Usually by default we take *completely true* as the basis of our discussion about confidence, e.g.: 1 in fuzzy valued logic, or true in fuzzy linguistic valued logic, but it is also useful to have a different truth base for the convenience of discussion and have confidences of different truth bases be convertible from one to other.

The EF is originally defined with both *truth-I* and *truth-II* of fuzzy valued logic and fuzzy linguistic valued logic (Z. Shen and L. Ding, 1994). In KWS *truth-I* of fuzzy valued logic is adopted.

Definition 11 (EF on fuzzy valued logic): Let $t \in [0, 1]$ be a truth value in fuzzy valued logic, then t can be represented in its exponential form B^c when

$$t = (B - U) \times c + U, \quad (22)$$

where $B \in [0, 1]$ is called the *fuzzy truth base*, $c \in (-\infty, \infty)$ is called *confidence exponent*, \mathbf{U} is the unknown point for inference. In *truth-I*, we further specify $\mathbf{U} = 0$, and $B \in (0, 1]$.

It is important to be aware of that a super confidence $c > 1$ may cause a loss of information in inference (Z. Shen and L. Ding, 1994), so a truth base $B \geq t$ is usually recommended.

When applying the EF originally defined with single truth value to IVC, we have the *IVC format of unknown* $\mathbf{U}_{\text{IVC}} = (0, 0, 0)$, the *IVC format of truth base* $\mathbf{B}_{\text{IVC}} = (B, B, B)$ with $B \in (0, 1]$, and the IVC format of confidence exponent $C = (a_c, m_c, b_c)$. So the above (22) can be rewritten as:

$$t_{\text{IVC}} = (a_t, m_t, b_t) = (B \times a_c, B \times m_c, B \times b_c). \quad (23)$$

Definition 12 (Base changing in EF): The exponential form of a fuzzy truth t on truth base B_1 can be converted to that on truth base B_2 by

$$t = B_1^{C_1} = B_2^{C_2}. \quad (24)$$

where $B_1, B_2 \neq 0$, \mathbf{U} is the unknown point of inference, $B_1, B_2 \neq \mathbf{U}$, and c_1, c_2, B_1 and B_2 satisfy the following relation:

$$c_2 = c_1 \times (B_1 - \mathbf{U}) \div (B_2 - \mathbf{U}). \quad (25)$$

Using the IVC format of truth base and unknown point, given two confidences $C_1 = (a_1, m_1, b_1)$ under truth base $B_1 = (B_1, B_1, B_1)$ and $C_2 = (a_2, m_2, b_2)$ under $B_2 = (B_2, B_2, B_2)$, the above (23) can be rewritten as:

$$C_2 = (a_2, m_2, b_2) = (a_1 \times B_1 \div B_2, m_1 \times B_1 \div B_2, b_1 \times B_1 \div B_2). \quad (26)$$

Definition 13 (Logical operations on EF): The AND, OR and NOT operations on EF are defined as:

$$\text{AND}(B^{C_1}, B^{C_2}, \dots, B^{C_n}) = B^{\text{AND}(c_1, c_2, \dots, c_n)} \quad (27)$$

$$\text{OR}(B^{C_1}, B^{C_2}, \dots, B^{C_n}) = B^{\text{OR}(c_1, c_2, \dots, c_n)} \quad (28)$$

$$\text{NOT}(B^C) = B^{\text{NOT}(c)} \quad (29)$$

where B is a given *common truth base*, and EF values originally with different truth bases are converted to the selected *common truth base* before carrying out logical operations.

B. Confidence transfer with arbitrary intelligent component

Assume an arbitrary intelligent component A with $m \geq 1$ input variables $\text{in}_1, \text{in}_2, \dots, \text{in}_m$ (Figure 14) without loss of generality, where K represents a knowledge-based mapping realized in this component. The input in_k ($1 \leq k \leq m$) is denoted by $\langle d_k, B_k^{C_k} \rangle$, where the data d_k is from other intelligent component $IC-k$, and associated with IVC $C_k = (a_k, m_k, b_k)$ under a selected truth base B_k . When a common truth base B is selected for all the inputs $\text{in}_1, \text{in}_2, \dots,$

in_m , can be represented in its simplified form by $\langle d_k, C_k \rangle$ without confusion caused. It will be considered as a special case of having an empty IC when d_k is directly from problem domain. The inference output of A is obtained through the following algorithm.

In a hybrid KBS constructed by KWS, the *default common truth base* is set as $T = (1, 1, 1)$, the *strong true* in IVC format. This also applies to intelligent components with interpretable rule-based type of knowledge, and so the discussion of confidence transfer in (L. Ding, 2008) can be rebuilt using EF with the default common truth base.

Algorithm-1 (Confidence transfer of IC uses less interpretable knowledge representation):

- 1) The input $\langle d_k, B_k^{C_k} \rangle$ ($1 \leq k \leq m$) is first converted to $\langle d_k, T^{C_{in-k}} \rangle$, where T is the *strong true* (1, 1, 1) in IVC format, and $C_{in-k} = (a_{in-k}, m_{in-k}, b_{in-k})$ is the confidence exponent in IVC format, through base changing;
- 2) The *combined confidence of input* is then calculated by

$$C_{in} = (a_{in}, m_{in}, b_{in}) = [\min_k(a_{in-k}), \min_k(m_{in-k}), \min_k(b_{in-k})];$$
- 3) The data $d_1, \dots, d_k, \dots, d_m$ are then accepted as input values with perfect confidence for the inference in A;
- 4) Assume that a data r is obtained as the content of inference result of A with the result confidence B_r^{Cr} , where $C_r = (a_r, m_r, b_r)$. The T^{Cr*} is converted to B_r^{Cr} through base changing, then the output confidence of A is calculated by

$$C_{out} = \text{AND}_{IVC}(C_{in}, C_r^*)$$

based on the Definition 4, and $\langle r, T^{C_{out}} \rangle$ is the output of A

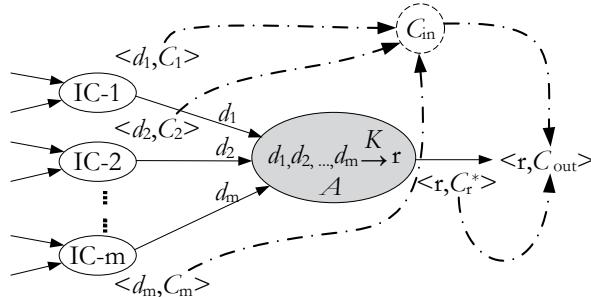


Fig. 14. Confidence transfer in an IC of hybrid KBS

3.3.3. Confidence Transfer in Hybrid KBS

Compared with a typical fuzzy inference system (J.-S.R. Jang et al, 1997; R.R. Yager, 1994), a hybrid KBS constructed by KWS usually does not have a universal knowledge base but multiple knowledge sources associated in individual intelligent components. In this sense, each knowledge source has only a local affection to the corresponding intelligent component realizing a mapping relation between its input and output. Given two arbitrary intelligent components A and B, having the output of A linked to the input of B means its content is passed on for further knowledge-based processing in B, and at the same time its confidence is integrated in the calculation of the confidence of output of B. Therefore, an IC is needed to distinguish the uncertainty associated with external input or introduced by its internal inference result. We represent the former as input confidence, and the latter as result

confidence. When all the intelligent components in a hybrid KBS use rule-based knowledge (fuzzy or precise) that can be approximately interpreted using AND, OR, and NOT relations, the inference of the KBS can be interpreted as a confidence flow on an extended logic-based network that embeds internal inference structure of individual IC into the knowledge hierarchy (L. Ding et al, 1996; L. Ding, 2008).

Example 4: With the rapid development of Internet technologies, people are receiving more and more e-mails for commercial promotion purpose and often need spend time and effort for filtering and cleaning. We consider a simple hierarchical KBS for the filtering function based on the title of e-mail. There are three major parts of title related to promotion: action (of promotion), e.g., "offer", "provide", or "sale"; benefit (to user), e.g., "saving", "earn", or "win"; currency (or percentage, number), e.g., "\$", "%", or "xx.xx".

The system consists of five intelligent components of type summarization, recognition, and decision (L. Ding, 2007b; L. Ding and S. Nadkarni, 2007) (Fig. 15).

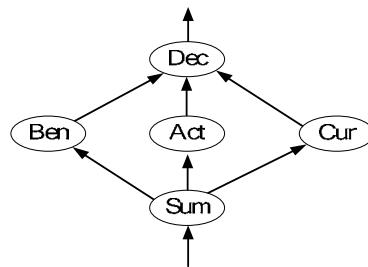


Fig. 15. An example of hierarchical inference with IVC flow

Sum (S) - a summarization component that performs pre-processing to eliminate less relevant words, such as "a", "and", and so on, indicated in a knowledge source of dictionary. The cleaned-up version of email title will be passed up to three recognition components for further processing.

Act (A) - a recognition component to recognize words that match the "action" category. The knowledge source defines the kind of words often used to positively describe promotion action, including the representative words and their major variants.

Ben (B) - a recognition component to recognize words that match the "benefit" category. The knowledge source defines the kind of words often used to highlight the potential benefit to attract people's attention, including the representative words as well as their major variants.

Cur (C) - a recognition component to recognize characters of currency, percentage, or numbers.

Dec (D) - a decision component to decide whether the text examined is suspicious for a promotion with the combined results from A, B, and C. The decision knowledge may be fuzzy association rules obtained through possible knowledge discovery, such as:

- rule-d1: If A **and** B, Then P (0.8)
- rule-d2: If B **or** C, Then P (0.4)
- rule-d3: If A **and** C, Then P (0.6)

When an e-mail is received with a title like:

"Special Offer for ... Saving up to 99% ...", the corresponding processing steps will be:

- (a) S filters out the less relevant words and obtained a cleaned-up version:
"Offer Saving 99%"
- (b) A recognizes "Offer" as an action word with a match in IVC $(0, 1, 1)$.
- (c) B recognizes "Saving" as a variant of benefit word "save" with IVC $(0, 0.8, 1)$.
- (d) C recognizes "%" as a percentage character with confidence $(0, 1, 1)$ as well as number "99" with confidence $(0, 1, 1)$. Assume the knowledge of C is defined as:

If \$or%-character or number then is-C.

So we have overall confidence for C is

$$\text{OR}_{\text{IVC}}[(0, 1, 1), (0, 1, 1)] = (1, 1, 1).$$

- (e) D combines the results from A , B , and C . Here, the confidence of conclusion is defined as **and** (confidence-of-premise, confidence-of-rule). We check each of the rules.

d1: confidence-of-premise

$$= \text{AND}_{\text{IVC}}[(0, 1, 1), (0, 0.8, 1)] = (0, 0.8, 1);$$

confidence-of-conclusion

$$= \text{AND}_{\text{IVC}}[(0, 0.8, 1), (0.8, 0.8, 0.8)] = (0, 0.8, 0.8).$$

d2: confidence-of-premise

$$= \text{OR}_{\text{IVC}}[(0, 0.8, 1), (1, 1, 1)] = (0.8, 1, 1);$$

confidence-of-conclusion

$$= \text{AND}_{\text{IVC}}[(0.8, 1, 1), (0.4, 0.4, 0.4)] = (0.4, 0.4, 1).$$

d3: confidence-of-premise

$$= \text{AND}_{\text{IVC}}[(0, 1, 1), (1, 1, 1)] = (0, 1, 1);$$

confidence-of-conclusion

$$= \text{AND}_{\text{IVC}}[(0, 1, 1), (0.6, 0.6, 0.6)] = (0, 0.6, 1).$$

- (f) Now we aggregate the results of d1~d3 from (e):

$$\text{OR}^{(g)}_{\text{IVC}}[(0, 0.8, 0.8), (0.4, 0.4, 1), (0, 0.6, 1)]$$

$$= (0.6, 0.8, 1).$$

Therefore, this e-mail is a commercial promotion with a high possibility 0.8 (if defuzzification is applied) or "very likely" as a linguistic interpretation.

Example 5: We replace the component Dec (D) in Example 4 with the below:

$Dec2$ ($D2$) - a decision component to decide whether the text under check is suspicious for a promotion with the combined results from A , B , and C . The decision knowledge used is case-based reasoning technique.

When an e-mail is received with a title like:

"Special Offer for ... Saving up to 99% ...", the corresponding processing steps will be:

- (a) ~ (d), the same as in Example 4;
- (e) $D2$ combines the results from A , B , and C . Using Algorithm-1 given in Section 4.3.2, we have:

Step-1: Result from A <"Offer", $(0, 1, 1)$ > converted to <"Offer", $(1, 1, 1)$ $(0, 1, 1)$ >;

Result from B <"Saving", $(0, 0.8, 1)$ > converted to <"Saving", $(1, 1, 1)$ $(0, 0.8, 1)$ >;

Result from C <"99%", $(0, 1, 1)$ > converted to <"99%", $(1, 1, 1)$ $(0, 1, 1)$ >.

Step-2: The combined confidence of input is then calculated by

$$\begin{aligned}
 C_{in} &= (a_{in}, m_{in}, b_{in}) \\
 &= (\min(0, 0, 0), \min(1, 0.8, 1), \min(1, 1, 1)) \\
 &= (0, 0.8, 1)
 \end{aligned}$$

Step-3: The data <"Offer", "Saving", "99%"> are then accepted as input values with perfect confidence for the inference in **D2**

Step-4: Assume that component **D2** through case-based reasoning technique to get the result <"Promotion", (0, 0.9, 1)>, we have:

$$C_r = (0, 0.9, 1) \text{ and } C_{in} = (0, 0.8, 1)$$

Then the output confidence of **D2** is calculated by

$$\begin{aligned}
 C_{out} &= \text{AND}_{IVC}(C_{in}, C_r^*) \\
 &= (0, 0.8, 0.9)
 \end{aligned}$$

- (f) Finally, we have the result <"Promotion", (1, 1, 1)(0, 0.8, 0.9)> and converted to <"Promotion", (0, 0.8, 0.9)>, it is the final result.

Therefore, this e-mail is a commercial promotion with a high possibility 0.8 (if defuzzification is applied).

4. KWS Inference Engine

As mentioned previously, KDL processor is to construct the knowledge hierarchy that forms the static inference structure of target KBS. The execution on such a static inference structure of KBS is carried out layer by layer in bottom-up manner. An inference engine is needed to control the execution of components in KBS by managing protocol between components, and sending necessary signals for the order of execution. A component in an inference structure constructed by KWS is a customized knowledge-based processing unit, and a field in the inference structure is a space that stores input data or intermediate result during inference. Fig. 16 gives an example of inference structure.

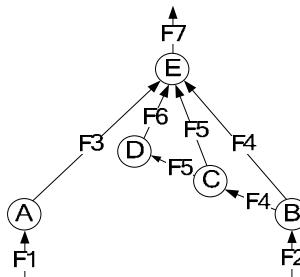


Fig. 16. An example of inference structure

4.1. Level of Component and Layer of Field

Based on the position of each component in inference hierarchy, a topological sorting determines the execution order with which a child component should always be executed before its parent component. For the purpose of such topological sorting, we need to first introduce two concepts: level of component, and layer of field. Algorithm-2 is to determine the level of all the components in a given knowledge hierarchy, as well as the layer of each of the fields associated with the components.

Algorithm-2 (Determine level of component and layer of field):

- 1) For a field f that receives input data directly from application, set $\text{layer}(f) = 1$, where $\text{layer}(f)$ is the layer of f ;
- 2) For a field f that serves as the output field of component C , set $\text{layer}(f) = \text{level}(C) + 1$, where $\text{level}(C)$ is the level of C ;
- 3) For a component C with h ($h \geq 1$) input fields $f_{c1}, f_{c2}, \dots, f_{ch}$, set $\text{level}(C) = \max[\text{layer}(f_{c1}), \text{layer}(f_{c2}), \dots, \text{layer}(f_{ch})]$.

The level of components and the layer of fields in the example given in Figure 16 are shown in Table 3-a and Table 3-b, respectively.

Except usual tree structure, in inference structure there are some special graph structures which need special handling by inference engine: 1) multiple parents, and 2) cross layer. For example, in Figure 16 component D and E are the parents of component C , component B of level-1 passes its result to component E of level-4. The order of execution is determined by a topological sorting according to level of components. The key issue here is the data consistency.

Component	Level
A, B	1
C	2
D	3
E	4

(a) The level of components

Field	Layer
F1, F2	1
F3, F4	2
F5	3
F6	4
F7	5

(b) The layer of fields

Table 3. The level of components and the layer of fields for Figure 16

4.2. Protocol between Components

The protocol between components is described from three aspects: syntax, semantics and data type. With the general classes of intelligent components defined, we have syntactical rules indicating the possible connections between different classes. For instance, a component of *Confirmation* class is allowed to send its output to the input of a component of *Decision* class, but not allowed to do the same to the input of a component of *Filtering* class.

For each allowable connection between classes, we further set semantic rules with more details to specify legal connections. A *Filtering* component may connect to another *Filtering* component syntactically. However, there may be semantic constraints based on the detailed types of knowledge used in each *Filtering* component. For instance, a *Dictionary* component can be the support (child component) for a *List* component, but the reverse case does not hold true.

At the component-to-component level, there are four kinds of protocol for the data type of implementation.

- 1) single-to-single: a singleton data is connected to an input field of singleton.
- 2) single-to-multiple: a singleton data is connected to an input field of vector.
- 3) multiple-to-single: a vector data is connected to an input field of singleton.
- 4) multiple-to-multiple: a vector data is connected to an input field of vector.

4.3. Forward Inference with Partial Feedback

It is always desired to get a “better” solution when knowledge-based processing involved in an intelligent component can provide multiple candidates of solution for output. In order to fulfil this purpose, the inference in KBS constructed by KWS is a forward inference with partial feedback. When a component receives inputs, it executes and generates the result as output. As a typical scenario, a component generates inference result and passes the result to its parent(s), and receives *Rerun* signal from parent(s) to provide next possible result when the previously submitted result is found unsatisfactory. The *rerun* mechanism provides a possible way to extend the forward inference mechanism in KBS. Final result will only be generated when the inference is successful.

The inference in KBS constructed by KWS is basically a forward inference. As the simple case when there is no feedback considered, the inference flow starts from layer-1 receiving input data directly from application, goes up for the level-1 components to execute and provide result as layer-2, and then further goes up for the level-2 components to execute, ..., finally has the last level components execute to provide result as the last layer, which represents the inference result.

For a more general case when there is partial feedback introduced, if a component of level- k ($k = 2, 3, \dots$) finds some of the input from its child component unsatisfactory, it will send a *Rerun* signal to the corresponding child component, and the current execution will be pulled back down to the level of the child component accordingly. When there are several components send *Rerun* signal to their child components, the current execution will be set as the level that is the lowest among the levels of components that received *Rerun* signal.

Considering again the example given in Fig. 16, if component *C* sent a *Rerun* signal to component *B*, then the current execution will be pulled back to level-1 for *B* to execute its function again to generate next possible results. With a similar spirit, if component *E* sent a *Rerun* signal to component *B*, then the current execution will also be pulled back to level-1. It is important to notice that there are other two components *C* and *D* at a higher level than *B*, and with the *Rerun* signal sent to *B*, any execution starting from *C* and *D* will be frozen temporarily to ensure the data consistency.

4.3.1 States of Component

In order to indicate the execution status of a component, we introduce *state of component*. The transition between states is shown in Fig. 17 and the explanation is listed in Table 4.

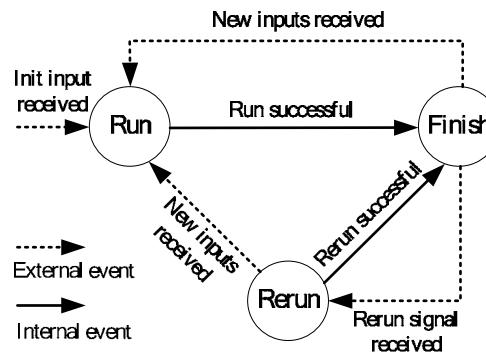


Fig. 17. The states of a component

State	Explanation
Run	The component will execute its function, if successful; the results will be sent to its parent, if unsuccessful, the component will send a rerun signal to its child components.
Rerun	The component will execute its rerun function, trying to generate next possible results. If successful, the results will be sent to its parent, if unsuccessful, the component will send a rerun signal to its child components.
Finish	The run or rerun of component is successful.

Table 4. The explanation of component states

The state of an output field as same as the corresponding component which sends result to the field, the explanation of field states is listed in Table 5.

State	Explanation
Run	Representing a field "waiting for obtaining result".
Rerun	Representing an output field "waiting for obtaining new result of rerun".
Finish	Representing a field "finished obtaining result".

Table 5. The explanation of field states

Example 6: Consider a scenario of execution on the inference structure given in Fig. .

Time-1: The current execution is at level-1, and Field F1 and F2 received inputs from layer-1, and component A and B executed and provided result as layer-2, finally, component A and B updated their state to be "Finish", and the current execution is updated to be at level-2.

Time-2: The current execution is at level-2. Assume that the result from component B is unsatisfactory for component C. C sent the *Rerun* signal to B to get next possible inputs, B received the *Rerun* signal, and its state is updated to "Rerun", and the current execution is pulled back down to level-1.

Time	Level	Component				
		A	B	C	D	E
1	1	R	R	R	R	R
2	2	F	F	R	R	R
2		Assume that, C sent rerun signal to B				
3	1	F	RR	R	R	R
3		Assume that, B generate next possible result				
4	2	F	F	R	R	R
5	3	F	F	F	R	R
6	4	F	F	F	F	R
6		Assume that, E sent rerun signal to B, C, D				
7	1	F	RR	RR	RR	R
7		Assume that, B generate next possible result				
8	2	F	F	R	RR	R
9	3	F	F	F	R	R
10	4	F	F	F	F	R
11	5	F	F	F	F	F
11		The final result is in F7				

R – Run; RR – Rerun; F – Finish

Table 7. An example of inference flow

Time-3: Component B executed and provided next possible results as layer-2.

Time-4, Time-5: Continued the inference in the same manner.

Time-6, Time-7: The situation is similar as Time-3.

Time-8, Time-9, Time-10, Time-11: Continued the inference.

Finally, the final result is in F7 at layer-5.

Table 7 lists out the state change of components.

4.3.2 Feedback Handling

Algorithm-3 and Algorithm-4 provide forward inference with partial feedback. The procedure execution() in Algorithm-3 is to call the specific component for knowledge-based processing. When the inference engine calling execution(C_r), it passes the control to the component C_r and waits for the return of execution result. When a feedback of reasoning is considered, necessary interruption should be introduced to adjust the execution sequence. Algorithm-3 does the main job of execution control but calls Algorithm-4 (PartialRerun) to monitor feedback handling.

Considering an inference carried out in a KBS constructed by KWS, when an intelligent component failed to work out a solution as its output with its local knowledge source, an effort is expected to "bring back" the process to those field(s) or component(s) that provided the input(s) to it. It introduces a need of bidirectional inference within part of the knowledge hierarchy. This is achieved by the *Rerun* control of the KWS inference engine.

A component under *Rerun* state means it is not successful in the previous run of inference and needs 'redo' the task to provide new (better) result. The handling of rerun starts by asking new input from child component(s), according to the type of protocol between an input field of component C_r currently under *Rerun* and the output field of its corresponding child component C_c . The main algorithm is given in Algorithm-3 with further implementation details omitted.

Algorithm-3 (Control of execution, with $k \geq 2$ components):

- 1) Get input data for all the layer-1 fields, and
set them as of *Finish*;
Set all other components and fields as of *Run*;
 - 2) **While** (not all the components are of *Finish*)
 - 2-1) If there are any components C_j , $2 \leq j \leq k$, currently under *Rerun*
Then set currentFrozen := $\min_{j \in [2, k]} level(C_j)$
Else
set currentFrozen := $1 + \max_{i \in [1, k]} [level(C_1), level(C_2), \dots, level(C_k)]$;
 - 2-2) Check component C_r ($2 \leq r \leq k$) in the execution list
following nondecreasing order of level:
 - 2-2-1) If C_r is of *Run* and
 $level(C_r) < currentFrozen$ and
all of its input field(s) are of *Finish*
Then If execution (C_r) /* successful */
Then set C_r and its output field as of *Finish*;
Else set C_r and its output field as of *Rerun*;
 - 2-2-2) Else If C_r is of *Rerun* and
 $level(C_r) \leq currentFrozen$ and
all of its input field(s) are of *Finish*
Then call PartialRerun(C_r).
/* else next component */
- /* end of while */

When the inference engine calling `getNext(Cc)`, it gives a signal to ask for the next possible output from C_c . The procedure `reExecution()` does a similar job as `execution()`, but calls the component to provide next new result (if any) with the same previous input data. The KWS inference engine tries to get new input data for the component C_r currently under *Rerun*, through either `getNext()` or `reExecution()`. When the effort of getting new input data from its child component C_c is successful either through `getNext()` or `reExecution()`, all the ancestor component(s) of C_c as well as their output fields will be updated to *Run* state by calling `setRunAncestor()` to clean up the result of previous run. As long as one of the child component of C_r could provide new input successfully, C_r will be of *Run* again, otherwise it will remain as of *Rerun* and all its child components as well as their output fields will be set as of *Rerun*.

In case that a single child component is supporting multiple parent components, a data inconsistency should be avoided when partial feedback and rerun are considered. This consistency is guaranteed by indicating the current frozen area. A component C_c being required for a ‘rerun’ by one of its parent components C_r will cause a temporary ‘frozen’ execution to other related components. A ‘frozen’ execution affects two groups of components: (a) all components of *Run* state at a level equal to or higher than `currentFrozen`; (b) all components of *Rerun* state at a level higher than `currentFrozen`.

Algorithm-4 (Partial Rerun from C_r):

```

For each of the  $h$  ( $h \geq 1$ ) input fields of  $C_r$ :  $f_{cr1}, f_{cr2}, \dots, f_{crh}$ ,
  If it is the output field of some child component  $C_c$ 
    Then Check the protocol connection from  $C_c$  to  $C_r$ :
      Case: multiple-to-single
        If getNext(Cc) /* successful */
          Then set  $C_c$  and its output field as of Finish;
            setRunAncestor(Cc);
            Return;
        Else If reExecution(Cc) /* successful */
          Then set  $C_c$  and its output field as of Finish;
            setRunAncestor(Cc);
            Return;
          /* end of case multiple-to-single */

      Case: single-to-single:
      Case: single-to-multiple:
      Case: multiple-to-multiple:
        If reExecution(Cc) /* successful */
          Then set  $C_c$  and its output field as of Finish;
            setRunAncestor(Cc);
            Return;
        /* check next input field of  $C_r$  */
      /* end of 1st for */

    For each of the  $h$  ( $h \geq 1$ ) input fields of  $C_r$ :  $f_{cr1}, f_{cr2}, \dots, f_{crh}$ ,
      If it is a direct input from application
        Then stop processing and report failure
      Else /* it is the output field of some child component  $C_c$  */
        set  $C_c$  and its output field as of Rerun;
      /* end of 2nd for */
  
```

5. Conclusions

We have introduced the KWS as a framework of development tool for developers to model and develop their customized KBS, provided the processing flow of KWS in constructing a KBS, and discussed the major sub-systems of KWS, including KWS inference engine, intelligent editor, KDL processor, and installer.

The inference in KBS constructed by KWS is a truth value flow inference (TVFI) realized at two levels simultaneously: the content level of inference that relies only on the knowledge sources stored "locally" in individual intelligent components, and the truth (confidence) level of inference that contributes to the confidence flow throughout the entire KBS. We have discussed the mechanism of TVFI as well as its implementation. The interval-valued confidence (IVC) has been adopted for the representation of imprecision and uncertainty in KBS constructed by KWS. An IVC is represented by a fuzzy number defined as a fuzzy subset of [0, 1]. Basic logic operations with IVC have been defined and their properties discussed. Based on the concepts of TVFI and IVC, confidence transfer with different types of intelligent component and corresponding interpretability has also been discussed. KWS inference engine has been explained in detail with the control algorithms of execution order of components for a forward inference with partial feedback, the management of protocols, and the handling of imprecision with TVFI and IVC.

Further effort will be put in handling knowledge imprecision with different types of intelligent processing and their integration in hybrid intelligent systems.

6. Acknowledgement

This work was supported in part by the Macao Science and Technology Development Fund under grant 048/2006/A.

7. References

- L. Ding and Z. Shen (1994). Neural Network Implementation of Fuzzy Inference for Approximate Case-based Reasoning, In: *Neural and Fuzzy Systems: The Emerging Science of Intelligence and Computing*, Mitra, Sunanda.; Gupta, Madan M.; and Kraske, Wolfgang, 28-56, SPIE Press
- L. Ding, H.H. Teh, P.Z. Wang, and H.C. Lui (1996). A Prolog-like inference system based on neural logic, *Fuzzy Sets and Systems*, Vol. 82, No. 2, 235-251
- L. Ding and H.C. Lui (1999), A Knowledge-based Approach Applied in Intelligent Hand Written Form Processing, *Journal of Advanced Computational Intelligence*, Vol. 3, No. 3, 193-199
- L. Ding (2007a). A Model of Hierarchical Knowledge Representation - Toward Knowware for Intelligent System. *Journal of Advanced Computational Intelligence & Intelligent Informatics*, Vol. 11, No. 10, pp. 1232-1240
- L. Ding (2007b). Design and development of knowware system. Proceedings of 2nd International Conference on Innovative Computing, Information and Control (ICICIC'2007), pp. 17-17, Kumamoto, Japan.
- L. Ding and S. Nadkarni (2007). Automatic Construction of Knowledge-Based System Using Knowware System. *Proceedings of 6th International Conference on Machine Learning and Cybernetics*, , pp. 789-794, Hong Kong, China

- L. Ding (2008). Inference in Hybrid KBS with Interval-Valued Confidence. *Proceedings of 2008 IEEE World Congress on Computational Intelligence / 2008 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2008)*, pp.1350-1357, Hong Kong, China.
- L. Ding and S.L. Lo (2008). Truth Value Flow Inference in Hybrid KBS Constructed by KWS. *Proceedings of 3rd International Conference on Innovative Computing Information and Control (ICICIC'2008)*, pp. 311-314, Dalian, China
- J.-S.R. Jang, C.-T. Sun and E. Mizutani (1997). Neural-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence, Prentice Hall, NJ.
- Z. Shen and L. Ding (1994). A Representation of Exponential Form on Fuzzy Logic. *Fuzzy Sets and Systems*, Vol. 68, pp.267-280
- P.Z. Wang and H.M. Zhang (1993). Truth value flow inference and its mathematical theory, In: *Between Mind and Computer*, Eds. P.Z. Wang, K.F. Loe, 325-358, World Scientific, Singapore.
- R.R. Yager and D.P. Filev (1994), Essentials of Fuzzy Modeling and Control, John and Wiley and Sons, Inc., NJ
- L.A. Zadeh (1975), The concept of a linguistic variable ans its applicatin to approximate reasoning: Parts 1, 2 and 3, *Informntion Sciences*, 8, pp.199-249; 8, pp.301-357; 9, pp.43-80.

Applying Fuzzy Bayesian Maximum Entropy to Extrapolating Deterioration in Repairable Systems

Chi-Chang Chang

Chung Shan Medical University

Taiwan

Ruey-Shin Chen

National Kinmen Institute of Technology

Taiwan

Pei-Ran Sun

Chung Shan Medical University Hospital

Taiwan

1. Introduction

In general, most complex systems, which are constructed by collecting more than one part to perform either single or multiple functions, are usually repaired rather than replaced after failures, since such systems can be restored to fully implement the required functions by methods other than replacing the entire system (Ascher & Feingold, 1984). However, the successive times between failures are not necessarily identically distributed, as in renewal processes. More generally, they can become smaller (an indication of deterioration), or conversely larger and larger (an indication of reliability growth) (Barlow & Proschan, 1965). If deterioration is detected, then the decision of when to overhaul or discard the system, given the costs of repairs and failures, is of fundamental importance.

At the time of the decision, the degree of future deterioration, which is likely to be uncertain, is of primary interest for the decision maker. Decision analysis seems to be able to provide methods to deal with such uncertain situation. However, the decision structure is usually formulated in such a way as to imposingly quantify particular qualitative characteristics on human being as decision rules (Freeling, 1984). The most important of these characteristics is that of human ability to precisely specify numerical values of ends and means in a decision process. These non-fuzzy decision rules are useful; however, they are limited in their applicability to real world situations where nearly all real human decision problems are imprecise, ill-definedness and vagueness (Dompere, 1982). In such situations, decision-making depends on numerous factors which limit human ability and increase difficulties to deal with (Asai & Okuda, 1975). Therefore, the use of fuzzy method may be very helpful in solving the decision making problems of deteriorating repairable systems (Bellman & Zadeh, 1970). In section 2, describes some related researches about modeling deteriorating

issue. In addition, the nonhomogeneous Poisson process (NHPP) was introduced. In section 3, delineates the implementation procedure for fuzzy Bayesian decision process. For the sake of information value analysis, we will discuss Bayesian decision process when the collected information is assumed to be fuzzy. Section 4, we will describes some fuzzy aggregation operations researches method and process. Further, section 5 a case study to illustrate the use of the models developed in the previous sections. Finally, section 6 discusses the work items and contributions of this study.

2. Deteriorating Repairable Systems

In order to model deterioration in repairable systems, the Non-Homogeneous Poisson Process (NHPP) was introduced since it seems more plausible for systems consisting of many components (Härtler, 1989). The system failure process is time-dependent and its intensity function of the failure process is usually assumed to be of the form $\lambda(x)=\lambda_0 h(\beta; x)$, where λ_0 is the scale factor, β is the deteriorating rate, x is the elapsed time, and $h(\cdot)$ can be any function that reflects the deteriorating process. Suppose we have a system whose failure process is given by a non-homogeneous Poisson process and with a power law intensity function of the form (Ascher & Feingold, 1984):

$$\lambda(x) = \lambda_0 \beta x^{\beta-1}, \lambda_0 > 0, \beta > 0 \quad (1)$$

The likelihood function of the first $N=n$ failure times for the case of time-truncated data is given by

$$L(x_1, x_2, \dots, x_n | \lambda_0, \beta) = \lambda_0^n \beta^n (\prod_{i=1}^n x_i)^{\beta-1} \exp(-\lambda_0 x_n^\beta) \quad (2)$$

Huang and Bier (1998) proposed a natural conjugate prior distribution for the power law failure model, which is given by

$$f(\lambda_0, \beta) = K \lambda_0^{m-1} \beta^{m-1} \{ \exp(-\alpha) y_m^m \}^{\beta-1} \exp(-\lambda_0 c y_m^\beta) \quad (3)$$

Comparing with other approaches, this natural conjugate prior distribution has many desirable properties, which are summarized as follows:

- (1) The marginal distribution of β is a gamma distribution with parameters m and α , expectation $\mu_\beta = m/\alpha$, and coefficient of variation $CV_\beta = 1/\sqrt{m}$.
- (2) The conditional distribution of λ_0 given β is a gamma distribution with parameters m and $c y_m^\beta$.
- (3) The expectation of λ_0 is given by $\mu_{\lambda_0} = \frac{c}{m} \left(\frac{\alpha}{\alpha + z_m} \right)^m$, where $z_m = \ln(y_m)$.
- (4) The coefficient of variation of λ_0 is given by $CV_{\lambda_0} = \sqrt{\frac{\eta^m (m+1)}{m} - 1}$, where $\eta = \frac{\alpha}{\alpha + z_m}$.

$$\eta = 1 + \frac{z_m^2}{\alpha^2 + 2\alpha z_m}, \text{ and } z_m = \ln(y_m).$$

These properties provide guidance on how to choose the parameters α , m , y_m and c to achieve joint distributions with the desired prior moments. For example, m can be chosen to give the desired value of CV_β , α can be selected to give the desired value for μ_β , y_m can be selected to give the desired value for CV_{λ_0} and c can be selected to give the desired value for μ_{λ_0} .

3. Fuzzy Bayesian Decision Process

Generally, the information is expressed by means of numerical values in a quantitative setting (Crow, 1974). However, in a qualitative setting, which is filled with vague or imprecise knowledge, the information cannot be estimated with an exact numerical value (Herrera & Herrera, 2000). In such case, a more realistic approach may be used for linguistic assessments instead of numerical values, that is, to presume that the variables involved in the problem are assessed by means of linguistic terms (Hisdal, 1984; Tong & Bonissone, 1984). This approach is appropriate since it allows a representation of information in a more realistic and adequate form when precision is not achievable.

In real Bayesian decision problems there are two different types of vagueness: states and information (Fruhwirth-Schnatter, 1993). In this section we will discuss decision models for the cases in which only the states are fuzzified and both the states and additional information are fuzzified (Roubens, 1997; Stephen & Donnell, 1979). Finally, we will also present the decision flowchart and the decision analysis process.

As mentioned in the previous section, Huang and Bier (1998) developed the joint natural conjugate prior distributions for λ_0 and β . This proposed conjugate prior distribution provides guidance on how to choose the parameters α , m , y_m and c by collecting the experts' knowledge and observed data about the prior moments (i.e., μ_β , CV_β , μ_{λ_0} , CV_{λ_0}). In other words, in order to apply the joint natural conjugate prior distribution into the decision process, the experts need to specify a specific value for each of these four moments, respectively. However, this is not necessarily realistic for real world cases, since it is usually a tough task for the experts to specify a value to a prior moment with sufficient confidence. Instead, the experts often think of a prior moment as a fuzzy number, that is, a range of numbers and each number within the range has a different membership value. Furthermore, since the prior moments are not exactly provided by the experts, the parameters (i.e., α , m , c , and y_m) are therefore formed as the functions of these fuzzy numbers. From the properties of the joint natural conjugate prior in the previous section, we can have (Huang & Chang, 2004)

$$\alpha \equiv \alpha(\mu_\beta, CV_\beta) = \frac{1}{\mu_\beta CV_\beta^2} \quad (4)$$

$$m \equiv m(CV_\beta) = \frac{1}{CV_\beta^2} \quad (5)$$

$$\begin{aligned} y_m &\equiv y_m(\mu_\beta, CV_\beta, CV_{\lambda_0}) \\ &= \exp\{\alpha[\varphi^{\frac{1}{m}}(CV_\beta, CV_{\lambda_0}) - 1 + \sqrt{[\varphi^{\frac{1}{m}}(CV_\beta, CV_{\lambda_0}) - 1]\varphi^{\frac{1}{m}}(CV_\beta, CV_{\lambda_0})}]\} \end{aligned} \quad (6)$$

and

$$c \equiv c(\mu_\beta, CV_\beta, \mu_{\lambda_0}, CV_{\lambda_0}) = \frac{m\mu_{\lambda_0}}{\left(\frac{\alpha}{\alpha + \ln(y_m)}\right)^m} \quad (7)$$

$$\text{where } \varphi(CV_\beta, CV_{\lambda_0}) = \frac{CV_{\lambda_0}^2 + 1}{CV_\beta^2 + 1}$$

Based on the above discussion and Equations (4) to (7), we start the discrimination problem with fuzzy states space $F=(\mu_\beta, CV_\beta, \mu_{\lambda_0}, CV_{\lambda_0})$ and exact observation space $X=\{x\}$ (Dompere, 1982). Consider a sequence of observations $x(n)=(x_1, x_2, \dots, x_n)$, therefore the fuzzy prior distribution of the fuzzy state F_j for the deteriorating repairable system is given by

$$\begin{aligned} f(F_j) &= K \int_{\mu_\beta} \int_{CV_\beta} \int_{\mu_{\lambda_0}} \int_{CV_{\lambda_0}} \lambda_0^{m-1} \beta^{m-1} \{exp(-\alpha)y_m^m\}^{\beta-1} exp(-\lambda_0 cy_m^\beta) \\ &\quad \chi_{F_j}(\mu_\beta) \chi_{F_j}(CV_\beta) \chi_{F_j}(\mu_{\lambda_0}) \chi_{F_j}(CV_{\lambda_0}) d\mu_\beta dCV_\beta d\mu_{\lambda_0} dCV_{\lambda_0} \end{aligned} \quad (8)$$

where $\chi_{F_j}(r)$ ($r = \mu_\beta, CV_\beta, \mu_{\lambda_0}$, or CV_{λ_0}) denotes the membership functions for the four fuzzy numbers, respectively, and $f(F_j)$ is the fuzzy integral over the fuzzy hyper space of four dimensions. Similar argument can be applied to the fuzzy posterior distribution derived from the fuzzy state F_j along with the additional exact data for the deteriorating repairable system, and which is given by

$$\begin{aligned} f(F_j | x_1, x_2, \dots, x_n) &= K' \int_{\mu_\beta} \int_{CV_\beta} \int_{\mu_{\lambda_0}} \int_{CV_{\lambda_0}} \lambda_0^{m+n-1} \beta^{m+n-1} \{exp(-\alpha)y_m^m \prod_{i=1}^n x_i\}^{\beta-1} exp[-\lambda_0(cy_m^\beta + x_n^\beta)] \\ &\quad \chi_{F_j}(\mu_\beta) \chi_{F_j}(CV_\beta) \chi_{F_j}(\mu_{\lambda_0}) \chi_{F_j}(CV_{\lambda_0}) d\mu_\beta dCV_\beta d\mu_{\lambda_0} dCV_{\lambda_0} \end{aligned} \quad (9)$$

where (x_1, x_2, \dots, x_n) means we observe the additional exact data until the nth failures. Furthermore, in the part of observe additional fuzzy data, the fuzzy likelihood function of observing the nth failure times is given by

$$\begin{aligned} Lik(x'_1, x'_2, \dots, x'_n | \lambda_0, \beta) &= \int_{X_1} \dots \int_{X_n} \lambda_0^n \beta^n (\prod_{i=1}^n x'_i)^{\beta-1} exp(-\lambda_0 x'_n) \chi_1(x'_1) \dots \chi_n(x'_n) dx'_1 \dots dx'_n \end{aligned} \quad (10)$$

where $(x'_1, x'_2, \dots, x'_n)$ means we observe additional fuzzy data until the n th failures, and $\chi_i(x'_i) (i=1,2,\dots,n)$ denotes the membership functions for the n fuzzy failure times, respectively. If we observe fuzzy state F_j and additional fuzzy data $(x'_1, x'_2, \dots, x'_n)$, then the posterior distribution will satisfy

$$\begin{aligned} & f(F_j | x'_1, x'_2, \dots, x'_n) \\ &= K'' \int_{\mu_\beta CV_\beta} \int_{CV_{\lambda_0}} \int_{X_1} \dots \int_{X_n} \lambda_0^{m+n-1} \beta^{m+n-1} \{ \exp(-\alpha) y_m^m \prod_{i=1}^n x_i^{\beta-1} \exp[-\lambda_0(cy_m^\beta + x_i^\beta)] \} \\ & \quad \chi_1(x'_1) \dots \chi_n(x'_n) \chi_{F_j}(\mu_\beta) \chi_{F_j}(CV_\beta) \chi_{F_j}(\mu_{\lambda_0}) \chi_{F_j}(CV_{\lambda_0}) \\ & \quad dx'_1 \dots dx'_n d\mu_\beta dCV_\beta d\mu_{\lambda_0} dCV_{\lambda_0}. \end{aligned} \quad (11)$$

Note that K , K' and K'' in Equations (8), (9) and (11) are normalizing factors. Figure 1 shows a fuzzy Bayesian analysis in deteriorating repairable systems with experts' prior knowledge. The same decision elements proposed by Huang (2001) of a Bayesian decision analysis for a deteriorating repairable system are as follows:

- (a) Parameter space $\mathcal{O}: \{(\lambda_0, \beta) | \lambda_0 > 0\}$.
- (b) Action space $A: \{a_1, a_2\}$, where a_1 is the status quo, and a_2 is the risk reduction action.
- (c) Loss function L : a real function defined on $\mathcal{O} \times A$. If we decide to keep the system operating, then the loss we face is $L(\theta, a_1)$; if we decide to take the risk reduction action, then the loss we face is $L(\theta, a_2)$.
- (d) Sample space X : The additional information available to be collected (e.g., successive failure times). The cost of collecting this additional data or information should also be reflected in the decision process.

We assume (i) that the status of system after a repair is essentially the same as it was immediately before failure occurred (as good as old); and (ii) that the repair times can be neglected. The following terminology will be used in the decision analysis process:

C_F : the cost of a failure if it occurs.

C_R : the cost of the proposed risk reduction action.

C_I : the cost of collecting additional information.

ρ : the reduction in failure rate that would result from the proposed risk reduction action ($0 < \rho < 1$).

T : the time horizon under consideration.

t : the time at which the decision is being made.

Ψ : the expected number of failures during the time period $[t, T]$ under the status quo.

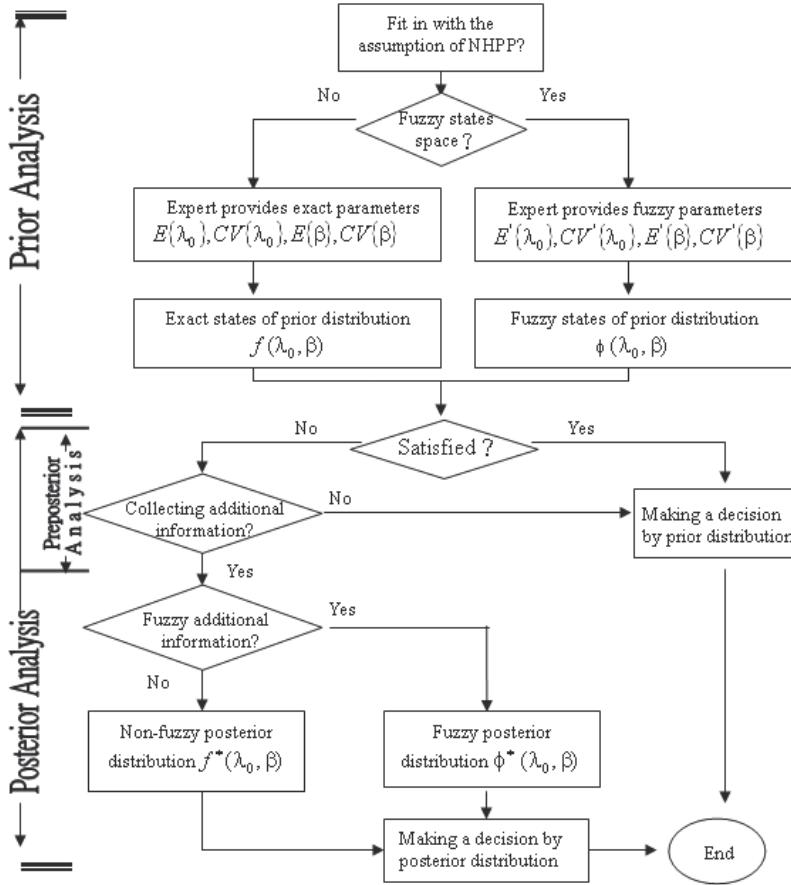


Fig. 1. Flowchart for Fuzzy Bayesian Decision Process

Suppose that the repairable system has a planned lifetime (i.e., time horizon) T , and the decision of whether to maintain the status quo or perform some risk reduction action must be made at time t . The decision variable we are dealing with is then the expected number of failures during the time period $[t, T]$. Since the system failure times are assumed to be drawn from a non-homogeneous Poisson process with power law intensity function, the expected number of failures in $[t, T]$ under the status quo is given by

$$\Psi = \Psi(T, t, \lambda_0, \beta) = \int_t^T \lambda(s) ds = \int_t^T \lambda_0 \beta s^{\beta-1} ds = \lambda_0 (T^\beta - t^\beta) \quad (12)$$

Suppose that the risk reduction action will reduce the failure intensity by a fraction ρ , where $0 < \rho < 1$. Then the expected number of failures in $[t, T]$ if the risk reduction action is performed is given by

$$\int_t^T \lambda(s)(1-\rho) ds = (1-\rho)\Psi \quad (13)$$

On the basis of the assumptions given above, we therefore have a two-action problem with a linear loss function, where the loss for taking action a_1 (i.e., continuing with the status quo) is $C_F\Psi$ and the loss for taking action a_2 (i.e., undertaking the risk reduction action) is $C_F(1-\rho)\Psi + C_R$. The expected loss for the status quo is simply $C_F E\{\Psi\}$, and the expected loss for the risk reduction action is $C_F(1-\rho)E\{\Psi\} + C_R$. Since the fuzzy prior and posterior density functions for Ψ are available by using defuzzified techniques and bivariate transformation for Equations (8), (9) and (11), respectively, the prior and posterior mean values of Ψ can be evaluated. Therefore, Fuzzy Bayesian decision analyses can be performed by comparing the prior and posterior mean values of Ψ with the cutoff value $\Psi_C = C_R/(C_F\rho)$. If the relevant mean is smaller than Ψ_C , then we should keep the system operating as in the status quo; if not, then we should perform the risk reduction action.

4. Fuzzy Aggregation Operation Methods

Generally, changing one's beliefs when new information becomes available is a common mode of human reasoning. It is observed in the deliberate gathering of pertinent evidence during industrial troubleshooting, or medical diagnosis and so on. In another word, if one can make independence assumptions, many of the problems disappear, and in fact, this is often the method of choice even when it is obviously incorrect. There are different methodologies for dealing with this problem, e.g., maximal entropy and Dempster-Shafer Theory (Oberkampf et al., 2004). However, it still left some the Challenge Problems to solve, these questions were (Ferson et al., 2004; Fetzer & Oberguggenberger, 2004): (1) How should epistemic uncertainty about a quantity be represented? (2) How can epistemic and aleatory uncertainty about a quantity be combined and propagated in calculations? (3) How should multiple estimates of uncertain quantities be aggregated before calculation? (4) How should the technical issue of repeated uncertain parameters be handled in practical calculations? (5) How might various approaches be adapted for use in practical calculations based on sampling strategies?

This section reviews the (1) to (3) of five technical issues addressed by the Challenge Problems that are commonly involved in computational problems involving epistemic uncertainty. In a sense, this is a problem of too much information because it means the analyst must decide how to combine this information before proceeding with the analysis. In point of problem, we will examine the fuzzy entropy aggregation operators in two ways: through the fuzziness of the prior moments $\mu_\beta, CV_\beta, \mu_{\lambda_0}, CV_{\lambda_0}$ and through the fuzziness of failure data set. A fuzzy number is not a measurement. In other word, a fuzzy number is a subjective valuation assigned by one or more human operators. In addition, defuzzification methods have been widely studied for several years and were applied to fuzzy arithmetic (Kandel, 1986; Kim et al., 1998; Ma et al., 2002). The major idea behind these methods was to obtain a typical value from a given fuzzy set according to some specified characters, such as central gravity, median, etc. In other words, each defuzzification method provides a correspondence from the set of all fuzzy sets into the set of real numbers (Roychowdhury & Pedrycz, 2001). Therefore, in order to transfer the subjective valuation into real valuation, we have to use the fuzzy concept of entropy measure method (Chang, 2008). It should lend itself to probabilistic updating formulas by allowing heuristic estimation of the degree of

independence. After describing these formulae, one example illustrates how fuzzy entropy might be applied.

4.1 Uncertainty measure of Entropy

Entropy is a measure of the amount of uncertainty in the outcome of a random experiment, or equivalently, a measure of the information obtained when the outcome is observed. This concept has been defined in various ways (Shannon, 1948; Renyi, 1961; Kosko, 1986; Pal & Chakraborty, 1986) and generalized in different applied fields, such as communication theory, mathematics, statistical thermodynamics, and economics (Belahut, 1987; Cover & Thomas, 1992; Ching et al., 1995). Of these various definitions, Shannon contributed the broadest and the most fundamental definition of the entropy measure in information theory. In Shannon's entropy, entropy can be considered as a measure of the uncertainty of a random variable x . Let x be a discrete random variable with a finite alphabet set containing N symbols given by $\{x_0, x_1, \dots, x_{N-1}\}$. If an output x_j occurs with probability $p(x_j)$, then the amount of information associated with the known occurrence of output x_j is defined as

$$I(x_j) = -\log_2 p(x_j) \quad (14)$$

That is, for a discrete source, the information generated in selecting symbol x_j is $-\log_2 p(x_j)$ bits. On average, the symbol x_j will be selected $n \cdot p(x_j)$ times in a total of N selections, so the average amount of information obtained from n source outputs is

$$-n \cdot p(x_0) \log_2 p(x_0) - n \cdot p(x_1) \log_2 p(x_1) \cdots - n \cdot p(x_{N-1}) \log_2 p(x_{N-1}) \quad (15)$$

Dividing (15) by n , we obtain the average amount of information per source output symbol. This is known as the average information, the uncertainty, or the entropy, and is defined: The entropy $H(X)$ of a discrete random variable x is defined as (Shannon, 1948)

$$H(X) = -\sum_{j=0}^{N-1} p(x_j) \log_2 p(x_j) \quad (16)$$

$$\text{or } H(X) = -\sum_{j=0}^{N-1} p_j \log_2 p_j \quad (17)$$

where p_j denotes $p(x_j)$.

Hence, entropy is a function of the distribution of X . Further, this amount of information is estimated by the average weighted information provided by the expected probabilities of occurrence of the events as follows (Kosko, 1986):

$$-\sum_{i=1}^n p_i \cdot \ln p_i \quad (18)$$

where p_i is the probability of occurrence of event i and n is the number of events.

Equation (18) is also called Entropy as its form suggests, one can realize that when maximizing entropy, not only the probabilities of the events will affect the quantity of information, but also the number of events will cause certain impacts. Since these events will provide information, they are the factors concerning the information of the system and thus, their probabilities of occurrence are the weights of importance of these factors.

Several important properties regarding this Entropy model (Kosko, 1990; Kosko, 1997) which will be quoted by as below:

- (1) The objective function is a continuous function of p_1, p_2, \dots, p_n . Therefore, small changes in p_1, p_2, \dots, p_n will causes small changes in H_n . This means that information provided by factor i will be changed when the probability of occurrence of factor i changes.
- (2) H_n is a symmetric function of its arguments. Therefore, the amounts of information will not be changed by the different orders of factors.
- (3) $H_{n+1}(p_1, p_2, \dots, p_n, 0) = H_n(p_1, p_2, \dots, p_n)$. Thus, the amount of information is not changed if an impossible outcome is added to the probability scheme. That is, if a factor i with probability of occurrence equal 0, it will not give any contribution to the expected information and thus it can be deleted.
- (4) H_n will be reach the maximum when $p_1 = p_2 = \dots = p_n = 1/n$, and the maximum information by giving the outcomes equal probabilities of occurrence when the maximum uncertainty is faced.
- (5) The maximum value of H_n equals $\ln(n)$. The maximum value of H_n increase as n increased. So, when we investigate more factors of a system, the expected information about the system will increase.

In what follows, we will propose the fuzzy entropy measure which is an extension of Shannon's definition.

4.2 Fuzzy Entropy

In general, the membership function of a fuzzy set is determined by the users subjectively, which means that the membership function specified for the same concept by different persons may vary considerably. The shapes of the membership functions always present the knowledge grade of the elements in the fuzzy sets for the users (Zadeh, 1983). In other words, every membership function also presents the fuzziness of the corresponding fuzzy set in the idea of users. Therefore, it is necessary for us to have some measurements to measure the fuzziness of fuzzy sets. According to Szmidth and Kacprzyk (2000), fuzziness, a feature of imperfect information, results from the lack of crisp distinction between the elements belonging and not belonging to a set (i.e. the boundaries of the set under consideration are not sharply defined). A measure of fuzziness often used and cited in the literature is entropy first mentioned in 1965 by Zadeh (1965). The name entropy was chosen due to an intrinsic similarity of equations to the ones in the Shannon entropy. However, the two functions measure fundamentally different types of uncertainty. Basically, the Shannon entropy measures the average uncertainty in bits associated with the prediction of outcomes in a random experiment. Until now, there are several typical methods to be used to measure the fuzziness of fuzzy sets. In 1972, De Luca and Termini (1972) introduced some requirements which capture human intuitive comprehension of the degree of fuzziness. Kaufmann (1975) proposed that the fuzziness of a fuzzy set can be measured through the

distance between the fuzzy set and its nearest non-fuzzy set. Another way given by Yager (1979) suggested the measure of fuzziness can be expressed by the distances between the fuzzy set and its complement. De Luca and Termini (1972) utilized the conception of the entropy to indicate the fuzziness of a fuzzy set. Kosko (1997) investigated the fuzzy entropy in relation to a measure of subsethood.

In order to elicit expert's knowledge, and advances in numerical methods and computation have made it possible to implement fuzzy Bayesian analysis in ways previously research (Chang & Cheng, 2007). The fuzzy mutual entropy and explores the information theoretic structure of fuzzy cubes was be applied (we will explore it more in detail at following context). The fuzzy mutual entropy of a fuzzy set F acts as a type of distance measure between F and its set complement F^c . The logistic map equates the sum of a real vector's n components with the mutual entropy of some fuzzy set F and its complement F^c . This cube geometry motivates the ratio measure of fuzziness $E(F) = a/b$ (Kosko, 1986), where a is the distance $\ell^1(F, F_{near})$ from F to the nearest vertex F_{near} and b is the distance $\ell^1(F, F_{far})$ from F to the farthest vertex F_{far} . The fuzzy entropy theorem reduces this ratio of distances to a ratio of counts in Equation (19) and Figure 2 shows the fuzzy entropy theorem in the unit square.

$$E(F) = \frac{c(F \cap F^c)}{c(F \cup F^c)} \quad (19)$$

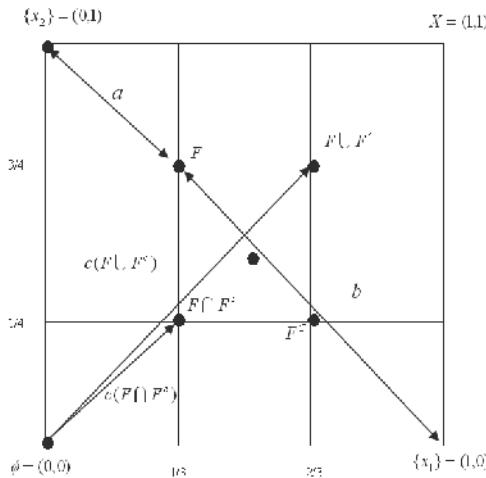


Fig. 2. Geometry of fuzzy entropy theorem (Data Source: Kosko, 1986)

Fuzzy cubes map smooth onto extended real spaces of the same dimension and vice versa. The 2^n infinite limits of extended real space $[-\infty, \infty]^n$ map to the 2^n binary corners of the fuzzy cube I^n . The real origin 0 maps to the cube midpoint. Each real point x will mapping to a unique fuzzy set F as Figure 3 shows.

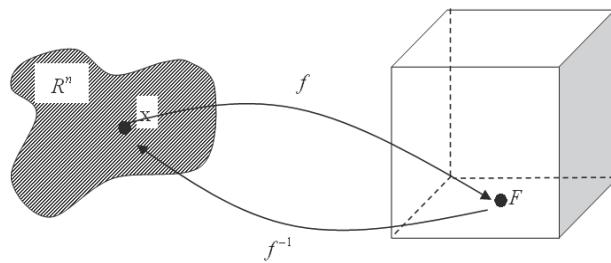


Fig. 3. Diffeomorphism from extended real space to fuzzy space (Data Source: Kosko, 1986)

Fuzzy mutual entropy equals the negative of the divergence of Shannon entropy. Uncertainty descriptions define points in the fuzzy cube parameter space. Versions of both extended Shannon entropy and fuzzy mutual entropy define vector fields on the fuzzy cube. As to the methods of defuzzification, there have been widely studied for decades and effectively utilized to the applications of fuzzy arithmetic (Kandel, 1986; Kim et al., 1998; Ma et al., 2002). The foremost idea behind these methods was to obtain a typical value from a given fuzzy set according to some specified characters, such as central gravity, mean, or median, etc. In other words, each defuzzification method provides a correspondence from the set of all fuzzy sets into the set of real numbers (Roychowdhury & Pedrycz, 2001). Therefore, in order to transfer the subjective valuation into real valuation, we have planning to apply the intuitionistic fuzzy sets and discuss the extension of Luca-Termini Axioms for the measurement of entropy-based defuzzification method. The following will be explored both contents more in detail:

First, a geometric interpretation of intuitionistic fuzzy sets and fuzzy sets is presented in Figure 4 which summarizes considerations presented in (Szmmidt & Kacprzyk, 2000). Basically, an intuitionistic fuzzy set X is mapped into the triangle ABD in that each element of X corresponds to an element of ABD , as an example, a point $x' \in ABD$ corresponding to $x' \in X$ is marked. In Figure 4, this condition is fulfilled only on the segment AB . Segment AB may be therefore viewed to represent a fuzzy set. The orthogonal projection of the triangle ABD gives the representation of an intuitionistic fuzzy set on the plane. (The orthogonal projection transfers $x' \in ABD$ into $x'' \in ABC$.) The interior of the triangle $ABC = ABD'$ is the area where $\pi > 0$. Segment AB represents a fuzzy set described by two parameters: μ and ν . The orthogonal projection of the segment AB on the axis μ (the segment $[0; 1]$ is only considered) gives the fuzzy set represented by one parameter μ only. (The orthogonal projection transfers $x'' \in ABC$ into $x''' \in CA$.) As it was shown in Szmmidt and Kacprzyk (2000), distances between intuitionistic fuzzy sets should be calculated taking into account three parameters describing an intuitionistic fuzzy set.

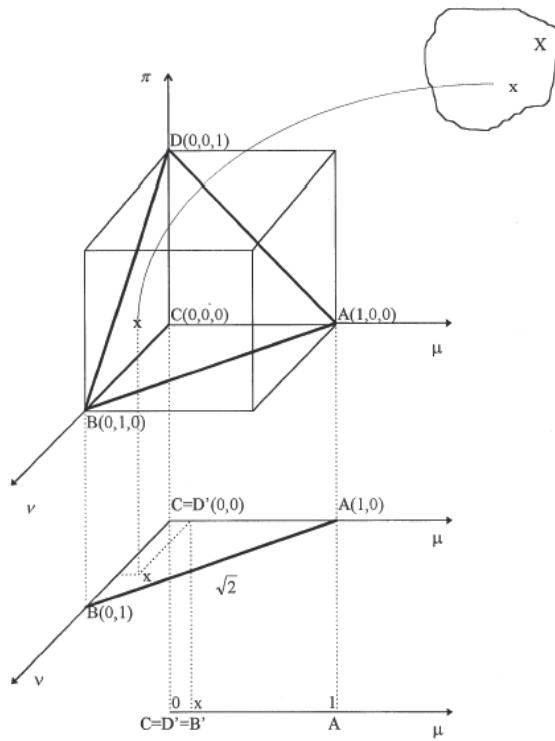


Fig. 4. A geometrical interpretation of an intuitionistic fuzzy set (Data Source: Szmida & Kacprzyk, 2000)

Second, in order to transfer the subjective valuation into real valuation, our proposed entropy-based defuzzification method is based on the Luca-Termini Axioms and Lee et al. (2001), and developed defined as follows:

- (1) Let $X = \{r_1, r_2, \dots, r_n\}$ be a universal set with elements r_i distributed in a pattern space, where $i = 1, 2, 3, \dots, n$.
- (2) Let \tilde{A} be a fuzzy set defined on an interval of pattern space which contains k elements ($k < n$). The mapped membership degree of the element r_i with the fuzzy set \tilde{A} is denoted by $\mu_{\tilde{A}}(r_i)$.
- (3) Let C_1, C_2, \dots, C_m represent m classes into which the n elements are divided.
- (4) Let $S_{C_j}(r_n)$ denote a set of elements of class j on the universal set X . It is a subset of the universal set X .
- (5) The match degree D_j with the fuzzy set \tilde{A} for the elements of class j in an interval, where $j = 1, 2, \dots, m$, is defined as

$$D_j = \frac{\sum_{r \in S_{C_j}(r_n)} \mu_{\tilde{A}}(r)}{\sum_{r \in X} \mu_{\tilde{A}}(r)} \quad (20)$$

(6) The fuzzy entropy $FE_{C_j}(\tilde{A})$ of the elements of class j in an interval is defined as

$$FE_{C_j}(\tilde{A}) = -D_j \log_2 D_j \quad (21)$$

(7) The fuzzy entropy $FE(\tilde{A})$ on the universal set X for the elements within an interval is defined as

$$FE(\tilde{A}) = \sum_{j=1}^m FE_{C_j}(\tilde{A}) \quad (22)$$

In Equation (21), the fuzzy entropy $FE_{C_j}(\tilde{A})$ is a non-probabilistic entropy and the match degree D_j in fuzzy entropy is measured via the membership values of occurring elements.

First of all, we will start our investigations with fuzzy states and consider the prior moments $\mu_\beta, CV_\beta, \mu_{\lambda_0}, CV_{\lambda_0}$ where be recorded with the interval $[i_1, i_2]$ of successive failure data ξ_r ($r = \mu_\beta, CV_\beta, \mu_{\lambda_0}$, or CV_{λ_0}) (i.e., about 0.3 within the interval $[i_1, i_2]$ for prior moment, respectively), the pattern space shown in the top of the Figure 5 (Hoffman et al., 1996). Subsequently, let us consider the importance weights of the prior parameters are assessed in linguistic terms represented by fuzzy numbers, such as “L” (Low), “M” (Medium) ‘“H” (High) where provided by experts’ linguistic form, and the membership functions of the three linguistic weight terms with triangular membership function are shown the bottom of the Figure 5. In addition, the value of a membership function can be viewed as the degree to which a pattern belongs to a specified pattern space. Fuzzy entropy of the observed interval $[i_1, i_2]$ has shows in the bottom of the Figure 5.

In order to show the advanced method, let us consider about the both parts of (a) and (b) with the probability formula in the Figure 5, the probability of “star” is: $p(star) = 4/5 = 0.8$, the probability of “circle” is: $1 - p(star) = 1/5 = 0.2$. In such situations, decision making depends on numerous factors which limit human ability and increase difficulties to deal with. Since, the result of Shannon’s entropy is:

$$\begin{aligned} H(X) &= -\sum_{j=0}^{N-1} p(x_j) \log_2 p(x_j) \\ &= -p(\star) \log_2 p(\star) - p(\circlearrowleft) \log_2 p(\circlearrowleft) \\ &= -0.8 \cdot \log_2 0.8 - 0.2 \cdot \log_2 0.2 \\ &= -0.25756814 - 0.464431896 \\ &= 0.722000036 \end{aligned}$$

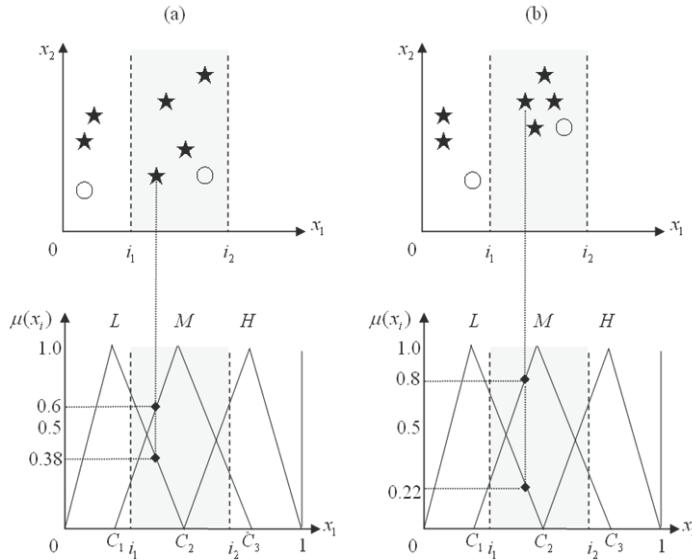


Fig. 5. Two cases of pattern distribution with corresponding membership functions (The symbol of \star and \circ stand for the degree of membership functions; C_1, C_2, C_3 denote the center of three triangular fuzzy sets, respectively)

As mentioned above, the match degree D_j in fuzzy entropy is based on mapped membership values of elements. Assume we begin by assigning three triangular membership functions with overlapped regions in the pattern space of $[0, 1]$, as shown in Figure 5. The value of a membership function can be viewed as the degree to which a pattern belongs to a specified pattern space. The fuzzy entropy of the interval $[i_1, i_2]$ for the degree of membership functions as show follows:

- (1). From the corresponding membership function \tilde{L} , the total membership degree of “ \star ” is $0.38 + 0.22 + 0 + 0 = 0.7$. Total membership degree of “ \circ ” is 0.0.

$$\text{The match degree of “}\star\text{” is } D_1 = \frac{0.7}{0.7+0} = 1.0$$

$$\text{The match degree of “}\circ\text{” is } D_2 = \frac{0}{0+0} = 0.0$$

The fuzzy entropy of $FE_{C_1}(\tilde{L})$

$$FE_{C_1}(\tilde{L}) = -1.0 \times \log_2(1.0) = 0$$

$$FE_{C_2}(\tilde{L}) = -0.0 \times \log_2(0.0) = 0$$

Hence, the fuzzy entropy of $FE_{C_1}(\tilde{L})$ for the patterns of the interval $[i_1, i_2]$ in the feature dimension x_1 is $FE(\tilde{L}) = FE_{C_1}(\tilde{L}) + FE_{C_2}(\tilde{L}) = 0$.

- (2). From the corresponding membership function \tilde{M} , the total membership degree of “★” is $0.6 + 0.88 + 0.85 + 0.57 = 2.3$. Total membership degree of “○” is 0.56.

The match degree of “★” is $D_1 = \frac{2.3}{2.3 + 0.56} = 0.80$

The match degree of “○” is $D_2 = \frac{0.56}{2.3 + 0.56} = 0.20$

The fuzzy entropy of $FE_{C_j}(\tilde{M})$

$$FE_{C_1}(\tilde{M}) = -0.80 \times \log_2(0.80) = 0.258$$

$$FE_{C_2}(\tilde{M}) = -0.20 \times \log_2(0.20) = 0.464$$

Hence, the fuzzy entropy of $FE_{C_j}(\tilde{M})$ for the patterns of the interval $[i_1, i_2]$ in the feature dimension x_1 is $FE(\tilde{M}) = FE_{C_1}(\tilde{M}) + FE_{C_2}(\tilde{M}) = 0.722$.

- (3). From the corresponding membership function \tilde{H} , the total membership degree of “★” is $0 + 0 + 0.1 + 0.4 = 0.5$. Total membership degree of “○” is 0.4.

The match degree of “★” is $D_1 = \frac{0.5}{0.5 + 0.4} = 0.56$

The match degree of “○” is $D_2 = \frac{0.4}{0.5 + 0.4} = 0.44$

The fuzzy entropy of $FE_{C_j}(\tilde{H})$

$$FE_{C_1}(\tilde{H}) = -0.56 \times \log_2(0.56) = 0.468$$

$$FE_{C_2}(\tilde{H}) = -0.44 \times \log_2(0.44) = 0.521$$

Hence, the fuzzy entropy of $FE_{C_j}(\tilde{H})$ for the patterns of the interval $[i_1, i_2]$ in the feature dimension x_1 is $FE(\tilde{H}) = FE_{C_1}(\tilde{H}) + FE_{C_2}(\tilde{H}) = 0.989$

Further, we can obtain the whole fuzzy entropy via summation of all corresponding fuzzy entropies as follows:

$$FE = FE_{C_j}(\tilde{L}) + FE(\tilde{M}) + FE(\tilde{H}) = 0 + 0.722 + 0.989 = 1.711$$

From the above illustration, the entropy-based defuzzification method will be able to discriminate the actual distribution of patterns better. By employing membership functions for measure match degrees, the value of entropy not only considers the number of patterns but also takes the actual distribution of patterns into account.

5. Application

In this section we will show the discrimination problems with both cases of fuzzy states and exact information and exact states and fuzzy information, which are studied from the viewpoint of fuzzy arithmetic measures.

5.1 Fuzzy States and Exact Information

Suppose that the states spaces of the four fuzzy moments and the linguistic importance weight of each value assigned by experts are assessed and shown in Table 1.

The Prior Moments of Interval Observations	Linguistic Weight
$\mu_{\lambda_0} = \xi_{\mu_{\lambda_0}} [0.4, 0.7]$	L[0.1,0.3,0.5]
$CV_{\lambda_0} = \xi_{cv_{\lambda_0}} [0.3, 0.9]$	M[0.3,0.5,0.7]
$\mu_{\beta} = \xi_{\mu_{\beta}} [0.6, 0.8]$	M[0.3,0.5,0.7]
$CV_{\beta} = \xi_{cv_{\beta}} [0.4, 0.6]$	H[0.5,0.7,0.9]

Table 1. The fuzzy prior moments provided by experts

Furthermore, by applying the fuzzy entropy method, the fuzzy number can be defuzzified into the crisp value. For example the defuzzified value of $\xi_{cv_{\lambda_0}} [0.3, 0.9]$ is attainable as shown in Figure 6.

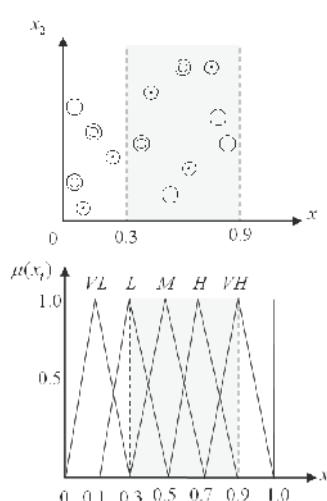


Fig. 6. The patterns of the interval $[0.3, 0.9]$ in the feature dimension $\xi_{cv_{\lambda_0}}$

- (1). From the corresponding membership function \tilde{VL} ,
The total membership degree of “○” is $0 + 0 + 0 = 0.0$
Total membership degree of “◎” is $0 + 0 + 0 = 0.0$
Total membership degree of “○” is $0 + 0 + 0 = 0.0$

The match degree of “○” is $D_1 = \frac{0}{0+0+0} = 0.0$

The match degree of “◎” is $D_2 = \frac{0}{0+0+0} = 0.0$

The match degree of “○” is $D_2 = \frac{0}{0+0+0} = 0.0$

The fuzzy entropy of $FE_{C_j}(\tilde{VL})$

$$FE_{C_1}(\tilde{VL}) = -0.0 \times \log_2(0.0) = 0$$

$$FE_{C_2}(\tilde{VL}) = -0.0 \times \log_2(0.0) = 0$$

$$FE_{C_3}(\tilde{VL}) = -0.0 \times \log_2(0.0) = 0$$

Hence, the fuzzy entropy of $FE_{C_j}(\tilde{VL})$ for the patterns of the interval [0.3, 0.9] in the feature dimension $\xi_{cv_{\lambda_0}}$ is $FE(\tilde{VL}) = FE_{C_1}(\tilde{VL}) + FE_{C_2}(\tilde{VL}) + FE_{C_3}(\tilde{VL}) = 0 + 0 + 0 = 0.0$

(2). From the corresponding membership function \tilde{L} ,

The total membership degree of “○” is $0 + 0 + 0 = 0.0$

Total membership degree of “◎” is $0.6 + 0 = 0.6$

Total membership degree of “○” is $0.5 + 0 + 0 = 0.5$

The match degree of “○” is $D_1 = \frac{0}{0+0.6+0.5} = 0.0$

The match degree of “◎” is $D_2 = \frac{0.6}{0+0.6+0.5} = 0.546$

The match degree of “○” is $D_3 = \frac{0.5}{0+0.6+0.5} = 0.454$

The fuzzy entropy of $FE_{C_j}(\tilde{L})$

$$FE_{C_1}(\tilde{L}) = -0.0 \times \log_2(0.0) = 0$$

$$FE_{C_2}(\tilde{L}) = -0.546 \times \log_2(0.546) = 0.4767$$

$$FE_{C_3}(\tilde{L}) = -0.454 \times \log_2(0.454) = 0.5173$$

Hence, the fuzzy entropy of $FE_{C_j}(\tilde{L})$ for the patterns of the interval [0.3, 0.9] in the feature dimension $\xi_{cv_{\lambda_0}}$ is $FE(\tilde{L}) = FE_{C_1}(\tilde{L}) + FE_{C_2}(\tilde{L}) + FE_{C_3}(\tilde{L}) = 0 + 0.4767 + 0.5173 = 0.994$

(3). From the corresponding membership function \tilde{M} ,

The total membership degree of “○” is $0.75 + 0 + 0 = 0.75$

Total membership degree of “◎” is $0.35 + 0.50 = 0.85$

Total membership degree of “ \odot ” is $0.55 + 0.4 + 0 = 0.95$

$$\text{The match degree of } \text{“}\odot\text{”} \text{ is } D_1 = \frac{0.75}{0.75 + 0.85 + 0.95} = 0.294$$

$$\text{The match degree of } \text{“}\odot\odot\text{”} \text{ is } D_2 = \frac{0.85}{0.75 + 0.85 + 0.95} = 0.333$$

$$\text{The match degree of } \text{“}\odot\odot\odot\text{”} \text{ is } D_3 = \frac{0.5}{0.75 + 0.85 + 0.95} = 0.373$$

The fuzzy entropy of $FE_{C_j}(\tilde{M})$

$$FE_{C_1}(\tilde{M}) = -0.294 \times \log_2(0.294) = 0.5192$$

$$FE_{C_2}(\tilde{M}) = -0.333 \times \log_2(0.333) = 0.5283$$

$$FE_{C_3}(\tilde{M}) = -0.373 \times \log_2(0.373) = 0.5307$$

Hence, the fuzzy entropy of $FE_{C_j}(\tilde{M})$ for the patterns of the interval $[0.3, 0.9]$ in the feature dimension $\xi_{cv_{j_0}}$ is

$$FE(\tilde{M}) = FE_{C_1}(\tilde{M}) + FE_{C_2}(\tilde{M}) + FE_{C_3}(\tilde{M}) = 0.5192 + 0.5283 + 0.5307 = 1.5782$$

(4). From the corresponding membership function \tilde{H} ,

The total membership degree of “ \odot ” is $0.15 + 0.5 + 0.2 = 0.85$

Total membership degree of “ $\odot\odot$ ” is $0.0 + 0.40 = 0.40$

Total membership degree of “ $\odot\odot\odot$ ” is $0.0 + 0.8 + 0.6 = 1.4$

$$\text{The match degree of } \text{“}\odot\text{”} \text{ is } D_1 = \frac{0.85}{0.85 + 0.40 + 1.4} = 0.32$$

$$\text{The match degree of } \text{“}\odot\odot\text{”} \text{ is } D_2 = \frac{0.40}{0.85 + 0.40 + 1.4} = 0.15$$

$$\text{The match degree of } \text{“}\odot\odot\odot\text{”} \text{ is } D_3 = \frac{0.14}{0.85 + 0.40 + 1.4} = 0.53$$

The fuzzy entropy of $FE_{C_j}(\tilde{H})$

$$FE_{C_1}(\tilde{H}) = -0.32 \times \log_2(0.32) = 0.5260$$

$$FE_{C_2}(\tilde{H}) = -0.15 \times \log_2(0.15) = 0.4105$$

$$FE_{C_3}(\tilde{H}) = -0.53 \times \log_2(0.53) = 0.4854$$

Hence, the fuzzy entropy of $FE_{C_j}(\tilde{H})$ for the patterns of the interval $[0.3, 0.9]$ in the feature dimension $\xi_{cv_{j_0}}$ is

$$FE(\tilde{H}) = FE_{C_1}(\tilde{H}) + FE_{C_2}(\tilde{H}) + FE_{C_3}(\tilde{H}) = 0.5260 + 0.4105 + 0.4854 = 1.4219$$

(5). From the corresponding membership function VH ,

The total membership degree of “○” is $0.0 + 0.5 + 0.8 = 1.3$

Total membership degree of “◎” is $0.0 + 0.0 = 0.0$

Total membership degree of “○” is $0.0 + 0.0 + 0.45 = 0.45$

The match degree of “○” is $D_1 = \frac{1.3}{1.3 + 0.0 + 0.45} = 0.74$

The match degree of “◎” is $D_2 = \frac{0.0}{1.3 + 0.0 + 0.45} = 0.0$

The match degree of “○” is $D_3 = \frac{0.45}{1.3 + 0.0 + 0.45} = 0.26$

The fuzzy entropy of $FE_{C_1}(\tilde{VH})$

$$FE_{C_1}(\tilde{VH}) = -0.74 \times \log_2(0.74) = 0.3214$$

$$FE_{C_2}(\tilde{VH}) = -0.0 \times \log_2(0.0) = 0.0$$

$$FE_{C_3}(\tilde{VH}) = -0.26 \times \log_2(0.26) = 0.5053$$

Hence, the fuzzy entropy of $FE_{C_j}(\tilde{VH})$ for the patterns of the interval [0.3, 0.9] in the feature dimension $\xi_{cv_{\lambda_0}}$ is

$$FE(\tilde{VH}) = FE_{C_1}(\tilde{VH}) + FE_{C_2}(\tilde{VH}) + FE_{C_3}(\tilde{VH}) = 0.3214 + 0.0 + 0.5053 = 0.8267$$

Finally, we can obtain the whole fuzzy entropy via summation of all corresponding fuzzy entropies as follows:

$$\begin{aligned} FE &= FE(\tilde{VL}) + FE(\tilde{L}) + FE(\tilde{M}) + FE(\tilde{H}) + FE(\tilde{VH}) \\ &= 0.0 + 0.994 + 1.5782 + 1.4219 + 0.8267 \\ &= 4.8208 \end{aligned}$$

Finally, Equations (8) and (9) can be used to study the prior and posterior decision for the decision maker when dealing with the decision problem for deteriorating repairable systems.

5.2 Fuzzy States and Fuzzy Information

When the states and the addition information are both fuzzy, besides the work for the fuzzy prior as described in the previous case, we have to also deal with the problem of defuzzifying the failure data as shown in Table 2.

Failure Data	Fuzzy interval (Hour / 24)	Linguistic Weight
1989.07.06 (ADD_1)	[0.33, 0.50]	L[0.1, 0.3, 0.5]
1989.10.23 (ADD_2)	[0.42, 0.58]	M[0.3, 0.5, 0.7]
1990.01.12 (ADD_3)	[0.58, 0.66]	M[0.3, 0.5, 0.7]
1990.09.08 (ADD_4)	[0.42, 0.50]	H[0.5, 0.7, 0.9]
1990.11.14 (ADD_5)	[0.54, 0.63]	H[0.5, 0.7, 0.9]

Table 2. The fuzziness of the added failure data

The fuzzy numbers of failure data can also be defuzzified into crisp values and form the likelihood function. Equations (8) and (11) can be used to study the prior and posterior decision for the decision maker when dealing with the decision problem for deteriorating repairable systems. The decision problems for both the previous two cases can be assessed; however, the computing problem in Huang (2001) for the posterior mean of the decision variable is also encountered. If the expected number of failures from the decision time until the system is discarded is used as the decision variable, the numerical integration is still needed for evaluating the values and therefore making the decision.

6. Conclusion

In this chapter, we have presented a method to solve the decision problem of deteriorating repairable systems and we also present an approach to illustrate the fuzzy entropy-based arithmetic approach for modeling experts' epistemic uncertainty in deteriorating repairable systems. The decision process is useful in selecting the best alternative when the deteriorating repairable system associated with alternatives are known in terms of linguistic variables (Zadeh, 1975), in particular, when these linguistic variables can be modeled by fuzzy numbers. In real world situations, the deteriorating phenomena are usually expressed as some degrees of severity. In such case, the proposed decision process can provide more realistic solutions. In this chapter, we have assumed that the importance weights of different criteria are assessed in linguistic terms represented by triangular fuzzy numbers. However, there are still several limitations and further study may undergo by considering other kinds of fuzzy membership function, since it still leaves lots of space for extension.

7. References

- Asai, K.; Tanaka, H.; & Okuda, T. (1975). Decision Making And Its Goal In A Fuzzy Environment, In: Zadeh et al. (Eds) *Fuzzy Sets and Their Applications to Cognitive and Decision Processes*, New York, London.
- Ascher, H.; & Feingold, H. (1984). *Repairable Systems Reliability*, Marcel Dekker, New York.
- Barlow, R; & Proschan, F. (1965). *Mathematical Theory of Reliability*, Wiley, New York.
- Belahut, R. (1987). *Principles and Practice of Information Theory*, Reading, MA: Addison-Wesley.
- Bellman, R.; & Zadeh, L. (1970). Decision Making In A Fuzzy Environment, *Management Science*, Vol. 17, pp. B-141-164.
- Chang, C.; & Cheng, C. (2007). A Bayesian Decision Process with Fuzzy Interpretability for Aging Chronic Disease, *International Journal of Technology Management*, Vol.40, No.1/2/3, pp.176-191.
- Chang, C. (2008). Entropy-Based Defuzzification Method with Experts' Epistemic Uncertainty for Deteriorating Repairable Systems, *The Seventh International Conference on Machine Learning and Applications (ICMLA'08)*, San Diego, California, USA, Dec, 2008.
- Ching, J. et al., (1995). Class-dependent discretization for inductive learning form continuous and mixed-mode data, *IEEE Trans. Pattern Anal. MachineIntell.*, Vol. 17, pp. 641-651.
- Cover, T.; & Thomas, J. (1992). *Elements of Information Theory*. NewYork: Wiley.

- Crow, L. (1974). Reliability Analysis For Complex Repairable Systems in: Proschan, F. and Serfling, R. J. (Eds), *Reliability and biometry*, Philadelphia, *Society for Industrial and Applied Mathematics*, pp. 379-410.
- De Luca A.; & Termini, S. (1972). A definition of a non-probabilistic entropy in the setting of fuzzy sets theory, *Information and Control*, Vol. 20, pp. 301-312.
- Domper, K. (1982). *The Theory of Fuzzy Decisions*, Approximate reasoning in decision analysis M.M. Gupta and E. Sanchez (eds.), North-Holland Publishing, pp. 365-379.
- Ferson, S.; Joslyn, C.; Helton, J.; Oberkampf, W.; & Sentz, K. (2004). Summary from the epistemic uncertainty workshop: consensus amid diversity, *Reliab Engng Syst Saf*. Vol. 85, pp. 355-369.
- Fetz, Th.; & Oberguggenberger, M. (2004). Propagation of uncertainty through multivariate functions in the framework of sets of probability measures, *Reliab Engng Syst Saf*. Vol. 85, pp. 73-87.
- Freeling, A. N. S. (1984). *Possibilities Versus Fuzzy Probabilities - Two Alternative Decision Aids* In: Zimmermann et al (Eds.) *Fuzzy Sets and Decision Analysis*, Amsterdam, New York, Oxford, pp. 67-82.
- Fruhwirth-Schnatter, S. (1993). On Fuzzy Bayesian Inference, *Fuzzy Sets and Systems*, Vol. 60, pp. 41-58.
- Härtler, G. (1989). The Non-Homogeneous Poisson Process - A Model For The Reliability Of Complex Repairable Systems, *Microelectronics and Reliability*, Vol. 29, No. 3, pp. 381-386.
- Herrera, F.; & Herrera, V. (2000). Linguistic Decision Analysis: Steps For Solving Decision Problems Under Linguistic Information, *Fuzzy Sets And Systems*, Vol. 115, No. 3, pp. 62-82.
- Hisdal, E. (1984). Decision Based On Statements In Natural Language, *TIMS/Studies in the Management Sciences*, Vol.20, pp. 357-381
- Hoffman, M.; Manevitz, L.; & Wong, E. (1996). Fuzzy independence and extended condition probability, *Information Sciences*, Vol. 90, pp. 137-156.
- Huang, Y. (2001). A Decision Model for Deteriorating Repairable Systems, *IIE Transactions*, Vol. 33, No. 6, pp. 479-485.
- Huang, Y.; & Bier, V. (1998). A Natural Conjugate Prior for The Non-Homogeneous Poisson Process With a Power Law Intensity Functions, *Communications in Statistics-Simulations and Computation*. Vol.27, No.2, pp. 525-551.
- Huang, Y.; & Chang, C. (2004). A study of defuzzification with experts' knowledge for deteriorating repairable systems, *European Journal of Operational Research*, Vol.157, No.3, pp. 658-670.
- Kandel, A. (1986). *Fuzzy Mathematical Techniques with Applications*, Addison-Wesley, New York.
- Kaufmann A. (1975). *Introduction to the Theory of Fuzzy Subsets*, Vol. 1: Fundamental Theoretical Elements, Academic Press, New York,
- Kim, J.; Cho C.; & Hyung L. (1998). A note on the set-theoretical defuzzification, *Fuzzy Sets and System*, Vol.98, pp. 337-341.
- Kosko , B. (1986). Fuzzy entropy and conditioning, *Inform. Sci.*, Vol. 40, pp. 165-174, Dec.
- Kosko , B. (1990). Fuzziness vs. probability, *International Journal of General Systems*, Vol. 17, no.2/3, pp. 211-240.
- Kosko , B. (1997). *Fuzzy Engineering*, Prentice-Hall, Englewood Cliffs, N.J.

- Lee, H.; Chen, C.; Chen, J.; & Jou, Y. (2001). An efficient fuzzy classifier with feature selection based on fuzzy entropy, *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, Vol. 31, No. 3, June.
- Ma, M.; Kandel, A.; & Friedman, M. (2002). Correction to A new approach for defuzzification, *Fuzzy Sets and System*, Vol.128, pp. 133-134.
- Oberkampf, W.; Helton, J.; Joslyn, C.; Wojtkiewicz, S.; & Ferson, S. (2004) Challenge Problems: uncertainty in system response given uncertain parameters, *Reliab Engng Syst Saf*. Vol. 85, pp. 11-19.
- Pal, S.; & Chakraborty, B. (1986). Fuzzy set theoretic measure for automatic feature evaluation, *IEEE Trans. Syst., Man, Cybern.*, Vol. SMC-16, pp. 754-760.
- Renyi, (1961). On the measure of entropy and information, in Proc. *Fourth Berkeley Symp. Math. Statistics Probability*, Vol. 1, Berkeley, CA, pp. 541-561.
- Roubens, M. (1997). Fuzzy And Decision Analysis, *Fuzzy Sets and System*, Vol. 90, pp. 199-206.
- Roychowdhury, S.; & Pedrycz, W. (2001). A Survey of defuzzification strategies, *International Journal of intelligent systems*, Vol.16, pp. 679-695.
- Shannon, C. (1948), A mathematical theory of communication, *Bell Syst.Tech. J.*, Vol. 27, pp. 379-423.
- Stephen, R.; & Donnell, M. (1979). Fuzzy Decision Analysis, *IEEE Transactions on systems, man, and cybernetics*, Vol. SMC-9, No.1, January.
- Szmidt E.; & Kacprzyk, J. (2000) On distances between intuitionistic fuzzy sets, *Fuzzy Sets and Systems*, Vol. 114, pp. 505-518.
- Tong, R.; & Bonissone, P. (1984) Linguistic Solutions To Fuzzy Decision Problem, *TIMS/studies in the management Sciences*, Vol. 20, pp. 323-334.
- Yager, R. (1979). A note on probabilities of fuzzy events, *Information Sciences*, Vol. 18, pp. 113-129.
- Zadeh L. (1965). Fuzzy sets, *Information and Control*, Vol. 8, pp. 338-353.
- Zadeh L. (1983). The role of fuzzy logic in the management of uncertainty in expert systems, *Fuzzy Sets and Systems*, Vol. 11, pp. 199-227.
- Zadeh, L. (1975). The Concept of a Linguistic Variable and Its Application to Approximate Reasoning - I, *Information Science*, Vol. 8, pp. 199-249.

Alarming Large Scale of Flight Delays: an Application of Machine Learning

Zonglei Lu

*College of Computer Science and Technology,
Civil Aviation University of China
People's Republic of China*

1. Introduction

Flight delays occur at almost all the airports every year. At major hub airports, flight delays occur more frequently and cause more serious problems. The FAA, i.e. Federal Aviation Administration, divides the flight delays into two types. One is the so-called technique delay, which is mainly caused by the flight waiting for the resources of air flow control or the limit of flow management, and the other is the so-called effective arrival delay. According to some papers, there are five kinds of flight delays, i.e. the delay of flow operating, the delay of aircraft ground operating, the aircraft technique delay, the delay of flow control and the airport delay. There are many reasons of flight delays. Macroscopically, the main reason of flight delays is that the capacity of airspace and airports cannot meet the requirement of the increasing air traffic. Bad weather, mechanical problems of aircraft, the operation issues of airlines and unexpected problems caused by passengers may be the main reasons in microscopic scales. In recent years, most of the researchers agree with that the main reason of flight delays is the limit of the runway capacity. Therefore flight delays can be reduced by increasing the number of runways, taxiways and the air flow control devices and improving flight management, such as the process of air flow control, the sequence of the arrival and departure flights.

Because of the serious consequences of flight delays, there has been a tremendous amount of researches on how to deal with the various problems caused by flight delays. However, very little research focuses on how to forecast flight delays. Since there are so many factors of the flight delays, it is very difficult to calculate the delays by the deduction. Any mistakes of the factors may lead the result invalid; in contrast, the prediction by statistical law may be more effective. From the view of statistics, the flights could be perceived as repeated experiments, although there may be a few differences between the environments of each experiment. Therefore, the delay time of the flight should satisfy some special probability distribution. The difference of the experiment environments is just the reason why the time taken by the flights is variable. The uncertainty of experiment environments would add the prediction difficulty of delay time. Conversely, assume that all the factors of two flights are same, i.e. the two experiments have been taken in the same environment. Then they must have same delay time, i.e. the results of the two experiments are same. The probability distribution

could show the a priori probability of the delay time. As a prediction system, the posterior probability, i.e. the probability of the delay time in some special condition, may be more important than the a priori probability. Machine learning is usually used to calculate the posterior probability from the data.

In this chapter, an example will be shown to introduce how to use machine learning methods to alarm large scale of flight delays, which is a very complex problem. This is a real application in a hub airport of China. We will use the symbols "*Airport A*" to denote this airport with omitting its real name. The concept of "flight delay" may be changed under different requirement. Thus, the background of the application will be introduced firstly in this chapter and the relative concepts, especially the application requirement, will be defined clearly. Besides, the data gotten from the practice, which is an important factor to use machine learning methods, will be depicted in this chapter. According to the data and the requirement, the process of learning could be discussed. To solve the problem of alarming flight delays, the delay levels, which are used to describe the serious degree of delays, are required. Therefore, there will be a section to show how to use the unsupervised learning method to calculate the delay levels. Depending on the defined levels, the supervised method could be used to predict the coming delays. Some classical methods, both supervised and unsupervised, will be tried in this chapter, although there will be only one method in the final application. After introducing these methods, how to choose the most suitable one will be discussed. At last, a conclusion of the whole chapter will be shown .

2. Application Requirements and Data Preparing

Flight is a civil aircraft which take a mission from a certain departure airport to the certain arrival airport in the certain time according to the flight schedule. There are two kinds of time to each flight in the schedule, i.e. the plan departure time and the plan arrival time. The flight departure or arrival at the time later than the plan departure or arrival time is called a departure delay flight or arrival delay flight respectively. Since the flight is influenced by multiple factors, the concepts of departure delay flight and arrival delay flight may be too strict. The constraint is usually relaxed in practical applications. In the *Airport A*, the flight which departs or arrives at the time later than the schedule time in 30 minutes is treated as a normal flight but not a delayed one. The concept "large scale of flight delays" means the number (or the ratio) of the delayed flights is more than a given threshold. To alarm large scale of flight delays is just to give a prediction, the main foundation of which is the recent status of the airport, before the delays.

The main difference between engineering application and laboratory experiment is the data, which are always ready in the laboratory. Collecting enough data and converting them into right form is very important to the engineering. The statistics theory, which aims at the asymptotic theory when sample size is tend to infinity, is the basic theory of machine learning. However, the size of sample usually is finite, so that some learning methods perform well only in theory. To guarantee the application of the machine learning method effectively, we need discuss the form of the data which could be collected from the airport. The form of data restricts the method which could be used in the application. Conversely, the method itself could tell us what kind of data are required.

Intuitively, finding the association event of flight delays may be a good way to do prediction. The alarm could be given when the association events happened. The recent researches show that there are five classes of factor which may lead flight delays, including: weather, the management of the airline company, the control of airports, the air traffic control and the factor of the passengers. However, it is not so easy to collecting the data of all these factors. For example, the factor of the passengers is really uncertainty. It is possible that some passenger, who had already passed the security checking, decided not to board in without any announcement. In this case, the staffs must find the passenger and confirm that he (or she) would not board in. The flight could not take off before the confirmation. Since this is just a stochastic event which may cause the flight delays, it is very difficult to monitor this factor. Therefore, the prediction from the factors may be invalidity.

Actually, the so-called "association event" may not be the direct reason of delay, but a concept of statistics. An event is called an association event of the flight delays means that the flight is very likely to delay when this event occurs. In the language of the probability theory, the conditional probability of the association event to the flight delays is large enough. Thus, we need to find the association events of the flight delays firstly, and then to find the pattern of the association events and the flight delays. Obviously, only the events in the records of *Airport A* could be used to predict flight delays. Therefore, we will show the Attributes of the records of *Airport A* in the following (See Table 1) in order to discuss the association events.

Attribute	Explanation	Type
FID	The ID of the flight	Nominal
AID	The ID of the aircraft	Nominal
D/A	Departure flight or arrival flight where the symbol "D" and "A" stands for departure flight and the arrival flight, respectively	{"D", "A"}
D/I	Domestic flight or International flight where the symbol "D" and "A" stands for Domestic flight or International flight, respectively	{"D", "I"}
AT	The type of the aircraft	Nominal
CT	The code of the task such as normal flight, charter flight and so on	Nominal
TB	Terminal building which will serve the flight	Numeric
AC	The air company that the flight belongs to	Nominal
SD	The time when the flight departs in the schedule	Numeric
PD	The predicting time when the flight will depart	Numeric
RD	The time when the flight departs	Numeric
TA	The target airport of the flight	Nominal
SA	The time when the flight arrives in the schedule	Numeric
PA	The predicting time when the flight will arrive	Numeric
RA	The time when the flight arrives	Numeric
BS	The boarding status of the flight	Nominal
BG	Which gate will be used to boarding to the flight	Nominal

Table 1. the Attributes of the records of *Airport A*

Table 1 shows all attributes recorded by *Airport A*. In other words, all predictions must be done only on these attributes. However, there is very little correlation between the above

attributes and the delays intuitively. Actually, the calculation of the data shows the same conclusion, i.e. none of the above attributes is the association event of flight delays. This may be a big problem. Without the association events, the prediction will miss the basis.

The above case is very common in many applications. Data are not so perfect where we cannot find what we expected. In this case, we must rebuild the data set with the association attributes. Rebuilding data set does not mean recollecting data. What we need to do is just to get the statistical information, which is associated to the flight delay intuitively.

One of the common methods is to build time associated attributes, i.e. the association of recent status and the past status. In most of systems, the recent status is an important factor of the future status. Since the computer could only process the discrete data, we assume that the monitoring system is discrete. For example, we can assume that the status is monitored every five minutes. Because the flight delay is mainly caused by the capacity of airspace and airports not large enough, only a few flights could be executed in five minutes. Thus, the recent delay flights in the waiting sequence may be also in the waiting sequence in the next five minutes. The number of delay flight in the next statistical period includes two parts: one is the recent delay flights and the other is the new arrival or departure delay flights. In other words, the recent status is associated with the future status. Furthermore, we can use the association to do prediction in order to alarm the flight delays.

The severity of flight delays finds expression not only in the amount of delayed flights, but also in the delay hours and passenger numbers involved, etc. Since we are going to build a model to predict the future delays, we must try to find all association events as possible as we can. According to the recent researches, there five aspects of flight delays including the number of delayed flights, total delay hours, the number of passengers involved, the number of air companies involved, and the number of airports which the delays have spread to directly. We can get all the value of the above attributes by statistical method except the number of passengers involved. The data provided by *Airport A* does not contain the information of passengers. But we can get an estimate value by the number of seats of each type of aircraft and the average attendance rate, which could be found in the civil aviation handbooks. Table 2 shows some samples of the association attributes in the following.

the number of delayed flights	total delay hours	the number of airports which the delays have spread to directly	the number of air companies involved	the number of passengers involved
118	142	61	24	14997
47	79	32	14	5904
85	301	49	22	10318
110	144	56	26	14160
156	147	65	32	20819
169	243	68	34	22411
172	219	62	27	22951
150	137	59	30	20074
102	115	55	21	13588
173	206	64	23	22140
...

Table 2. Samples of the Association Attributes to Flight Delays

The data shown in Table 2 are statistical values of each day at a given time. Considering there are big differences between the ranges of each attribute shown in Table 2. The attributes with the big range may influence the prediction more than the one with small range, which is unfair intuitively. We will normalize the data in order to eliminate effects.

3. Grading Flight Delays with Unsupervised Learning Methods

According to the data shown in Table 2, our prediction should be aim at the delay levels of *Airport A* at a given time. The basis of the prediction is the value of the five attributes shown in Table 2. Thus, we must build a model to show the relationship of these five attributes and delay levels. However, there is no information about the delay levels. Therefore, we must calculate the delay levels at given time according to the value of the five attributes shown in Table 2. The prediction model could be built only after the levels have been ready.

From the viewpoint of data, there should be some similarity among the data in same level. Assume that there is only one attribute (for example, the number of delayed flights) in Table 2. Then the values of data in the same level must be very close to each other. Similar with this simple case, the data in the same level must be similar with each other in the case of more than one attributes. The similarity could be calculated by the sum of squares of the difference of each attribute, which denotes the Euclid's distance, between two data. Thus, the level could be gotten by group the flight data with the similarity, which is just a process of clustering. Clustering is perceived as an unsupervised process since there are no predefined classes and no examples that would show what kind of desirable relations should be valid among the data.

To simplify the problem, we will use k -means algorithm, which is very popular and easy to be implemented, to grade the data of flight delays. There are three steps of k -means algoithm. First of all, select k means of clusters randomly. Then put each data to the cluster whose mean is nearest to the data, i.e. the similarity measure value of the data and the cluster's mean is bigger than the value of any other cluster's mean and the data. Then the algorithm recalculates the mean of each cluster. At last the algorithm output the clusters if they are not changed, otherwise it will return to the second step. The parameter k denotes the number of clusters, which means the number of levels of flight delays in this application. The recent subjective standard of flight delays in *Airport A* divides the flight delays into four levels, which is shown in Table 3.

Levels	Definition
Blue Level	there should be blue alarm if more than 40% of the departure flights will be delayed for the bad weather or the control of route
Yellow Level	there should be yellow alarm if the weather is so bad that more than 60% of the departure flights in the coming two hours will be delayed or more than 10 flights will be delayed for more than 4 hours.
Orange Level	there should be orange alarm if more than 80% of the departure flights in the coming two hours will be delayed for the bad weather.
Red Level	there should be red alarm if all of the departure flights in the coming two hours will be delayed for the bad weather.

Table 3. Recent Standard of Flight Delays in *Airport A*

The main factor of the levels shown in Table 3 is the weather, but there are only about 4.63% of delayed flights caused by the bad weather according to (Ma & Cui, 2004). Therefore, it is difficult to check the validity of this standard. Besides, there is no attribute about the weather in the data set provided by *Airport A*. Thus, we cannot mark the levels to the data. Although this subjective grading may not fit the data very well, it still provides some information about the level. After all, these levels are constituted by the domain experts. There are four levels in Table 3 without the level of few flight delays. Thus, it may be five levels of the data, which is an important information of the k -means algorithm. According to Table 3, the parameter of the k -means algorithm may be set as $k=5$ in this application. Certainly, this is just an assumption which needs to be checked in the experiment. Since there is no standard answer about what the clustering process learns, a measure index of clustering is required in order to make the clustering result fit the application and find the optimum parameters. We first introduce four clustering validity indexes, including Dunn's index, Davies-Bouldin's index, CS index and Lu's index, and then try to find the optimum parameter k of k -means by these four indexes.

Dunn's index is defined as

$$D_{n_c} = \min_{i=1, \dots, n_c} \left\{ \min_{j=i+1, \dots, n_c} \left\{ \frac{d(C_i, C_j)}{\max_{k} \text{diam}(C_k)} \right\} \right\} \quad (1)$$

where $d(C_i, C_j)$ is the dissimilarity function between two clusters C_i and C_j defined as $d(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y)$ and $\text{diam}(C)$ is the diameter of a cluster, which may be considered as a measure of clusters' dispersion. The diameter of a cluster C can be defined as follows:

$$\text{diam}(C) = \max_{x, y \in C} d(x, y) \quad (2)$$

If the data set contains compact and well-separated clusters, the distance between the clusters is expected to be large and the diameter of the clusters is expected to be small. The Davies-Bouldin's index is defined as

$$DB_{n_c} = \frac{1}{n_i} \sum_{i=1}^{n_c} R_i \quad (3)$$

where $R_i = \max_{j=1, \dots, n_c, i \neq j} \frac{s_i + s_j}{d_{ij}}$ and s_i is a measure of dispersion of a cluster C_i . It is clear for the

above definition that DB_{n_c} is the average similarity between each cluster C_i , $i=1, \dots, n_c$ and its most similar one. It is desirable for the clusters to have the minimum possible similarity to each other; therefore we seek clusterings that minimize DB . The DB_{n_c} index exhibits no trends with respect to the number of clusters and thus we seek the minimum value of DB_{n_c} in its plot versus the number of clusters.

The CS index is defined as

$$CS(N_c) = \frac{\sum_{i=1}^{N_c} \left\{ \frac{1}{|C_i|} \sum_{x_j \in C_i} \max_{x_k \in C_i} \{d(x_j, x_k)\} \right\}}{\sum_{i=1}^{N_c} \left\{ \min_{j=1,2,\dots,N_c, j \neq i} \{d(v_i, v_j)\} \right\}} \quad (4)$$

where x_j and v_i is the j th data and the center of the i th cluster, respectively. The Lu's index is defined as

$$LI(N_c) = \frac{1}{2} \sum_{i=1}^{N_c} \sum_{x_j \in C_i} \left(\left| \{d_k \in C_j : d(d_j, d_k) > \lambda\} \right| + \left| \{d_k \notin C_j : d(d_j, d_k) \leq \lambda\} \right| \right) \quad (5)$$

where λ is a threshold of similarity. The Lu's index is not with respect to the concept of "cluster center" so that it could be used in the case of non-spherical clusters.

We will cluster the data by k -means algorithm with the parameter k varies from 3 to 10. The value of the above indexes on the clusters is shown in Table 4.

Number of Clusters	Dunn's Index	Davies-Bouldin's Index	CS Index	Lu's Index
3	0.0417	0.7622	1.3653	0.1884
4	0.0373	0.7944	1.3290	0.1719
5	0.0454	0.8357	1.2638	0.1704
6	0.0353	0.9170	1.4038	0.2006
7	0.0372	0.9182	1.2662	0.2083
8	0.0353	0.9435	1.3217	0.2318
9	0.0376	1.0129	1.3776	0.2514
10	0.0305	0.9918	1.3257	0.2261
Optimal Number	5	3	5	5

Table 4. The Values of Indexes on Each Clustering

Level	the number of delayed flights	total delay hours	the number of airports which the delays have spread to directly	the number of air companies involved	the number of passengers involved
very low delays	0.1113±0.0576	0.0367±0.023	0.2094±0.0967	0.2084±0.0961	0.1170±0.0605
low delays	0.2342±0.0508	0.0578±0.0185	0.3893±0.0574	0.4130±0.0898	0.2436±0.0531
moderate delays	0.3820±0.0585	0.0885±0.0217	0.5333±0.0522	0.5579±0.0938	0.3991±0.0575
significant delays	0.5615±0.0685	0.1539±0.0515	0.6784±0.0588	0.6778±0.0997	0.5774±0.0649
excessive delays	0.8233±0.0934	0.3867±0.174	0.8299±0.0757	0.7859±0.1246	0.8276±0.0921

Table 5. The Mean and Standard Deviation of Each Cluster

According Table 4, there should be five levels of flight delays, which is consistent with the conclusion of the domain experts. Intuitively, these five levels should be very low delays, low delays, moderate delays, significant delays, excessive delays, respectively. The mean and the standard deviation of each cluster is shown in Table 5.

For each formula $x \pm y$ in the grids of Table 5, x and y denotes the mean and standard deviation, respectively. As a learning result from data, these levels may fit the data better than the recent subjective standard.

4. Train the Supervised Learning Methods

The unsupervised learning model builds the levels of the flight delays, which could act as the attribute “class” of the records. With this attribute, we can train the classification models as the alarming model. Notice that it is not always validity to training classification models with the attribute built by clustering. The clustering is a process to build the classes (clusters) by the given relation (similarity) of the data while the classification is to find the relation of the data by the given classes. Thus, the clustering and the classification are the inverse processes of each other in some sense. The relation found by the classification does just fit to the class. If the class is built by clustering, the one which relation fits to is just the relation on which the clustering is based. Then the relation learning by the classification on the same data set may be invalid. In the application of alarming large scales of flight delays, the supervised learning method is used to find the relationship between the recent status and the future delays, which is unknown before training. Therefore, the unsupervised method and the supervised method in this application are not on the same training data set. According to the theory of machine learning, the result of the learning method fits both the a priori knowledge and the data. Therefore, the a priori knowledge is an important factor of the effective of learning result. Usually, the a priori knowledge of the supervised learning is the structure of the model. Thus, we must choose a suitable model in order to meet with good results. The decision tree model, Bayesian learning model and the artificial neural networks model are the most common classification methods. We will try to build the alarming model based on these methods. Before the application, there is a brief introduction of these method in the following.

The problem of constructing a decision tree can be expressed recursively. First, select an attribute to place at the root node and make one branch for each possible value. This splits up the example set into subsets, one for every value of the attribute. Now the process can be repeated recursively for each branch, using only those instances that actually reach the branch. If at any time all instances at a node have the same classification, stop developing that part of the tree.

C4.5 builds decision trees from a set of training data, using the concept of information entropy. At each node of the tree, C4.5 chooses one attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. Its criterion is the normalized information gain (difference in entropy) that results from choosing an attribute for splitting the data. The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithm then recurses on the smaller sublists.

The following shows the main steps of C4.5. First of all, for each attribute a , find the normalized information gain from splitting on a . And then let a_{best} be the attribute with the highest normalized information gain. Create a decision node that splits on a_{best} . At last,

recurse on the sublists obtained by splitting on a_{best} , and add those nodes as children of node.

One highly practical Bayesian learning method is the naive Bayes learner, often called the naive Bayes classifier. The naive Bayes classifier applies to learning tasks where each instance x is described by a conjunction of attribute values and where the target function $f(x)$ can take on any value from some finite set V . A set of training examples of the target function is provided, and a new instance is presented, described by the tuple of attribute values (a_1, a_2, \dots, a_n) . The learner is asked to predict the target value, or classification, for this new instance.

The naive Bayes classifier is based on the simplifying assumption that the attribute values are conditionally independent given the target value. In other words, the assumption is that given the target value of the instance, the probability of observing the conjunction a_1, a_2, \dots, a_n is just the product of the probabilities for the individual attributes.

The study of artificial neural networks has been inspired in part by the observation that biological learning systems are built of very complex webs of interconnected neurons. In rough analogy, artificial neural networks are built out of a densely interconnected set of simple units, where each unit takes a number of real-valued inputs (possibly the outputs of other units) and produces a single real-valued output (which may become the input to many other units)

Backpropagation, or propagation of error, is a common method of teaching artificial neural networks how to perform a given task. It is a supervised learning method, and is an implementation of the Delta rule. It requires a teacher that knows, or can calculate, the desired output for any given input. It is most useful for feed-forward networks (networks that have no feedback, or simply, that have no connections that loop). The term is an abbreviation for "backwards propagation of errors".

The backpropagation neural network works in the following steps. Present a training sample to the neural network. Compare the network's output to the desired output from that sample. Calculate the error in each output neuron. For each neuron, calculate what the output should have been, and a scaling factor, how much lower or higher the output must be adjusted to match the desired output. This is the local error. Adjust the weights of each neuron to lower the local error. Assign "blame" for the local error to neurons at the previous level, giving greater responsibility to neurons connected by stronger weights. Repeat from above step of calculating what the output should have been on the neurons at the previous level, using each one's "blame" as its error.

Usually, the common method to choose models is to analyze the data, especially the intuitive meaning of the attributes. However, there are so few recent researches about the flight data that we cannot judge which model will be better before the experiments. The only way is to train the models and compare the statistical results of each model. The Kappa statistic is a common statistical measure of inter-rater agreement for qualitative (categorical) items. It is generally thought to be a more robust measure than simple percent agreement calculation since Kappa statistic takes into account the agreement occurring by chance. The Equation 6 shows the formal definition of Kappa statistic in the following.

$$K = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)} \quad (6)$$

where $\Pr(a)$ is the relative observed agreement among raters, and $\Pr(e)$ is the hypothetical probability of chance agreement, using the observed data to calculate the probabilities of each observer randomly saying each category. If the raters are in complete agreement then $K=1$. If there is no agreement among the raters (other than what would be expected by chance) then $K \leq 0$.

Besides, the mean absolute error, the root mean squared error, the relative absolute error and the root relative squared error are other common indexes to measure the learning result. These indexes could be easily found in the statistical books so that we do not show the formulas here.

We will try to train the above three supervised learning models on the data with the classes built by the unsupervised method, and then choose the best one. The training data set is built based on the normalized data of each day after 8:00 in some year of *Airport A*. Table 6 shows the statistical results of each model.

Models	Incorrectly Classified Instances	Kappa statistic	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
Naive Bayes	54.5205%	0.2967	0.2266	0.3966	75.0904%	102.1397%
C4.5 Decision Tree	20.274%	0.7291	0.1212	0.2462	40.1601%	63.3979%
BP Neural Network	38.3562%	0.4771	0.2075	0.3214	68.7521%	82.7804%

Table 6 Statistical Results of Supervised Learning Models

As Table 6 shows, the C4.5 decision tree model is the best one according to the statistical values. Thus, we can use this model to predict the flight delays. It may be the final step of laboratory experiment that the model has been trained ready. However, this is an engineering application, but not a laboratory experiment. In the engineering application, the intuitive meaning of the model is required in order to check the validity by experiences. The decision tree is not intuitive enough to be read. A common way to solve this problem is to convert the decision tree to rules, which is very easy to read. And then the rules may be modified by the domain experts in the applications. The finally model to alarm large scale of flight delays is built by these checked and modified rules.

Since the alert of large scale of flight delay is a momentous decision, the alarming model must be controllable and explainable. Good statistical result is only a necessary condition. This is very important in the engineering applications, but usually not required in the laboratory. Compare with the Bayesian models and the artificial neural network models, the decision tree models could be explained more intuitively. Then the users could check whether it needs to be modified by their domain knowledge and experience. Thus, the model with clearly intuitive meaning is always chosen in the applications.

5. Conclusion

This chapter shows an example to use machine learning method in applications. Limited by the space of pages, some details have been skipped to get the main point. As a supplement, we will show some reference in which the details could be found. The flight delays will cause a series of serious consequences. However, it is very difficult to predict flight delays accurately. Some models have been presented to describe the flight delays, such as Milan Janic's disruption model (Janic, 2005), Ning Xu's Bayesian network model (Xu, 2007) and Zonglei Lu's decision tree model (Lu et al, 2008a) and so on. However, there is no model could predict the flight delays accurately up to now. These models could give only some reference of the prediction.

There are some machine learning methods mentioned in this chapter. The systematic introduction about these methods could be found in the classical machine learning books, such as (Mitchell, 1997) and (Witten & Frank, 2005). For more details, the k -means clustering algorithm is first mentioned in (Jain, 1967). The Dunn's index, Davies-Bouldin's index, CS index and Lu's index has been introduced by (Dunn, 1974), (Davies & Bouldin, 1979), (Chou et al, 2004) and (Lu et al, 2008b), respectively. Some details about these clustering validity indexes could also be found in (Halkidi et al, 2002), a survey of clustering validity index. The C4.5 decision tree model, the naive Bayes model and the BP neural network model is mentioned in (Quinlan, 1993), (John & Langley, 1995), (Werbos, 1974), respectively.

6. References

- Chou C.H; Su M.C & Lai. E. (2004). A new cluster validity measure and its application to image compression. *Pattern Analysis and Applications*, Vol. 7, No. 2. (June 2004) pp.205-220, ISSN: 1433-7541.
- Davies D.L. & Bouldin D.W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 1, No. 2 (April 1979) pp.224-227, ISSN: 0162-8828
- Dunn, J.C. (1974). Well Separated Clusters and Optimal Fuzzy Partitions. *Journal of Cybernetica*, 1974, Vol. 4, No. 1 (January 1974) pp.95-104. ISSN: 0022-0280
- Halkidi M.; Batistakis Y. & Vazirgiannis M. (2002). Clustering validity checking methods: part II. *SIGMOD Record*, Vol. 31, No. 3 (September 2002) pp.19-27
- Janic, M. Modeling the Large Scale Disruptions of an Airline Network. *Journal of Transportation Engineering*. Vol. 131, No. 4 (April 2005) pp.249-260. ISSN: 0733-947X
- Jone, G.H. & Langley, P. (1995). Estimating Continuous Distributions in Bayesian Classifiers. *Proceedings of the Eleventh Conference on of Uncertainty in Artificial Intelligence*. pp. 338-345, ISBN: 9781558603851, Montreal CA, Morgan Kaufmann, San Francisco, USA
- Lu Z.L.; Wang J.D. & Li Y. (2008a). An index of cluster validity based on modal logic. *Journal of Computer Research and Development*. Vol. 45 No. 9. (September 2008) pp.1477-1485. ISSN: 1000-1239
- Lu Z.L.; Wang J.D. & Zheng G.S. (2008b). A New Method to Alarm Large Scale of Flights Delay Based on Machine Learning. *Proceeding of 2008 International Symposium on Knowledge Acquisition and Modeling*, pp.589-592, ISBN: 9780769534886, Wuhan China, (December 2008), IEEE CS, New York US.

- Ma Z.P. & Cui D.G. (2004). Optimizing airport flight delays. *Journal of Tsinghua University (Science and Technology)*, Vol.44 No.4, (April 2004) pp.474-477,484. ISBN: 1000-0054
- Mitchell T.M. (1997) *Machine Learning*, McGraw-Hill, ISBN: 0070428077, New York US.
- Quinlan, R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, ISBN: 1558602380, San Mateo, CA.
- Werbos, P.J. (1974). Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. PhD thesis, Harvard University.
- Witten, I.H. & Frank, E. (2003). *Data Mining: Practical Machine Learning Tools and Techniques Second Edition*. Morgan Kaufmann, ISBN: 9780120884070
- Xu, N.; Laskey, K.B.; Chen, C.H. et al (2007). Bayesian Network Analysis of Flight Delays. *Proceeding of Transportation Research Board 86th Annual Meeting*. Washington DC, US.

Machine Learning Tools for Geomorphic Mapping of Planetary Surfaces

Tomasz F. Stepinski¹ and Ricardo Vilalta²

¹Lunar and Planetary Institute, Houston, TX

²University of Houston, Houston, TX
USA

1. Introduction

Terrain topography carries information that is fundamental for geomorphic modeling and, ultimately, for understanding geologic processes responsible for land-surface form. Classification of terrain is tantamount to organizing the expressions (features) of terrain topography into landforms (classes) – patches of topography having similar characteristics and commonly recognizable semantic labels. Because of the spatial character of topographic data, such classification is referred to as geomorphic mapping. The result of classification of an entire landscape scene into a set of mutually exclusive and exhaustive landform classes is referred to as a geomorphic map.

Geomorphic mapping of terrestrial and planetary surfaces has been done traditionally via visual interpretation of images (Wilhelms, 1990; Tanaka, 1994). This manual method is slow, labor intensive, and suffers from subjectivity. Presently, remote sensing instruments onboard spacecrafts are providing increasingly large volumes of topographic data related to Earth as well as surfaces of other planets. This data rich environment challenges the ability of the geosciences community to turn a significant portion of all collected data into products (like, for example, geomorphic maps) that could be utilized in research. Simply put, advances in geomorphic mapping have not kept up with advances in data collection. If left to manual mapping, the percentage of planetary surfaces mapped to the level of detail permitted by an increased resolution of newly collected data will continue to drop precipitously. In order to prevent this decline in rate of data to map conversion, it is necessary to automate the mapping process. Fortunately, the surface properties that distinguish between different landforms can be described quantitatively by a set of numerical measures called terrain attributes which are derived from a digital elevation model (DEM) of the surface. This opens the opportunity for automation of the mapping process. The topic of auto-mapping landforms from topography has received some attention in the geosciences literature; however, such approaches rely frequently on hand-made rules for classification and are designed exclusively for terrestrial applications.

Machine learning can play a vital role in automating the process of geomorphic mapping. A learning system can be employed to either fully automate the process of discovering meaningful landform classes using *clustering* techniques; or it can be used instead to predict

the class of unlabeled landforms - after an expert has manually labeled a representative sample of the landforms - using *classification* techniques. We refer to the two techniques as unsupervised and supervised learning, respectively. Unsupervised learning techniques are applicable in cases of *exploratory* mapping, where no prior knowledge about the surface exists and both landform types and their spatial presence need to be derived by an algorithm. Exploratory mapping finds application in expediting creation of geologic maps in planetary science context where surfaces are still being explored and landform classes are not yet defined. Supervised learning techniques are applicable in cases of *exploitation* mapping when only the spatial presence of a priori defined landform classes is required. Exploitation mapping finds application in creating final geomorphic maps for terrestrial and planetary sites for which constituting landform classes are known a priori.

In developing machine learning-based mapping tools we face a number of design choices, starting from the selection of a basic unit of surface, through the choice of features (terrain attributes), to a pick of an appropriate machine learning technique. The fundamental problem is to design a technique that results in a map that has information content and visual esthetics similar to those found in manually drawn maps; such outcome ensures a large impact in the geosciences community and, consequently, has the greatest scientific value.

2. Preliminaries and previous work

All our tools are based on topographic data which provides a fundamental description of a surface and is well-suited for automated mapping. Topographic data is available as a grid-based DEM, a raster that stores site's elevation value at each pixel in a corresponding raster node. All features used by our machine learning-based mapping tools are derived from the DEM. These features are divided into at-pixel features and area statistics features (Evans, 1998). At-pixel features, except for elevation itself, require a small region or neighborhood around the pixel to calculate their values. Area statistics features depend on the range or distribution of values within the selected, larger neighborhood.

Previously published methods for auto-mapping of landforms can be divided into those that utilize machine learning and those that don't (Gallant et al, 2005; Dragut and Blaschke, 2006; van Asselen and Seijmonsbergen, 2006; Iwahashi and Pike, 2007). Machine learning-based methods can be further grouped into those using unsupervised learning techniques (Irvin et al, 1997; Burrough et al, 2000; Adediran et al, 2004; Stepinski and Vilalta, 2005; Bue and Stepinski, 2006; Ehsani and Quiel, 2008;) and those using supervised learning (Brown et al, 1998; Hengl and Rossiter, 2003; Prima et al, 2006; Stepinski et al, 2006; Stepinski et al, 2007). Moreover, all methods can be grouped into pixel-based methods (Irvin et al, 1997; Brown et al, 1998; Burrough et al, 2000; Hengl and Rossiter, 2003; Adediran et al, 2004; Stepinski and Vilalta, 2005; Bue and Stepinski, 2006; Prima et al, 2006; Iwahashi and Pike, 2007; Ehsani and Quiel, 2008), where an algorithm assigns landform label for each pixel in a DEM separately, and segmentation-based methods (Dragut and Blaschke, 2006; van Asselen and Seijmonsbergen, 2006; Stepinski et al, 2006; Stepinski et al, 2007; Ghosh et al, 2009), where an algorithm assigns landform labels to multi-pixel but attribute-homogeneous segments of the landscape. Fig. 1. illustrates the conceptual difference between pixel-based and segmentation-based approaches. Proposed methods differ broadly in classification algorithms and feature selection. We claim that a segmentation-based approach utilizing

supervised or unsupervised machine learning has the potential to generate maps most comparable to manual maps, and thus most useful. Consequently, our own recent efforts have concentrated on such approaches. Here we discuss the spectrum of tools that we have developed for both exploratory and exploitation purposes. We discuss the tools for exploratory mapping that are both, pixel-based and segmentation-based. We also discuss the tools for exploitation mapping, which are exclusively segmentation-based. Our work focuses on mapping the planet Mars, because Mars is the only planet besides Earth for which global topographic data is currently available (Smith et al, 2003). However, the tools are applicable to mapping terrestrial landmass for which a global, high resolution DEM is available. Moreover, these tools are also applicable to mapping the surfaces of planet Mercury and the Moon once the DEMs for these planets become available (Krishna et al, 2009; Araki et al, 2009) in the near future. The resolution of DEMs of planetary surfaces is coarser than the resolution of terrestrial DEMs. This presents unique challenges for auto-mapping their surfaces as the coarse resolution excludes the direct use of area statistics features decreasing the number of features available to a classifier.

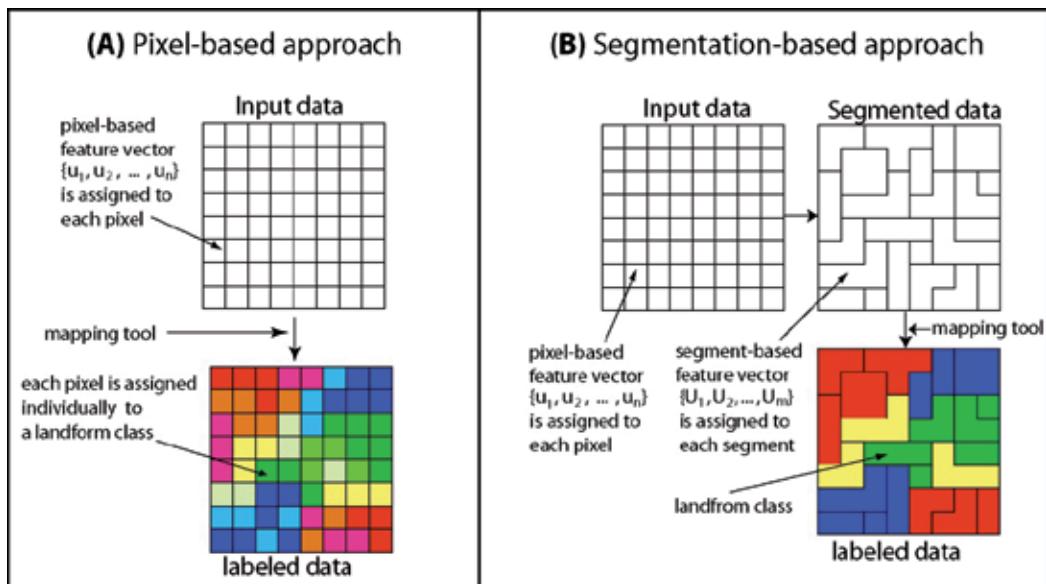


Fig. 1. Two different approaches to assigning geomorphic labels to topographic data

3. Unsupervised learning for exploratory mapping

Development of tools for exploratory geomorphic mapping of Martian surfaces is motivated by a desire for taking stock of all potential landforms present in the site regardless of their semantic meaning. Exploratory mapping is best achieved by unsupervised learning that relies on clustering techniques to automatically discover natural clusters in data. We discuss generating exploratory maps of Martian geomorphology using both, pixel-based and segmentation-based method.

3.1 Pixel-based, unsupervised mapping

In our pixel-based application to exploratory mapping of Mars (Stepinski and Vilalta, 2005), an unsupervised learning algorithm groups pixels that are similar in the space of geomorphic features. The choice of features is dictated by the type of surface to be mapped. A large portion of Martian surface consists of cratered plateau; planetary geomorphologists are interested in mapping various parts of craters, non-crater ridges, linear landforms known as channels, and various types of plateau. This interest dictates the choice of features that are best to discriminate between potential landforms of interest.

In the first tool that we have developed (Stepinski and Vilalta, 2005) the mapping is based on six features (terrain attributes). The first feature, u_1 , is the elevation itself. The second feature, u_2 , is a “flooding adjustment”; In order to calculate u_2 we first artificially modify the original elevation using the so-called “flooding” algorithm (O’Callaghan and Mark, 1984). It identifies all enclosed topographic depressions and raises their elevation to the level of the lowest pour point around their edge thus producing a “flooded” elevation field. The flooding adjustment (u_2) is the difference between flooded and original elevations; it has non-zero values only for pixels located inside topographic depressions (craters). The third feature, u_3 , is the steepest slope between a focus pixel and the eight of its nearest neighboring pixels calculated using the original elevation field. The fourth feature, u_4 , is the steepest slope calculated using the flooded elevation field. The fifth feature, u_5 , is a contributing area. The contributing area is the total number of pixels “draining” through a focus pixel; the term draining is used here as a metaphor for connectivity between different pixels in a landscape. A pixel counts toward the contributing area of a focus pixel if there is a chain of steepest slope directions linking it to the focus pixel. Small values of u_5 flag pixels located on topographic peaks, ridges, and divides. Large values of u_5 flag channels. Finally, the sixth feature, u_6 , is the contributing area based on the flooded elevation field.

The set of six features $\{ u_1, u_2, u_3, u_4, u_5, u_6 \}$ is calculated for all pixels in a site. The basic object of clustering is a pixel in the DEM that carries a vector of six features. Two pixels are similar if their feature vectors are close in the sense of Euclidean metric. A clustering algorithm applied for all pixels produces as output a set of k classes, $C_k = \{ c_1, c_2, \dots, c_k \}$ where each class c_i contains a list of pixels that are similar to each other. The set of classes is mutually exclusive and exhaustive. The map is generated by assigning each pixel a color corresponding to its class. In our first implementation (Stepinski and Vilalta, 2005) of our pixel-based exploratory mapping tool we cluster the DEM using probabilistic clustering algorithm that follows the Expectation Maximization (EM) technique (McLachlan and Krishan, 1997). It groups vectors into classes by modeling each class through a probability density function. Each vector in the dataset has a probability of class membership and is assigned to the class with highest posterior probability. The number of classes is calculated using cross-validation (Cheesman and Stutz, 1996). Because a typical Martian DEM of interest contains a large amount of data, a direct clustering via the EM technique is computationally expensive. To alleviate this problem we sample the DEM to create a smaller, initial dataset of pixels. This initial dataset is clustered into C_k using the EM technique. The remaining pixels are classified into C_k using a decision tree learning algorithm (Quinlan, 1993) constructed on the basis of the initial dataset.

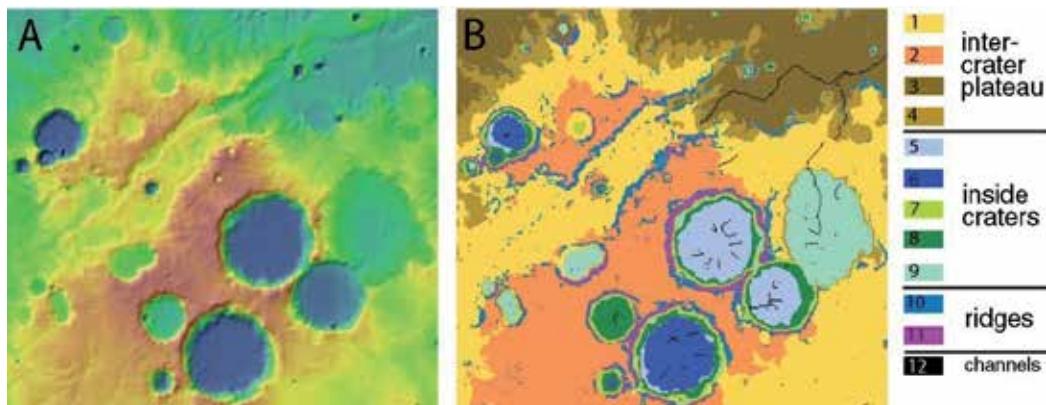


Fig. 2. (A) Topography of Tisia Valles site on Mars. (B) The 12-class geomorphic map created by a pixel-based tool using a probabilistic clustering algorithm.

In order to demonstrate the utility of our tool for producing an exploratory geomorphic map of landscape on Mars we have applied it to a test site called Tisia Valles. The six-feature vector was calculated for each of the site's 163,240 pixels. Because different features have different physical meaning and different range of values, we have normalized all features so that their values are in the range (0, 1). This normalization assures that every feature contributes with equal weight to the "distance" between different pixels. The 40,000 pixels were randomly chosen to create an initial dataset. We have assured uniform sampling in order to obtain an unbiased representation of all, even rare landscape features. Our clustering algorithm has grouped these 40,000 pixels into 12 separable and exhaustive clusters. The remaining pixels were classified into those 12 clusters using a decision tree algorithm.

Fig. 2A. shows the topography of the test site; red-to-blue gradient indicates high-to-low elevation. Fig. 2B. shows a geomorphic map generated by our tool; different landform classes (clusters of similar feature vectors) are shown using different colors. The semantic interpretation of these classes requires expert judgment; an analyst needs to review statistics of feature vectors values in each class and spatial distribution of classes with respect to each other. A simplified result of such interpretation is given in the legend of Fig. 2. An analyst divided the 12 classes into 4 different groups pertaining to plateau, craters, ridges, and channels, respectively. Some groups, for example the plateau group, may include several landforms classes. An expert grouped the four classes (labeled 1, 2, 3, and 4) into the plateau group because they are identical from a geomorphic point of view, just located at different elevations. This example illustrates a "problem" with mapping based on the principle of unsupervised learning - a reasonable cluster derived under a proximity measure may not constitute a "novel" landform as perceived by an analyst. Nevertheless, in face of lack of any previous knowledge about the site's landscape, unsupervised learning delivers valuable, first draft information.

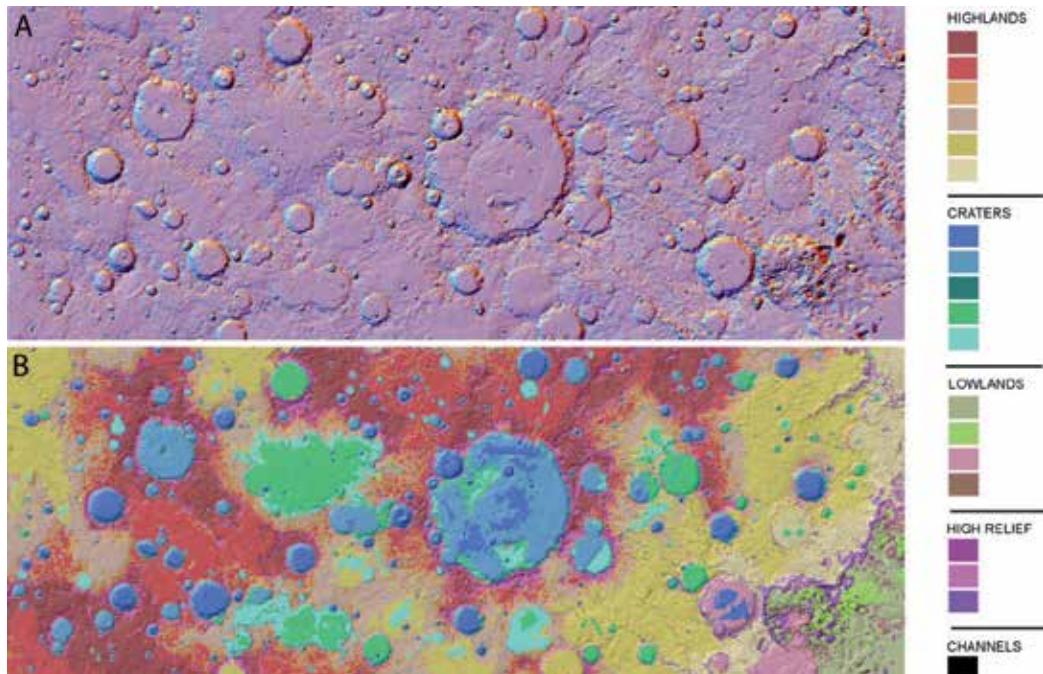


Fig. 3. (A) Topography of Terra Cimmeria site on Mars. (B) The 20-class geomorphic map created by a pixel-based tool using a SOM-Ward clustering algorithm.

Our tool for exploratory mapping using pixel-based unsupervised learning technique was further modified (Bue and Stepinski, 2006) to achieve greater computational efficiency necessary for mapping larger sites. The clustering efficiency was significantly improved by using a two-level clustering procedure (Vesanto and Alhoniemi, 2000) consisting of a self-organizing map (SOM) (Kohonen, 1995) and the Ward hierarchical clustering (Ward, 1963). The SOM is a neural network technique that groups similar vectors into nearby points on a 2-D grid composed of nodes. Through an unsupervised, iterative procedure, a large set of feature vectors is mapped onto the much smaller number of SOM's nodes in such a way that similar feature vectors are associated with the same node or neighboring nodes. Because the number of nodes is much smaller than the number of vectors, many similar vectors are mapped onto a single node. The bundle of feature vectors associated with a given SOM node is typified by a single representative vector referred to as a codebook vector. The final clustering of feature vectors into an assigned number of k clusters (landform classes) is achieved by the Ward's minimum variance grouping algorithm (Ward, 1963) applied to the set of codebook vectors.

Fig. 3A. depicts a topography of a 5,303,888 pixels-large site on Mars referred to as Terra Cimmeria. We used 30 x 30 rectangular SOM grid to perform a first step of clustering the feature vectors associated with those pixels into 900 codebook vectors. In the final step the codebook vectors were clustered into 20 landform classes shown on Fig. 3B using different colors. As in the previous example, the semantic interpretation of the classes requires expert judgment; a simplified result of such interpretation is given in the legend of Fig. 3.

3.2 Segmentation-based, unsupervised mapping

Our second tool for exploratory mapping via unsupervised learning combines aspects of pixel-based and segment-based mapping approaches (Stepinski and Bagaria, 2009). We constructed a two-stage algorithm consisting of a pixel-based *base* classifier and a segment-based *meta* classifier. A base classifier is applied to multiple pixels in a neighborhood of a focus pixel resulting in an ensemble of landform type predictions. A meta classifier is an unsupervised segmentation/classification algorithm that combines these predictions and outputs a segment-based map of emergent landform regions or classes. This tool is designed for exploratory mapping of very large regions using small number of original features.

In order to increase a computational efficiency of our tool we utilize a rule-based classifier as our base classifier. The rule-based classifier uses empirical knowledge to construct a decision tree; submitting a set of terrain features to a trunk of the tree results in a landform type label at the leave of the tree. The nested means technique (Iwahashi and Pike, 2007) is used to construct a decision tree because it outputs landform types whose meanings do not correspond directly to named terrestrial formations, thus, they won't lose their relevance in application to non-terrestrial surfaces. Our rule-based classifier uses only three original terrain features (slope gradient, surface texture, and local convexity) to label each pixel into one of 16 statistically predefined landform types.

The segment-based meta classifier uses a set of secondary features designed to capture contextual information around a given pixel. The secondary features are calculated from the labels (1 to 16) returned by the base classifier; they are combined into a pixel-attached feature-vector which describes, in a generalized manner, surface character in the neighborhood of this pixel. We calculate 19 secondary features. The first 16 features are normalized frequencies of labels outputted by the base classifier contained within a $N \times N$ pixels square window centered on the focus pixel. The value of N controls the level of generalization from landform types to landform classes. Two windows may have similar frequencies but different spatial distributions of the labels. The last three secondary features measure pattern of landform types in a neighborhood and are based on a modification of Multi-Scale Local Binary Pattern (LBP) concept (Ojala et al., 2002). The 19-dimensional vector of secondary features is used to generate a final segmentation-based map.

We use the Recursive Hierarchical Segmentation (RHSEG) algorithm (Tilton, 2000) that *simultaneously* segments the DEM on the basis of secondary features and cluster the segments into landform classes. The RHSEG is an iterative algorithm that produces hierarchies of both, segmentation levels, and clustering levels. Starting from individual pixels as regions seeds, the algorithm alternates between merging similar adjacent regions into larger regions (segmentation) and merging labels of non-adjacent similar regions (clustering). Both steps utilize similarity criteria based on statistics of secondary features of pixels constituting the segments. As this two-step process is iteratively repeated, it produces a natural hierarchy of both, spatial segmentations and clusters of features. Stopping RHSEG at a given iteration level yields a map of a certain geographical and feature-space resolution.

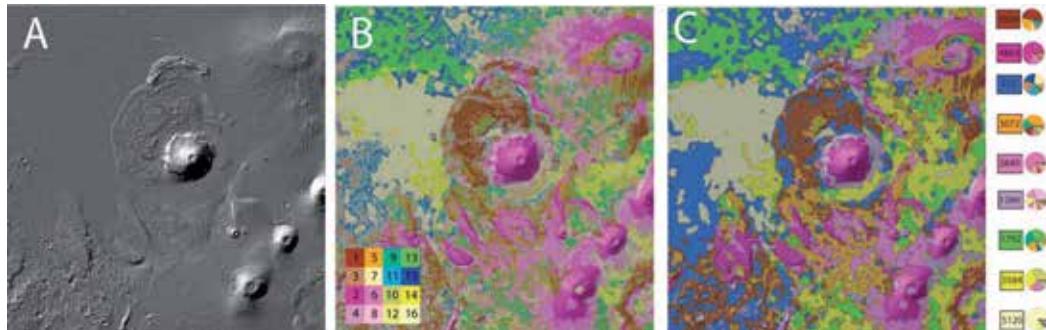


Fig. 4. (A) Topography of Tharsis region on Mars. (B) The results of rule-based classification. (C) The 9-class geomorphic map created by the meta classifier.

In order to demonstrate the utility of our segmentation-based exploratory mapping tool we have applied it to the Tharsis region on planet Mars. The Tharsis region on Mars (Fig. 4A) is an enormous volcanic plateau containing a number of large volcanoes including Olympus Mons – the largest volcano in the solar system. We used a 1024 x 1024 pixels DEM of Tharsis region with the resolution of 4 km/pixel. The base classifier labeled each of 1,048, 576 pixels with one of 16 labels resulting in a pixel-based map as shown on Fig. 4B. The legend to Fig. 4B is organized in a square array (see insert): the top row groups terrain types (1, 5, 9, 13) representing rough, convex landscapes; the second row groups terrain types (3, 7, 11, 15) standing for rough, concave landscape; the third row groups terrain types (2, 6, 10, 14) representing smooth, convex landscapes; the last row groups terrain types (4, 8, 12, 16) corresponding to smooth, concave landscapes. In each row progressively larger values of labels indicate gentler landscape. The secondary features are calculated using $N=11$ pixels moving window. We set the parameters of RHSEG algorithm so it starts saving the segmentation results when the feature-vectors are already clustered into 20 landform classes. This most-detailed of all retained partitioning is referred to as level 0 segmentation. Subsequent, progressively coarser segmentations are referred to as level 1 to 19, respectively. Fig. 4C shows the level 11 segmentation consisting of 2382 segments grouped into 9 landform classes. The legend to Fig. 4C shows a color and a numerical label assigned to each landform class.

Notwithstanding superficial visual similarity (this similarity decreases rapidly when the close-ups of the two maps are examined) between the map generated by the base classifier (Fig. 4B) and the map generated by the meta classifier (Fig. 4C), the map generated by RHSEG algorithm has a higher utility because it partitions a site in a fashion similar to what an analyst would do manually – into fewer larger, more heterogeneous areas corresponding to more broadly defined landform classes. The small pie diagrams next to label annotations in a legend to Fig. 4C indicate a “composition” of each landform class in terms of types outputted by the base classifier. Different classes are characterized by different degrees of terrain type inhomogeneity reflecting the reality of natural landscape. The visual esthetics of the map shown on Fig. 4C resembles manually drawn geologic maps, however, direct, formal comparison of our map with a manually drawn geologic map is difficult because analysts use not only objective criteria (such as, surface morphology) but also subjective criteria (such as, nomenclature, history of previous investigations, etc.). Nevertheless, our

map shown on Fig. 4C shows a rough correspondence to a manually drawn geologic map of the Tharsis region (Scott and Tanaka, 1986).

4. Supervised learning for exploitation mapping

In many cases planetary scientists know a priori what landform classes they want to map in a given site. Automation of such exploitation mapping should not be based on the principle of unsupervised learning that offers no control over the character of outcome classes; instead, it should be based on the principle of supervised learning where classes are set a priori. Recognizing that automatically generated maps must conform to expectations of the planetary science domain, our efforts to automate the process of exploitation mapping focus on the concatenation of a segmentation-based technique with supervised learning (Stepinski et al, 2006; Stepinski et al, 2007; Ghosh et al, 2009). The idea of segmentation-based classification follows from the realization that pixels are not the best fundamental units of visual or topographic scenes, and it is more natural and efficient to work with more perceptually meaningful entities obtained from low-level grouping processes. Such entities are referred to as superpixels (Mori, 2005) in the computer vision community and as segments (Benz et al, 2004) in the remote sensing community. The diagram in Fig. 1B illustrates the concept of segmentation-based classification of landform classes. The segmentation-based classification technique has many desired properties: a) segments are perceptually meaningful, b) they are computationally efficient, c) their geometric and statistical properties provide additional information that can be incorporated into classification, d) because the technique results in oversegmentation of the site, most structures in the site are conserved and there is little loss of information over using individual pixels.

We have developed a family of tools for automating exploitation mapping of planetary surfaces; each tool utilizes a specific combination of segmentation and classification algorithms. We employ two different segmentation algorithms. The dividing algorithm splits the landscape on the basis of abrupt discontinuities in pixel-based feature vectors. The agglomerative algorithm initially treats each pixel as an individual segment; these initial segments are combined into larger segments as long as a user-defined criterion for the uniformity of constituent pixel-based feature vectors holds. Both algorithms use the same pixel-based feature vectors. We also employ three different learning algorithms for segment classification and to generate maps of landforms. Thus, altogether, we have evaluated six different tools, corresponding to six different segmentation/classification combinations.

4.1 Segmentation methods

The segmentation procedure subdivides the landscape into mutually exclusive and exhaustive segments containing pixels having approximately uniform pixel-based feature vectors. These segments constitute topographic objects, which, subsequently, are classified into landforms classes. Raster segmentation has been the subject of intense study in the domain of image analysis, however, requirements for an effective segmentation for the purpose of mapping are different from those encountered in the field of computer vision. In particular, for the purpose of mapping-by-classification, it is desirable to oversegment the site. Having small segments eliminates the danger of a particularly large segment being misclassified, which would avoid producing a grossly incorrect map. Moreover, having

approximately equal-sized segments assures that statistics of pixel-based features are calculated from comparable ensembles of member pixels.

Our dividing segmentation algorithm (Stepinski et al, 2006) uses the watershed transform (Beucher, 1992) applied to a gray-scale image that encapsulates gradients of pixel-based feature vectors. This image is calculated using a computationally simple homogeneity measure H (Jing et al, 2003). A pixel located in a region that is homogeneous with respect to pixel-based features has a small value of H . On the other hand, a pixel located in a region which is inhomogeneous with respect to features has a large value of H . A raster constructed by calculating the values of H for all pixels in the landscape can be interpreted as a gray-scale image and is referred to as the H -image. White areas in H -image represent boundaries of homogeneous regions, whereas the dark areas represent the actual regions. The watershed transform of H results in (over) segmentation of the H -image (and thus the landscape).

Our agglomerative segmentation algorithm (Stepinski et al, 2007) uses a contiguity-enhanced variant of the standard K -means clustering algorithm, which uses – in addition to terrain attributes – spatial coordinates of pixels as features. The additional spatial features control the size of the segments while providing the resultant segments with very desirable geometric properties. For example, in areas where terrain features are approximately uniform, the local gradient of the total feature vector is dominated by changes in spatial coordinates leading to the formation of round-shaped segments. On the other hand, in areas where change in the total feature vector is dominated by change in terrain attributes, segments tend to exhibit an elongated shape in direction perpendicular to the gradient of the terrain-only sub-vector. These properties constitute additional knowledge that could be exploited by the classification module. The actual segmentation invokes a simple K -means algorithm applied to spatially-enriched, pixel-based feature vectors. The size of the segments is controlled by the value of K (which needs to be large to achieve over-segmentation).

4.2 Application of segmentation methods

In order to demonstrate the working of segmentation algorithms in practice we applied them to the Tisia Valles site on Mars (see Section 3.1). The site is segmented on the basis of three pixel-based terrain features $\{ u_1, u_2, u_3 \}$ using both, watershed and K -means, algorithms. Note that the features used here are different from those we choose for exploratory mapping (see Section 3.1); they are: u_1 =slope, u_2 =curvature, and u_3 =flooding adjustment. The watershed algorithm produced 7708 segments with sizes ranging from 1 to 267 pixels, whereas K -means algorithm (with the value of $K = 5000$) produced 6593 single-connected segments having sizes ranging from 4 to 117 pixels. Note that the K -means algorithm yields more than K segments because the resulting K clusters do not correspond to K single-connected spatial segments. In order to derive the segmentation we assign a unique segment identifier to each subset of a cluster corresponding to a single spatially connected region. The two algorithms yield segmentations having very different characters (see Fig. 5.). The watershed algorithm yielded a mosaic of segments that, by themselves, do not reveal the landforms present in the site. On the other hand, the inclusion of spatial coordinates into the K -means algorithm resulted in segments that reflect the geometry of the landforms – one can notice the major landforms just from the segmentation image.

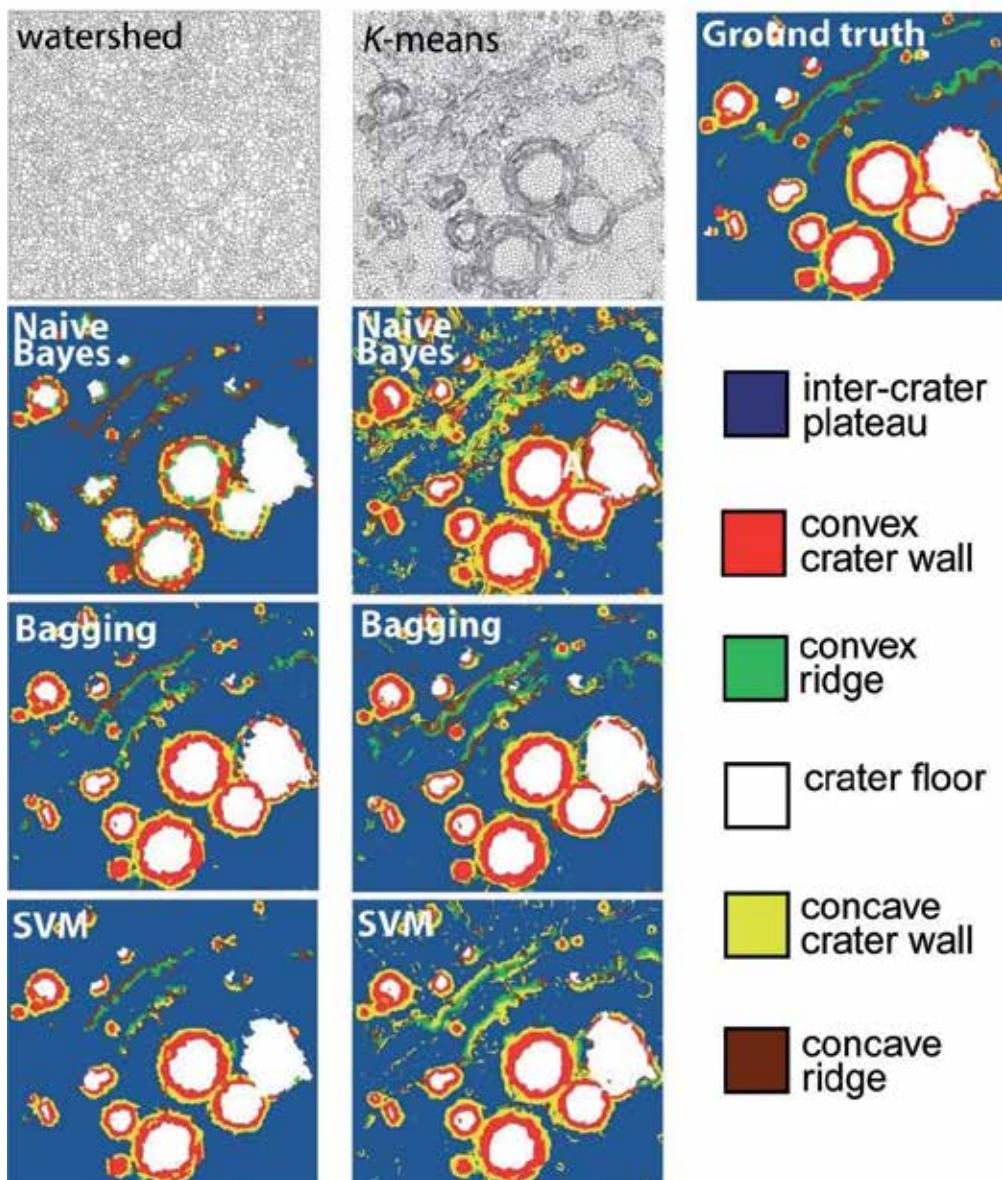


Fig. 5. Six-classes geomorphic maps Tisia Valles site using different combination of segmentation and classification algorithms.

4.3 Segment-based features

In the segmentation-based classification, pixel-based features used for segmentation are different from segment-based features used for classification. Each segment, regardless of an algorithm used to obtain the segmentation, is represented by a combination of physical and spatial segment-based features. Physical features are pixel-based features averaged over the

constituent pixels of the segments. Spatial features are obtained using each segment's shape measure and the neighborhood context measure. The shape measure is computed in terms of the Shape Complexity Index (SCI). The SCI is a measure of segment circularity. The closer the value of SCI is to 1.0, the more circular the object; on the other hand, thin ring-like shapes tend to have SCI values of 2.5 and higher. One of the challenges of the automatic classification of landforms is feature similarity of some landform classes that differ mostly by their spatial context. For instance, segments making up craters' walls and segments constituting ridges not associated with craters may have similar values of slope, curvature, but are located in different spatial contexts. In our segmentation-based tool we take into consideration spatial context by means of neighborhood context measures. Ideally, we would like to know classes of segment's neighbors to establish its spatial context, but such information is not available prior to classification. However, we can categorize the unlabeled segments into low, medium, and high categories based on statistics of the values of their physical features. Such categorization is used to calculate the neighborhood property of each segment using a nine-dimensional vector $\{a_h^s, a_m^s, a_l^s, a_h^c, a_m^c, a_l^c, a_h^f, a_m^f, a_l^f\}$, where a_j^i $j=h, m, l$ and $i=s$ (slope), c (curvature), f (flooding adjustment) is a percentage of the focus segment boundary with neighbors belonging to category high (h), medium (m), or low (l), respectively. Thus, a segment-based feature vector has 13 components, three physical features, the value of SCI, and 9 values of a_j^i .

4.4 Classification and mapping

We applied three different learning algorithms for segment classification and to generate geomorphic maps. First, the simple Naive Bayes algorithm provides a baseline for comparison with other classifiers. Second, the Support Vector Machines (SVM) algorithm that works by finding an optimal hyper-plane in a (transformed) feature space (Boser et al, 1992). The optimal hyper-plane maximizes the separation between classes. SVM exploits local data patterns and has been found to be effective in spatial data mining applications (Sharifzadeh et al, 2003). Third, bagging ensemble learning algorithm (Breiman, 1996) generates multiple models by running a single learning algorithm multiple times over bootstrapped samples of the training set. The final class label is the result of voting over the contributing models (one from each bootstrap sample). Bagging is known to work well for complex datasets and is particularly attractive when the training set is noisy (Dietterich, 2000). We use a decision tree (C4.5) as the base learner in the bagging algorithm.

We applied these classifiers to segments generated by watershed and K-means generated divisions of the Tisia Valles site. We have chosen six landform classes for mapping: crater floors, convex crater walls, concave crater walls, convex ridges, concave ridges, and inter-crater plateau. The choice of these particular landform classes stems from our interest in the quantitative characterization of old, cratered Martian surface. The labeled (training) set of segments was generated by manually labeling 30% (by surface area) of the Tisia site into the aforementioned six classes. Fig. 5 offers a visual assessment of the maps generated by different combination of segmentation and classification algorithms. The "ground truth" map of Tisia (an extension of the training set to the entire site) was hand-labeled. It shows how a typical analyst would map the six landforms in this site; it does not really constitute a ground truth (in the strict meaning of the concept) because an analyst is likely to draw an idealized map that misses details and projects a human conceptualization of the entire landscape, even if it contradicts local measurements. Maps based on the watershed

segmentation have a “simple” look as they lack small-scale details, whereas maps based on the K-means segmentation look exhibit more small-scale details. On the basis of only a visual inspection one could conclude that maps stemming from watershed segmentation are “better” because they look more like the ground truth map. However, closer inspection of the generated maps shows that maps based on K-means segmentation correctly reflect some small-scale details that are absent from the watershed segmentation and the analyst-drawn map. The maps generated by Naïve Bayes are inaccurate and inferior to maps generated by Bagging and SVM.

watershed based segmentation							
classifier	overall accuracy	plateau	floor	cvx. wall	con. wall	cvx. ridge	con ridge
NB	84.67	0.94/0.97	0.93/0.93	0.87/0.39	0.57/0.37	0.12/0.12	0.20/0.51
B	89.31	0.95/0.96	0.97/0.87	0.80/0.83	0.65/0.75	0.51/0.45	0.38/0.37
SVM	89.71	0.92/0.98	0.96/0.90	0.87/0.75	0.72/0.70	0.62/0.29	0.48/0.29
K-means based segmentation							
classifier	overall accuracy	plateau	floor	cvx. wall	con. wall	cvx. ridge	con ridge
NB	74.11	0.99/0.76	0.99/0.80	0.70/0.78	0.32/0.83	0.14/0.19	0.08/0.25
B	87.42	0.96/0.92	0.94/0.92	0.80/0.85	0.67/0.72	0.34/0.47	0.25/0.40
SVM	86.10	0.97/0.91	0.98/0.87	0.77/0.79	0.46/0.83	0.46/0.54	0.18/0.12

Table 1. Assessment of performance of different methods used to map the Tisia Valles site. The entries for individual landform are precision/recall. NB – Naïve Bayes, B – Bagging with C4.5, SVM – Support Vector machines.

Table 1 gives accuracy rates for maps of the Tisia site. Disregarding maps produced by the Naïve Bayes algorithm, accuracy rates are above 86%. Note that maps based on the watershed segmentation have slightly higher rates than maps based on the K-means segmentation in line with their greater similarity to the analyst drawing. Precision and recall rates for six landform classes are also given in Table 1. Results show that inter-crater plateau, crater floor, and convex crater walls landforms are mapped with high accuracy. Concave crater walls are detected with less accuracy, and ridges are difficult to identify correctly. This is because local ridges look like crater walls, even though they are different landforms in the context of the entire landscape.

5. Summary and conclusion

Geomorphic auto-mapping of planetary surfaces is a challenging problem. Here we have described how machine learning techniques, such as clustering or classification, can be utilized to automate the process of geomorphic mapping for exploratory and exploitation purposes. Relatively coarse resolution of planetary topographic data limits the number of features that can be used in the learning process and makes planetary auto-mapping more challenging than terrestrial auto-mapping. With this caveat, the methods discussed here are also applicable to terrestrial surfaces.

The major challenge in exploratory (unsupervised learning) mapping is to generate a map that has an appearance and utility similar to maps already used by the geosciences community. This means that a clustering algorithm should be able to generalize from a simple similarity of feature vectors to a similarity of ensembles of feature vectors. In other

words, an algorithm should be able to generate classes of varying degree of homogeneity based on spatial considerations. This is what our algorithm described in section 3.1 has been designed to do. Future research will address better criteria for deciding which classes should be homogeneous and which should be more heterogeneous. Overall, our two-stage tool for exploratory mapping is expected to be adopted by the geosciences community because it is matured enough for immediate application in geologic mapping, quantitative comparative geomorphology, and landscape visualization.

The major challenge in exploitation mapping (supervised learning) is the issue of spatial context. An analyst can map landforms having very similar features as different classes depending on broader spatial context. Thus, spatial context must be incorporated into the mapping algorithm in order to generate maps similar to those that are manually drawn. We have demonstrated that a choice of a particular segmentation method and a particular (capable) classification algorithm results in somewhat different maps, but, in general, all generated maps were acceptable. Indeed, regardless of the segmentation/classification combination, most misclassifications were the results of confusion due to spatial context. Our simple method of taking some account of spatial context proved insufficient to prevent misclassifications between elements of crater walls and elements of ridges. Future work needs to investigate more robust approach, such as, for example, Markov Random Fields, to incorporate spatial context information (Besag, 1986) into the learning algorithm.

Acknowledgements: This work was supported by the National Science Foundation under Grants IIS-0812271 and by NASA under grant NNG06GE57G. A portion of this research was conducted at the Lunar and Planetary Institute, which is operated by the USRA under contract CAN-NCC5-679 with NASA. This is LPI Contribution No. 1492.

6. References

- Adediran, A.O.; Parcharidis, I.; Poscolieri, M. & Pavlopoulos, K. (2004). Computer-assisted discrimination of morphological units on north-central Crete (Greece) by applying multivariate statistics to local relief gradients. *Geomorphology*, Vol.58, pp.357-370.
- Araki, H.; Tazawa, S.; Noda, H.; Ishihara, Y.; Goossens, S.; Kawano, N.; Sasaki, S.; Kamiya, I.; Otake, H.; Oberst, J. & Shum, C.K. (2009). The lunar global topography by the laser altimeter (LALT) onboard Kaguya (Selene): Results from the one year observations. *Proceedings of 40th Lunar and Planetary Science Conference*, #1432.
- Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society B*, Vol.48, pp.259–302.
- Benz, U.; Hoffmann, P.; Willhauck, G.; Lingenfelder, I. & Heynen, M. (2004). Multi-Resolution, Object-Oriented Fuzzy Analysis of Remote Sensing Data for GIS-Ready Information, *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol.58, pp.239-258.
- Beucher, S. (2003). The watershed transformation applied to image segmentation. *Scanning Microscopy International*, Vol.6, pp.299–314.
- Boser, B. E.; Guyon, I. & Vapnik, V. (1992). A training algorithm for optimal margin classifiers. *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pp.144–152.

- Brown, D. G.; Lusch, D. P. & Duda, K. A. (1998). Supervised classification of types of glaciated landscapes using digital elevation data. *Geomorphology*, Vol.21, pp.233-250.
- Breiman, L. (1996). Bagging predictors. *Machine Learning*, Vol.24 No.2, pp.123-140.
- Bue, B. D. & Stepinski, T. F. (2006). Automated classification of landforms on Mars. *Computers & Geosciences*, Vol. 32 No.5, pp. 604-614.
- Burrough, P.A.; van Gaans, P.F.M. & MacMillan, R.A. (2000). High-Resolution landform classification using fuzzy k-means. *Fuzzy Sets and Systems*, Vol.113, pp.37-52.
- Cheesman, P. & Stutz, J. (1996). Bayesian Classification (AutoClass) Theory and Practice. In: *Advances in Knowledge Discovery and Data Mining*, U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy (Eds.), pp.153-180, MIT Press.
- Dietterich, T.G. (2000). Ensemble methods in machine learning. In: *Lecture Notes in Computer Science # 1857*, pp.1-15.
- Dragut, L. & Blaschke, T. (2006) Automated classification of landform elements using object-based image analysis. *Geomorphology*, Vol.81, pp.330-344.
- Evans, I.S. (1998), What do terrain statistics really mean?, In: *Landform monitoring, modelling and analysis*, Lane, S.N., Richards, K.S., and Chandler, J.H., (Eds.), pp. 119-138, J.Wiley, Chichester.
- Ehsani, A.H. & Quiel, F. (2008). Geomorphometric feature analysis using morphometric parameterization and artificial neural networks. *Geomorphology*, Vol.99, pp.1-12.
- Gallant, A.L.; Brown, D.D. & Hoffer, R. M. (2005). Automated mapping of Hammond's landforms. *IEEE Geoscience and Remote Sensing Letters*, Vol.2 No.4, pp.384-288.
- Ghosh, S.; Stepinski, T. F. & Vilalta, R. (2008). Automatic Annotation of Planetary Surfaces with Geomorphic Labels. *IEEE Transactions on Geoscience and Remote Sensing*, submitted.
- Hengl, T. & Rossiter, D.G. (2003). Supervised landform classification to enhance and replace photointerpretation in semi-detailed soil survey. *Soil Science Society of America Journal*, Vol.67, pp.1810-1822.
- Irvin, B.J.; Ventura, S.J. & Slater, B.K. (1997). Fuzzy and isodata classification of landform elements from digital terrain data in Pleasant Valley, Wisconsin. *Geoderma*, Vol. 77, pp. 137-154.
- Iwashashi, J. & Pike, R. J. (2007). Automated classifications of topography from DEMs by an unsupervised nested-means algorithm and a three-part geometric signature. *Geomorphology*, Vol. 86, pp. 409-440.
- Jing, F.; Li, M.; Zhang, H. & Zhang, B. (2003). Unsupervised image segmentation using local homogeneity analysis. *Proceedings of 2003 International Symposium on Circuits and Systems*, pp.II-456-II459.
- Kohonen, T. (1995). *Self-organizing Maps*. Springer, Berlin.
- Krishna, B.G.; Amitabh; Singh, S.; Srivastava, P.K. & Kiran Kumar, A.S. (2009). Digital elevation models of the lunar surface from Chandrayan-1 terrain mapping camera (TMC) imagery - initial results. *Proceedings of 40th Lunar and Planetary Science Conference*, #1694.
- McLachlan, G. & Krishnan, T. (1997) *The EM Algorithm and Extensions*. John Wiley and Sons, New York, NY.
- Mori, G. (2005). Guiding Model Search Using Segmentation. *Proceedings of the Tenth IEEE International Conference on Computer Vision*, pp.1417 - 1423.

- O'Callaghan, J.F. & Mark, D.M. (1984). The extraction of drainage networks from digital elevation data. *Computer Vision, Graphics and Image Processing*, Vol.28, pp.328–344.
- Ojala, T.; Pietikainen, M. & Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 24 No.7, pp.971–987.
- Prima, O.; Echigo, A.; Yokoyama, R. & Yoshida, T. (2006). Supervised landform classification of northeast Honshu from DEM derived thematic maps. *Geomorphology*, Vol.78, pp.373–386.
- Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Francisco.
- Scott, D.H. & Tanaka, K.L. (1986). Geologic map of the western equatorial regions of Mars, *US Geol. Surv. Inv. Series Map*, I-1802 A.
- Sharifzadeh, M.; Shahabi, C. & Knoblock, C. (2003). Learning approximate thematic maps from labeled geospatial data. *Proceedings of International Workshop on Next Generation Geospatial Information*, Cambridge (Boston), Massachusetts, USA.
- Smith, D.; Neumann, G.; Arvidson, R.; Guinness, E. & Slavney, S. (2003). Mars global surveyor laser altimeter mission experiment gridded data record. *NASA Planetary Data System*, (MGS-M-MOLA-MEGDR-L3-V1.0), 2003.
- Stepinski, T.F. & Vilalta, R. (2005). Digital topography models for Martian surfaces. *IEEE Geoscience and Remote Sensing Letters*, Vol. 2 No.3, pp. 260–264.
- Stepinski, T.F.; Ghosh, S. & Vilalta, R. (2006). Automatic recognition of landforms on Mars using terrain segmentation and classification, *Proceedings of Ninth International Conference on Discovery Science*, Lecture Notes in Computer Science # 4265, pp. 255–266.
- Stepinski, T.F.; Ghosh, S. & Vilalta, R. (2007). Machine learning for automatic mapping of planetary surfaces, *Proceedings of The Nineteenth Innovative Applications of Artificial Intelligence Conference*, AAI Press.
- Stepinski, T.F. & Bagaria, C. (2009). Segmentation-based Unsupervised Terrain Classification for Generation of Physiographic Maps, *IEEE Geoscience and Remote Sensing Letters*, in press.
- Tanaka, K.L. (1994). The Venus Geologic Mappers' Handbook, *U.S. Geol. Surv. Open File Rep.* pp.99–438.
- Tilton, J.C. (2000). Method for recursive hierarchical segmentation by region growing and spectral clustering with a natural convergence criterion. In: *Disclosure of Invention and New Technology*, NASA Case Number GSC 14,328-1.
- Wilhelms, D.E. (1990). Geologic Mapping. In: *Planetary Mapping*, Greeley, R.; and Batson, R. (Ed.), pp.209–260, Cambridge Univ. Press, Cambridge, UK.
- van Asselen, S. & Seijmonsbergen, A.C. (2006.) Expert-driven semi-automated geomorphological mapping for a mountainous area using a laser DTM. *Geomorphology*, Vol.78, pp. 309–320.
- Vesanto, J.; & Altoniemi, E. (2000). Clustering of the self-organizing map. *IEEE Transactions on Neural Networks* Vol.11 No.3, pp.586–600.
- Ward Jr., J.H. (1963). Hierarchical grouping to optimize an objective function, Vol.58 No.301, pp.236–244.

Network Intrusion Detection using Machine Learning and Voting techniques

Tich Phuoc Tran¹, Pohsiang Tsai¹, Tony Jan¹ and Xiaoying Kong²

¹*Centre for Innovation in IT Services and Applications (iNEXT)*

University of Technology, Sydney

²*Centre for Real-time Information Networks (CRIN)*

University of Technology, Sydney

Keywords: Network Intrusion Detection, Neural Network, Adaptive Boosting, Multi-Expert Classification.

Abstract:

As the result of recent advent and rapid growth of the Internet, there have been an increasing number of corporations relying on computers and networks for communications and critical business transactions. Because of the network complexity and advanced hacking techniques, such reliance on computer networks often presents unanticipated risks and vulnerabilities. A huge volume of attacks on major sites and networks have been recently reported including those of private companies, government agencies and even military classified networks. Therefore, it is important to deploy protection measures for networks and their services from unauthorized modification, destruction, or disclosure of sensitive information. Intrusion detection systems (IDS) have emerged as an important part of today's network security infrastructure which can monitor the network traffic and detect possible attacks. Currently existing IDS suffer from low detection accuracy and system robustness for new and rare security breaches. To improve detection capability of IDS, this chapter proposes an innovative Machine Learning (ML) framework in which different types of intrusions will be detected with different classifiers, including different attribute selections and learning algorithms. Outputs of these classifiers are then combined by appropriate voting techniques. Experiments on the KDD-99 dataset show that our approach obtains superior performance in comparison with other state-of-the-art detection methods, achieving low learning bias and improved generalization at an affordable computational cost.

1. Introduction

As a result of the revolutionary advances in computing science and the wide spread deployment of the Internet, people are encouraged to communicate and exchange information over the computer-mediated environment. This provides convenience and

benefits such as shortening the effective geographical distances and sharing information efficiently. On the other hand, information exchange in such environments pose a problem, which is that, intruders or malicious users may compromise the communications. The safeguarding of security is becoming even more difficult, because the possible technologies of attacks are very sophisticated; at the same time, less technical ability is required for the novice attackers due to easy access to proven past methods through the Web. A traditional approach to defend computer networks was based on static defense mechanisms in which software, such as the operating system, was kept up-to-date to prevent the exploitation of security holes; and the firewalls deployed at critical network segments to improve the security at the entry level. However, firewalls aim to regulate and control the flow of information into and out of a network rather than detecting whether or not the network is under attack. To complement simple firewalls, Intrusion Detection Systems (IDS) are normally used to gather and analyze network data to identify possible security breaches, which include both intrusions (attacks from outside the organization) and misuse (attacks from within the organization). Although IDS have become an important part of most network security architectures which provides essential second line of defense, the majority of them face a number of challenges such as low detection rates which can miss serious intrusion attacks and high false alarm rates, which falsely classifies a normal connection as an attack and therefore obstructs legitimate user access to the network resources (Sommer, 2008). These problems are due to the sophistication of the attacks and their intended similarities to normal behavior.

Most IDS utilize some Machine Learning (ML) techniques to obtain high detection capability for novel attacks and automation to save human labor from manually constructing signatures of attacks or specifying the normal behavior of a sensor node. Theoretically, it is possible for a ML algorithm to achieve the best performance, i.e. it can minimize the false alarm rate and maximize the detection accuracy; however, this normally requires infinite training sample sizes (Kononenko & Kukar, 2007). In practice, such condition is impossible due to limited computing resources and real-time response requirement of IDS. Intrusion detection, therefore, remains very challenging. In this chapter, a learning framework is proposed to enhance the performance of intrusion detection for rare and complicated attacks; that is, the framework can increase the detection accuracy and decrease false alarm with acceptable computational expenses. In particular, characteristics of different anomaly categories are captured using different strategies, also referred to as local experts, with different feature extraction schemes and advanced learning methods. The outputs of these experts are then fused by appropriate voting techniques. In addition to this framework, we also introduce a highly performing ML algorithm that combines a light-weight Radial Basis Function Neural Network and an Ensemble Learning technique. This algorithm is compared against other learning methods for the purpose of local expert creation. This work falls well under the category of bias-variance-computations tradeoff problem. In general, we wish to reduce bias (for higher accuracy), variance (for fewer false alarms) and computations (for fast real time response). An extensive empirical analysis conducted on the Knowledge Discovery and Data Mining (KDD-99) intrusion detection data suggests that the proposed framework obtains noticeable performance improvement compared with other state-of-the-art techniques, in terms of detection accuracy, system robustness and total cost.

This chapter starts with an overview of network intrusion detection technology and the related works of ML approaches for Network Security domain, followed by a study of the

Radial Basis Function Neural Network (RBKNN) family which has been reported for great successes in many applications. Emphasis is put on the Generalized Regression Neural Network (GRNN) and Vector-Quantized GRNN (VQ-GRNN) for their typical learning and system robustness properties. We also provide an overview of Ensemble Learning methods in which multiple classifiers are trained to solve the same problem and their decisions are then aggregated in some manner. Such methods can be used to boost predictive performance of some learning algorithms. Next, the Multiple-Expert Classification Framework (MECF) with implementation of advanced voting techniques is presented. The usefulness of this model is then illustrated through its application to the network intrusion detection problem, focusing on detection capability on rare attack categories. Finally, this chapter is concluded with future research direction discussed.

2. Intrusion Detection and Machine Learning techniques

2.1. Intrusion Detection Systems

As more and more corporations rely on computers and networks for communications and critical business transactions, securing digital information has become one of the largest concerns of the business community. A powerful security system is not only a requirement but essential to the livelihood of enterprises (Kemmerer & Vigna, 2002). In recent years, there has been a great deal of research conducted in this area to develop intelligent and automated security tools which can fight the latest cyber attacks. The security achieved must be reasonable yet sufficient, balancing needs for accountability with equally important needs for privacy and accessibility. Alongside the existing techniques for preventing intrusions such as encryption and firewalls, Intrusion Detection technology has established itself as an emerging research field that is concerned with detecting unauthorized access and abuse of computer systems from both internal users and external offenders. An Intrusion Detection System (IDS) is defined as a protection system that monitors computers or networks for unauthorized activities based on network traffic or system usage behaviors, thereby detecting if a system is targeted by a network attack such as a denial of service attack (McHugh, Christie, & Allen, 2000). In response to those identified adversarial transactions, IDS can inform relevant authorities to take corrective actions.

There are a large number of IDS available on the market to complement firewalls and other defense techniques. These systems are categorized into two types of IDS, namely (1) misuse-based detection in which events are compared against pre-defined patterns of known attacks and (2) anomaly-based detection which relies on detecting the activities deviating from system "normal" operations.

2.2. Application of Machine Learning to Intrusion Detection

Artificial Intelligence (AI) is the key technology in many of today's novel applications, ranging from banking systems that detect attempted credit card fraud or a robot that can sense and respond to human emotions, to software systems that can work as a human expert to offer appropriate advice when needed. These technologies would not exist without the knowledge gained from AI research. As a major part of AI, Machine Learning (ML) refers to algorithmic mechanisms that allow computers to learn from experience, examples and analogy (Mitchell, 1997). The output of this learning process is actionable knowledge that can be used to solve a specific problem. In the case of Intrusion Detection, learning

involves discovering patterns of normal behavior or intrusive behavior by analyzing the sample data of such activities. This sample data is also called a training set. It should be sufficient to represent the whole population of patterns to be discovered. The learned models can then be used to make classification on a new data instance based on its similarity to normal behavior (anomaly detection) or known attack signatures (misuse detection). Many security systems have implemented ML to achieve a generalization capability from limited training data. That means, given known intrusion signatures, a security system should be able to detect similar or new attacks.

2.2.1. Related works

One of the rule-based methods which is commonly used by early IDS is the Expert System (ES) (Bauer & Koblentz, 1988; Ilgun, 1995). In such systems, the knowledge of human experts is encoded into a set of rules. This allows more effective knowledge management than that of a human expert in terms of reproducibility, consistency and completeness in identifying activities that match the defined characteristics of misuse and attacks. However, ES suffers from low flexibility and robustness. Unlike ES, Data Mining approach derives association rules and frequent episodes from available sample data, not from human experts. It utilizes statistical techniques to discover subtle relationships between data items, and from that, constructs predictive models. Using the derived rules, Lee et. al. developed a data mining framework for the purpose of intrusion detection (W. Lee, Stolfo, & Mok, 1999a, 1999b). In particular, system usage behaviors are recorded and analyzed to generate rules which can recognize misuse attacks. The drawback of such frameworks is that they tend to produce a large number of rules and thereby, increase the complexity of the system.

Decision Trees are one of the most commonly used supervised learning algorithms in IDS (Amor, Benferhat, & Elouedi, 2004; J.-H. Lee, Lee, Sohn, Ryu, & Chung, 2008; Levin, 2000a; V. Miheev, Vopilov, & Shabalin, 2000; Pfahringer, 2000a) due to its simplicity, high detection accuracy and fast adaptation. Another highly performing method is Artificial Neural Networks (ANN) which can model both linear and non-linear patterns. The resulting model can generate a probability estimate of whether given data matches the characteristics that it has been trained to recognize. Cannady (1998) developed a network-based detection system in which 9-packet-level network data was retrieved from the RealSecure database and then classified by a feed-forward neural network (Cannady, 1998). Though this prototype is not a complete IDS, the results clearly demonstrate the potential of an ANN in detecting network attacks. Latter ANN-based IDS (Mukkamala, 2002; Zhang, Li, Manikopoulos, Jorgenson, & Ucles, 2001) have reportedly achieved great successes in detecting difficult attacks. For unsupervised intrusion detection, data clustering methods can be applied (Shah, Undercoffer, & Joshi, 2003). These methods involve computing a distance between numeric features and therefore they cannot easily deal with symbolic attributes, resulting in inaccuracy. Another well-known ML technique used in IDS is Naïve Bayes classifiers (Amor et al., 2004). Because Naïve Bayes assumes the conditional independence of data features, which is often not the case for intrusion detection, correlated features may degrade its performance. In (Kruegel, Mutz, Robertson, & Valeur, 2003), the authors apply a Bayesian network for IDS. The network appears to be attack specific and its size grows rapidly as the number of features and attack types increase. Beside popular decision trees and ANN, Support Vector Machines (SVMs) are also a good candidate for intrusion detection systems (Ambwani, 2003) which can provide real-time detection capability, deal with large

dimensionality of data. SVMs plot the training vectors in high dimensional feature space through nonlinear mapping and labeling each vector by its class. The data is then classified by determining a set of support vectors, which are members of the set of training inputs that outline a hyperplane in the feature space. SVMs are scalable as they are relatively insensitive to the number of data points (Ambwani, 2003).

2.2.2. The Knowledge Discovery and Data Mining Benchmark

Current security systems are facing two fundamental challenges. First, the unbalanced nature of security dataset indicates dramatic changes in the distribution of classes compared with the normal trends i.e., some classes dominate others with their overwhelming occurrences (Kemmerer & Vigna, 2002). This will bias the resultant predictive models to favor the dominant classes. Second, increased dimensionality, especially when noise is involved, can degrade learning significantly (Kemmerer & Vigna, 2002). Together, these two characteristics make the detecting intrusive activities very challenging.

In order to facilitate the comparison of advanced research in the area of network security, the Lincoln Laboratory at the Massachusetts Institute of Technology (MIT) conducted the 1998 and 1999 evaluations of intrusion detection (McHugh et al., 2000). Funded by The Defense Advanced Research Projects Agency (DARPA), the purpose of the evaluation program is to provide a basis for making comparisons of existing IDS under a common set of circumstances and assumptions. Data obtained from these programs were then used as the benchmark training and test data sets for "Classifier Learning Contest" organized in conjunction with the 5th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining in 1999 (KDD-99).

The KDD-99 dataset has simulated the two challenging problems discussed earlier, namely, the unbalanced nature and high dimensionality of security data. It contains 7 weeks of training traffic data and 2 weeks of testing data (McHugh et al., 2000). Preprocessing was applied to abstract and summarize the raw tcpdump data to form network connections.

a) Attack types and categories

Each connection record in the KDD-99 dataset is labeled as either normal or one type of attack. There are totally 39 types of attacks which are grouped into 4 major categories (McHugh et al., 2000): Probe, Denial of Service (DoS), User to Root (U2R) and Remote to Local (R2L). In particular, Probe attacks refer to the incidents in which some malicious programs can automatically scan a network of computers to gather sensitive information or search for security vulnerabilities while a DoS attack prevents normal use of network resources for legitimate purposes by consuming the bandwidth or overloading the computational resources of the victim system. The R2L attacks occur when an intruder who has no valid account on a machine can exploit some system vulnerabilities to gain local access as a legitimate user by sending packets over a network. In contrast, U2R attacks assume that the attacker has already access to a system as a normal user and he can exploit some security holes to gain user root privileges.

b) Features

41 features were used to summarize the connection information. These features are grouped as basic features and additional features respectively (McHugh et al., 2000).

Basic Features

Bro is used as the network analyzer to derive the 9 basic features from packet headers without inspecting the packet contents (Kemmerer & Vigna, 2002). Some examples of basic features include duration of connection, protocol types and service types.

Additional Features

- *Content features:* The payload of TCP packets is assessed by applying the domain knowledge. Examples of content-based features include the number of unsuccessful logins and whether the root access was gained or not.
 - *Time based features:* It is important to inspect the packets within some time interval to cope with the temporal nature of network attacks. These features are designed to capture properties within a 2 second temporal window. Number of connections to the same host is an example of time-based features.
 - *Host based features:* Utilize a historical window estimated over the number of connections (100 connections in KDD-99) instead of time. Host based features are therefore used to assess attacks which span over intervals longer than 2 seconds.
- c) Learning methods that use KDD-99 dataset

Among intrusion detection models tested on KDD-99 dataset, most of them are reported to provide unacceptably low detection capability for U2R and R2L attacks. Some typical examples of such models include a rule-based predictive model (PNrule) (Agarwal, 2000) which is studied to effectively detect DoS and Probe attacks; the winning entry of KDD99 contest (Pfahringer, 2000b) which is composed from 50×10 C5 decision trees fused by cost-sensitive bagged boosting. Similar techniques are also developed such as a decision tree forest constructed by Kernel Miner (KM) tool (Levin, 2000b) and two layers of voting decision trees augmented with human security expertise (V. Miheev, Vopilov, A. Shabalin, I., 2000). Due to poor performance of these approaches on some sophisticated attacks, we are motivated to develop a new learning method to improve the overall detection performance on KDD 99 benchmark.

3. Artificial Neural Network and Ensemble Learning

Unlike other pattern recognition tasks which may sacrifice accuracy for system robustness and stability, Intrusion Detection requires very high accuracy which implies both high detection rate and low false alarm rate (Sommer, 2008). An undetected intrusion can cause serious damage to computer networks. In this regards, high detection accuracy is of great importance for new security systems. In addition to accuracy, security systems must be also fast enough not to cause bottlenecks in communication networks. That is, network administrators should be alerted that their systems have been penetrated or have been used as springboards for attacks on other systems right after the incidences have occurred. In general, security system with high accuracy requires heavy computations. In our approach, we develop a system that achieves high accuracy for real time IDS but requires relatively small computational complexity. This ensures that the systems can both perform accurately and respond to incidences in a timely fashion.

3.1. Bias-Variance Dilemma

Though several ML techniques have been adopted in the Network Security domain with certain success, there remain performance limitations including low detection accuracy and

high false alarm rates, especially for rare and complicated attacks. For instance, the winning entries of KDD-99 competition do not provide satisfactory performance on U2R and R2L attack categories due to their low frequency and complicated nature. Several learning methods have been developed to increase the detection capability including ANN models. As a flexible "model-free" learning method, ANN can fit training data very well and thus provide a low learning bias. However, they are also susceptible to the overfitting problem, which can cause instability in generalization (Mitchell, 1997). This degraded performance is the consequence of the overfitting or overtraining problem, in which data sensitivity causes the resulting classifier to have small bias but large variance.

The learning bias is defined as the measure of how accurately the model fits the available sample data while the generalization variance measures how stably the model performs for prediction or classification tasks (Mitchell, 1997). To avoid overfitting, some methods which are less dependent on available data are introduced, but they may misrepresent the true functional relationships and have a large bias. The bias and variance hence are said to be inversely related (Mitchell, 1997), i.e. with a fixed data set, reducing one will inevitably cause the other to increase.

Some approaches are proposed to improve the generalization stability by reducing generalization variance at the cost of higher learning bias, i.e. allowing underfitting. This would deteriorate the overall performance to a certain level. In critical modeling applications, underfitting is not acceptable because a miss in detection may be very costly, i.e. causing the whole computer network compromised. Therefore, a sensible detection system which can achieve both stable generalization and accurate data learning is very much desirable. Theoretically, both bias and variance may be reduced at the same time given infinite sized models. Nevertheless, this condition is generally infeasible since the model complexity must be limited in real life. In this research, we seek a compromise solution which can retain the desirable data-fitting capacity of ANN while reducing generalization variance at a minimal computational cost. A learning algorithm is proposed by combining a radial basis function neural network with an adaptive boosting method. An overview of these relevant technologies is provided in the next 2 sections.

3.2. Overview on VQ-GRNN

A family of ANN models, RBFNN, has recently drawn great research attention due to its good generalization ability and a simple network structure that avoids unnecessary and lengthy calculations as compared to the Multilayer Feedforward Networks (MFN) (Zaknich, 2003). Considering the node characteristics and the training algorithms, RBFNN are very different from MFN. The node characteristics for MFN are usually chosen as sigmoidal functions while for RBFNN, as indicated in the name, radial basis functions are employed. A popular algorithm in RFBNN family is the Generalized Regression Neural Network (GRNN) proposed by Specht (Specht, 1991) which contains a hidden layer of radial units. Each radial unit models a Gaussian response surface which can be determined by its center point and a radius. Because these functions are nonlinear, it is enough for a single hidden layer to describe any shape of function. The output of these Gaussians is then linearly weighted to produce the desirable response. The following is the general form of GRNN:

$$\hat{y}(\underline{x}) = \frac{\sum_{n=1}^{NV} y_n f_n(\underline{x} - \underline{x}_n, \delta)}{\sum_{n=1}^{NV} f_n(\underline{x} - \underline{x}_n, \delta)} \quad (1)$$

Where

\underline{x} : Input vector (under line refers to vector)

\underline{x}_n : All other training vectors in the input space

δ : Single smoothing parameter chosen during network training

y_n : Scalar output related to \underline{x}_n

NV : Total number of training vectors

In many applications, GRNN provides high accuracy. However, it is computationally expensive as well as sensitive to the selection of variances for smoothing functions. In fact, GRNN incorporates each and every training example $\{\underline{x}_i \rightarrow y_i\}$ into its architecture, i.e. all of the training vectors needs to be processed and Gaussian function's parameters such as centers and variance will need to be computed with respect to all other surrounding vectors. In order to overcome this problem, an approximation of GRNN, namely, the Vector Quantized – Generalized Regression Neural Network (VQ-GRNN) is proposed by Zaknich (Zaknich, 1998) for application to general signal processing and pattern recognition problems. VQ-GRNN is a generalization of Probabilistic Neural Network (PNN) and is related to Generalized Regression Neural Network (GRNN) classifiers (Spath, 1991). In particular, this method approximates GRNN by quantizing the data space into clusters and associate a specific weight for each of these clusters.

If there exists a corresponding scalar output y_i for each local region (cluster) which is represented by a center vector \underline{c}_i , then a GRNN can be approximated by a VQ-GRNN formulated as follow:

$$\hat{y}(\underline{x}) = \frac{\sum_{i=0}^M Z_i y_i f_i(\underline{x} - \underline{c}_i, \delta)}{\sum_{i=0}^M Z_i f_i(\underline{x} - \underline{c}_i, \delta)} \quad (2)$$

Where

\underline{c}_i = center vector for cluster i in the input space

y_i = scalar output related to \underline{c}_i

Z_i = number of input vectors \underline{x}_j within cluster \underline{c}_i

δ = single smoothing parameter chosen during network training

M = number of unique centers \underline{c}_i

Comparing Equation 2 and Equation 1 of GRNN, we can find the only difference is that VQ-GRNN applies its computation on a smaller number of clusters of input vectors represented by centers vectors \underline{c}_i rather than working on individual input vectors \underline{x}_n . Though VQ-GRNN has minimal computational overheads and hence, more stable than GRNN, it may be less accurate in predicting attacks with low frequency of occurrence. The next section discusses a possible approach to enhance the predictive power of VQ-GRNN.

3.3. Overview on Ensemble Learning

The goal of learning algorithms is to discover the underlying functional relationship of input variables. Ordinary ML methods work by searching through a space of possible functions, called hypotheses, to find the best approximation to the unknown function. The best hypothesis can be identified based on how well it fits the training data and how consistent it is with any available prior knowledge about the problem. Ensemble learning algorithms take a different approach. Rather than finding one best learner to explain the data, they construct a set of learners, called a committee or ensemble, and then have those learners vote in some manner to predict the label of new data points. Even though the component learners within the ensemble are all attempting to solve the same problem, it is likely that each of them would have different strengths and weaknesses in different situations. Realizing and managing the situations in which the learner do not perform as well as expected is the key challenge for ensemble research (Costa, Filippi, & Pasero, 1995). A number of research (Windeatt & Roli, 2003) has supported a widespread view that for an ensemble to achieve best performance on a task, the component predictors should exhibit "diverse errors", meaning that they should have different error rates. However, in achieving this, the individual accuracy may be affected. Therefore, training an ensemble is actually a balancing act between error diversity and individual accuracy.

Due to the significant performance improvements over single classifiers, ensemble construction has become one of the most active fields of AI and has received immense research attention. In particular, ensemble algorithms iteratively run a base learning algorithm (called base learner) and then form a vote out of the resulting hypotheses (Schapire, 1999). There are two main approaches to producing these component hypotheses. The first approach, namely bagging, is to construct each hypothesis independently in such a way that the resulting set of hypotheses is accurate and diverse, that is, each individual hypothesis has reasonably low error rate for making new predictions and yet the hypotheses disagree with each other in many of their predictions. It is empirically shown that an ensemble of those hypotheses is more accurate than any of its component classifiers, because their disagreements will "cancel out" when the ensemble comes to the joint classification stage (Optiz & Maclin, 1999).

Unlike bagging, which relies on resampling the training dataset randomly with a uniform probability distribution, boosting (Schapire, 1999) guides changes of the training data to direct further classifiers toward more "difficult cases". This method is a stepwise technique that combines learners in such a way that the composite - boosted learner - outperforms the single learner. Amongst popular boosting variants, Adaptive Boosting or AdaBoost is the most widely adopted method which allows the designer to continue adding weak learners until some desired low training error has been achieved (weak learners have accuracy only slightly better than chance whereas weak hypotheses are generated based on the performance of previous ones). AdaBoost is "adaptive" in the sense that it does not require prior knowledge of the accuracy of these hypotheses (Schapire, 1999). Instead, it measures the accuracy of a base hypothesis at each iteration and sets its parameters accordingly.

Without loss of generality, let us consider the standard two-class supervised ML problem: given a set of N independent and identically distributed (i.i.d) training examples (x_n, y_n) , $n = 1, \dots, N$, with $x_n \in X$ and $y_n \in Y := \{-1, +1\}$, we would like to learn a function $f: X \rightarrow Y$ that is able to generalize well on unseen data generated from the same distribution as the training data. To obtain such a function, the boosting algorithm iteratively trains a weak

hypothesis on a weighted data sample. As boosting progresses, training examples that are hard to predict correctly, get incrementally higher weights than the other examples. The intended effect is to force the weak learner to concentrate on examples and labels that will be most beneficial to the overall goal of finding a highly accurate classification rule. This update process is repeated, until a certain stopping condition is met (e.g a given number of weak classifiers are trained or the learning error reaches a desirable level). The final joint classification is the linear weighted combination of the base hypotheses (Huang, Ertekin, Song, Zha, & Giles, 2007):

$$f_{\alpha}(x) = \text{sign}[\sum_{t=1}^M \alpha_i h_i(x)]$$

Motivated by the need of an accurate detection system for network security applications, we seek a learning algorithm which provides a good tradeoff for learning bias, generalization variance and computational requirement. In theory, the GRNN can achieve the optimal Bayesian estimate (with infinity network size) but with a cost of extremely demanding computation resource. The VQ-GRNN reduces the computationally extensive nonparametric GRNN to a semiparametric neural network by applying vector quantization techniques on the input space. This reduction significantly improves the robustness of the algorithm (low variance), but also affects its learning accuracy to some extent. To overcome this limitation, AdaBoost is used to boost its performance. The boosted version of VQ-GRNN which is referred to as Boosted VQ-GRNN will be implemented in the Multi-Expert Classification Framework.

4. Multi-Expert Classification Framework

Different learning algorithms behave variably on different classes. They may obtain superior performance on some classes but present unacceptable low accuracy for others. The imbalance of predictive performance motivates this research to construct an intelligent multi-expert learning framework which can aggregate expert knowledge from class-specific models, i.e. classifiers specialized in detecting a specific class. There is a good deal of research that shows the potentials of models that combines classification results from individual sub-models. Basically, there are two forms of classifier combination, the multi-stage (or hierarchical) (Vuurpijl, 2000) methods and the ensemble (or late fusion) (Kuncheva, 2002.) methods. In the first approach, the classifiers are placed in a multi-layered architecture where the output of one layer affects the model selection in the next layer. On the other hand, the second approach explores ensembles of classifiers, trained on different distributions of the original dataset and using different or similar features and learning algorithms. The outputs of these classifiers are then fused into one compound classification using voting techniques.

For a multi-class classification problem such as network intrusion detection, instead of trying to design a learning algorithm that is accurate over the entire space, we can focus on creating a model that can predict well for a specific portion within the space. We then combine such models to obtain a joint classifier which performs accurately on many classes. Under this light, a Multi-Expert Classification Framework (MECF) combining different classifiers for different types of attacks is proposed. Its sub-models are trained in an attack-specific manner and then integrated to accumulate their specializations. Boosted VQ-GRNN will be compared with several algorithms and then used for creating component classifiers.

4.1. Framework description

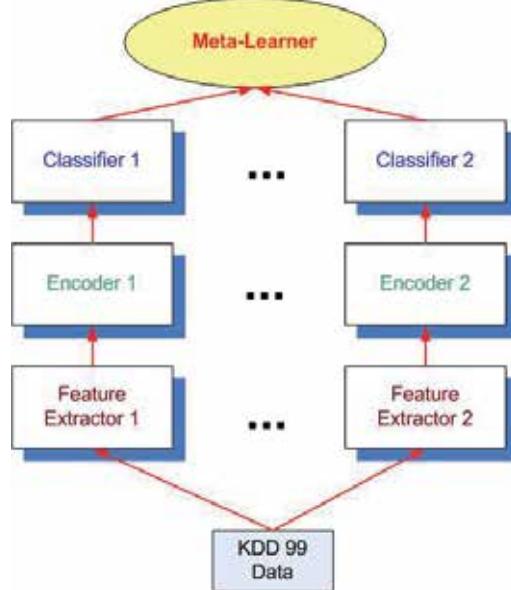


Fig.1. Multi-expert classifier for Intrusion Detection

Fig.1 describes a generic predictive model which combines different classifiers; each with special expertise in detecting a specific attack type. Each of these classifiers will be trained on different subsets of an underlying universal dataset. These subsets differ from each other in terms of attribute selections (Feature selectors) and attack-specific encoding schemes (Encoder). We aim to construct the class-specific classifiers, called experts, which have high Detection Rate on specific classes. To do so, several combinations of different attributes will be tested to gain the best performance for a particular attack category. We then train those classifiers on the dataset whose labels indicate whether a data instance belongs to a particular attack category or not. For example, if a classifier is created to recognize Probe attacks (Probe expert), then the data labels will be encoded as Probe for instances belonging to probe category or Non-probe otherwise. This has an effect of reducing a multi-class learning problem into a multiple binary classifications. The learning speed will be faster and the resulting classifier will be more “specialized” in detecting particular categories of attacks and less prone to overfitting problem.

Another useful aspect of this approach originates from the fact that even when different classifiers are trained on the same dataset and have comparable performance on the test set, they still have different “inductive biases” (Mitchell, 1997). This prevents these models from generalizing in identical ways. Under the proposed arrangement, component classifiers are very different from each other in terms of their biases. From experiments, it is shown that if a classifier is trained with a dataset which emphasizes a particular attack, it will have good detection rate for that particular attack but does not detect other attacks well. One of the widely used approaches is the cross-validation which perceives the different “inductive biases” as an indication to select “super” classifiers which perform best on all classes. As a result, some models will be discarded because of their low performance. This leads to a potential loss of useful information and effort. In contrast, an ensemble can effectively make

use of such complementary information to reduce model variance and bias (Tumer & Ghosh, 1995). The meta-learner in this framework could be as simple as a lookup table to a more advanced voting techniques. A brief review of related voting methods is given in the next section.

4.2. Voting techniques for pattern recognition

In human society, voting is a common concept in which voters indicate their preference choices from multiple options (candidates) by means of a vote (Parhami, 1994). These votes are then integrated into one final decision (the winner). This process is called an election. In the context of classifier combination, the voters are the individual classifiers that can generate a single class or a ranked list of all classes as a vote; the possible classes are the candidates and an election is the classification of one sample. The winner is the candidate that is chosen as result of the classification procedure of the sample by the combination of classifiers. There are a number of families of voting techniques.

Firstly, the un-weighted voting methods consider each vote equally and the only differentiation between the candidates is the number of votes they have received. As a consequence, voters cannot express the degree of preference of one candidate over another (Parhami, 1994). Apart from this limitation, un-weighted voting such as majority voting is still commonly used, due to its simplicity and relatively good performance. Particularly, every voter has one vote that can be cast for any one candidate and the candidate that obtained the majority of the votes will win the election.

The second family of voting methods is confidence voting in which voters can express the degree of their preference for a candidate by assigning a confidence value to candidates. The higher the total confidence value a candidate received, the more it is preferred by the voter. In our experiments, confidence value is equivalent to probabilities of class membership that are generated by local experts. There are 3 common ways of computing the total confidence votes: (1) summing up all confidence values (Sum rule); (2) multiplying all confidence values (Product rule); (3) repeatedly applying a majority vote based on the highest ranked candidate of each voter's preference ranking and transferring votes between candidates (Single transferable vote-STV) (Doron, 1977). The basic principle of STV is that voters rank the candidates in order of preference. In order to be elected, a candidate must achieve a computed quota. The votes can be transferred in two cases:

- Excess votes over the quota are appropriately down-weighted and allocated to the next preference of voters (this is not applicable in our case because we terminate voting when a winner is selected).
- If no candidate reaches the quota, the candidate with the least number of votes is eliminated and their votes transferred to next preferences.

In the context of classifier combination, voting techniques like STV are necessary because it can better integrate the preference choices of the local experts. For example, if no expert has enough confidence to classify an input vector, instead of marking it as an "unknown" instance which implies overheads for further investigation, the least voted candidate class is eliminated and its votes will be transferred to other classes. By this means, we not only utilize the votes that are otherwise wasted but also reduce the need for further processing of the unknown instances. In our experiments on the KDD-99 dataset, we attempt to use different voting methods and examine their behaviors in a multi-expert framework.

5. Experimental analysis

5.1. Cost-based Analysis

The KDD-99 dataset takes the cost sensitivity into consideration in evaluating learning methods. An error on a particular class may not be equally serious as errors on other classes. To make comparison between intrusion detection methods sensitive to cost, a cost matrix (CostM) is given for different attack categories.

Predicted \ Actual	Normal (0)	Probe (1)	DoS (2)	U2R (3)	R2L (4)
Normal (0)	0	1	2	2	2
Probe (1)	1	0	2	2	2
DoS (2)	2	1	0	2	2
U2R (3)	3	2	2	0	2
R2L (4)	4	2	2	2	0

Table 1. Cost matrix for the KDD-99 dataset (Levin, 2000a)

In this table, rows correspond to actual categories, while columns correspond to classified values. The Normal category is symbolized as class 0, Probe as 1 and so forth. According to this cost matrix, if a R2L attack is falsely classified as Normal connection, the incurred penalty cost is 4 while misclassification of a Probe attack as normal has a cost of 1. This suggests that R2L attacks are more serious than Probes.

During the testing phase, the outputs of a classifier will be generated in form of a Confusion Matrix (ConfM) which summarizes the classification results. The difference between CostM and ConfM is that an entry at row i and column j in the cost matrix, CostM(i,j), represents the cost associated with a connection which actually belongs to class i and is classified as class j while the same position in the confusion matrix, ConfM(i,j), displays the number of connections of type i and is classified (correctly or incorrectly) as class j. Given a test set, the average cost of a classifier is calculated as below (McHugh et al., 2000):

$$Cost = \frac{1}{N} \sum_{i=1}^5 \sum_{j=1}^5 ConfM(i,j) * CostM(i,j)$$

Where

N: total number of connections in the dataset

ConfM(i,j): the entry at row i, column j in the confusion matrix.

CostM(i,j): the entry at row i, column j in the cost matrix.

5.2. Experiment design

We are motivated to explore how different learning algorithms perform for different attack categories, i.e. to check whether a certain algorithm may achieve superior performance for a specific attack category. In the light of this possibility, we compare several detection models using different pattern recognition methods and select the best performing algorithms as well as the most discriminant features for each attack category. A multi-classifier system then evolves which improves the overall detection performance on the KDD-99 benchmark.

A generic ensemble model is developed as in Fig.1, containing 5 classifiers (experts) which specialize in detecting certain classes (Normal, Probe, DoS, U2R and R2L). There are 3 major phases in this model:

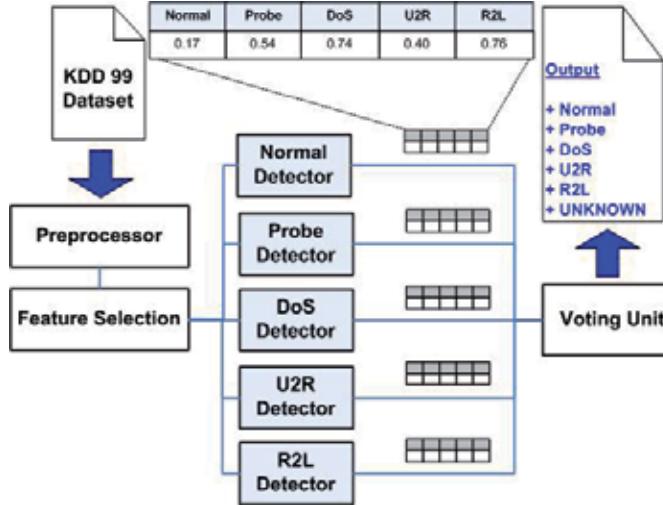


Fig. 2. Multi-expert classification framework (MECF)

5.2.1. Data preprocessing

The KDD-99 dataset contains attributes of different forms such as continuous, discrete and symbolic with varying resolutions and ranges. In order to build predictive models, this data is first converted into a compatible format. Several preprocessing techniques include data reduction (removing duplicated data and sampling data into smaller sets), data encoding and normalization (mapping k-value nominal and ordinal attributes into k integers, re-scaling numeric attributes), dimensionality reduction (extract most informative attributes by adapting Protocol-based Logistic Regression (PLR) (Yu, Wu, & Wong, 2008)).

5.2.2. Local expert creation

Local experts (detectors) are constructed by selecting the best performing learning algorithms and best discriminant features for specific attack types. For each class (normal and attack categories), a specialized classifier is created by three steps:

- Apply class-specific encoding schemes on data

For example, to produce a DoS detector, each data record is encoded to indicate whether it is actually a DoS attack or not, i.e. its label is assigned to 1 if it belongs to this class, or 0 otherwise.

Given a dataset S in which the input features are represented by vector x and the output (or target) class is denoted by label c where $c=1,\dots,K$ and K is the number of possible labeling (in KDD-99, $K=5$). To construct a local classifier which is specialized in detecting a specific class k , the label c should be recoded to y_k such that:

$$\begin{cases} y_k = 1 \text{ if } c = k \\ y_k = 0 \text{ if } c \neq k \end{cases}$$

- Select important features from the input data

We train some classifiers with different combinations of attributes in the encoded data. The combination which gives the best performance will be selected for that particular class.

c) Choose the best performing classifier

The learning algorithm with the best trade-off between high detection capability and low false alarm rate will be selected.

5.2.3. Expert Combination

Given an input vector, each local expert computes an array of probabilities (ranging from 0 to 1) of class membership for each available class. These probabilities are merged by voting methods to decide the final classification of the input vector. Several voting approaches will be implemented and their performance will be compared in the next section.

5.3. Experiment results

5.3.1. Constructing MECF

To construct local experts, different predictive models are trained with different combinations of data feature groups including basic group (B), content-based group (C) and traffic-based group (T) for basic, content-based and traffic based features respectively. In particular, the semi-parametric algorithms such as Boosted VQ-GRNN will be compared against the parametric models such as decision trees (linear discriminant), boosted trees and non-parametric methods (MLP, GRNN). The boosted tree algorithm is the combination of J48 and AdaBoostM1 methods which are available from the Weka package. For the GRNN and Boosted VQ-GRNN models, a similar model size was chosen with one hidden layer containing 15 hidden nodes. The MLP has a structure of three layers with the number of input neurons equal to the number of input features, five hidden neurons and five output neurons (1 for each class). The above numbers of hidden nodes are selected from multiple experiments (number of hidden nodes varying from 5 to 55 with steps of 10) by choosing the setting with lowest bias and variance.

Model	B	T	C	B+T	B+C	T+C
Detection rate for Normal (%)						
J48 Tree	81.1	11.4	11.2	79.9	80.2	41.2
Boosted J48	94.6	12.5	13.1	92.0	92.2	43.2
MLP	85.6	8.6	8.5	85.5	86.7	30.4
SVM	90.4	10.9	10.4	91.2	91.5	29.8
GRNN	95.1	13.5	12.5	93.8	94.6	40.2
Boosted VQ-GRNN	95.6	14.5	13.5	95.1	95.2	33.8
Detection rate for Probe (%)						
J48 Tree	46.2	20.7	22.4	86.1	56.4	39.1
Boosted J48	45.1	25.1	23.1	90.7	56.8	33.2
MLP	39.1	27.7	11.3	82.3	40.1	32.1
SVM	42.7	30.8	16.9	87.5	50.6	68.3
GRNN	44.6	25.1	21.7	84.3	47.8	44.1
Boosted VQ-GRNN	45.1	29.1	20.3	86.2	50.2	43.5

	Detection rate for DoS (%)					
J48 Tree	78.2	17.2	34.7	88.6	76.1	51.3
Boosted J48	79.0	24.1	50.1	90.3	77.3	53.7
MLP	66.7	15.7	40.1	87.6	60.1	49.1
SVM	78.2	20.2	33.5	85.6	58.3	50.3
GRNN	78.1	22.2	44.8	92.0	75.7	51.4
Boosted VQ-GRNN	78.2	23.1	43.2	92.3	78.2	50.5
	Detection rate for U2R (%)					
J48 Tree	0.4	0.0	5.0	0.3	22.3	7.6
Boosted J48	0.1	0.1	4.3	1.1	27.2	7.8
MLP	0.3	0.8	6.7	0.1	18.1	7.1
SVM	0.0	0.5	4.6	0.2	11.6	6.6
GRNN	0.1	3.2	7.2	0.1	28.6	6.0
Boosted VQ-GRNN	0.0	3.2	7.2	0.4	27.2	6.1
	Detection rate for R2L (%)					
J48 Tree	0.9	0.0	2.9	0.0	34.0	2.8
Boosted J48	0.5	0.0	2.8	0.0	35.2	2.3
MLP	0.0	0.1	2.1	0.1	22.1	3.0
SVM	0.0	0.0	1.5	0.0	33.8	2.5
GRNN	0.9	0.1	2.3	0.3	36.2	3.3
Boosted VQ-GRNN	0.1	0.1	2.1	0.3	41.2	3.1

Table 2. Detection rate for different classes and features

From the results in Table 2, for each type of attacks, the best performing combination of features and learning algorithms will be chosen and highlighted. For example, the combination of basic and traffic features with the Boosted J48 Tree achieve the highest detection rate for Probe attacks (90.7%). Note that, these models are trained on the data which is encoded for specific categories. Therefore, their detection rates are only valid on the encoded data. The Table 3 shows the best performing strategies (feature combination, attack-specific encoding scheme and learning algorithm) selected.

Model	Features used	Encoding scheme used	Algorithm used
<i>Normal Expert</i>	Basic	1 (Normal) ; 0(non- Normal)	<i>Boosted VQ-GRNN</i>
<i>Probe Expert</i>	Basic + Traffic	1 (Probe) ; 0(non-probe)	<i>Boosted J48 Tree</i>
<i>DoS Expert</i>	Basic + Traffic	1 (DoS) ; 0(non- DoS)	<i>Boosted VQ-GRNN</i>
<i>U2R Expert</i>	Basic + Content	1 (U2R) ; 0(non-U2R)	<i>GRNN</i>
<i>R2L Expert</i>	Basic + Content	1 (R2L) ; 0(non- R2L)	<i>Boosted VQ-GRNN</i>

Table 3. Local experts' configuration

5.3.2. Performance evaluation

The classification results of the constructed local experts are combined in the Multi-Expert Classification Framework (MECF) using different voting strategies including majority vote (MECF-MV), sum rule (MECF-SR), product rule (MECF-PR) and Single Transferable Vote

(MECF-STV) voting methods. These models are then compared against other existing methods, including the KDD-99 winner (Pfahringer, 2000a), the rule-based PNrule approach (Agarwal, 2000) and the Columbia Model (W. Lee & Stolfo, 2000). Results from some of these techniques may not be complete (e.g. FAR is not available or results of Normal class are not provided). The comparison between learning algorithms is presented Table 4 where for each class, the highest DR and lowest FAR are in bold and the best performing method is highlighted.

	Normal	Probe	DoS	U2R	R2L	DR/FAR (%)
KDD 99 winner (Pfahringer, 2000a)	99.5	83.3	97.1	13.2	8.4	DR
	27.0	35.2	0.1	28.6	1.2	FAR
PNrule(Agarwal, 2000)	99.5	73.2	96.9	6.6	10.7	DR
	27.0	7.5	0.05	89.5	12.0	FAR
Columbia Model (W. Lee & Stolfo, 2000)		96.7	24.3	81.8	5.9	DR
MECF–MV	99.4	88.0	97.2	29.3	11.9	DR
	30.2	40.1	2.8	14.5	20.0	FAR
MECF–SR	99.5	92.0	96.7	21.8	17.1	DR
	3.3	6.7	0.09	7.1	8.7	FAR
MECF–PR	82.1	85.3	98.0	11.4	6.8	DR
	5.2	21.4	0.56	4.3	15.7	FAR
MECF–STV	99.8	99.3	98.1	89.7	48.2	DR
	3.6	1.1	0.06	0.03	0.19	FAR

Table 4. Detection Rate (DR %) and False Alarm Rate (FAR %) comparison

Across the classes, in comparison with existing techniques, MECF using simple majority vote (MECF-MV) does not provide noticeable improvement in DR while its FAR is quite high in most cases (Probe, U2R, R2L and Normal). The product rule voting technique (in MECF-PR) is found unstable because it suddenly increases DR for the DoS attack (98.0%) while its results for the remaining classes are largely degraded. The MECF-SR, on the other hand, has a stable performance with fairly high DR and low FAR for most of the classes (it has lowest FAR for the Normal class).

Among the methods considered here, our classification framework that uses STV technique (MECF-STV) and the LCRF (Gupta, Nath, & Kotagiri, 2008) are the most recent and they seem to be the most accurate models (high DR and low FAR). Most methods do not perform well for the U2R attacks (DR is lower than 30%), except for a dramatic increase in DR is noted for Decision Tree (J.-H. Lee et al., 2008) (58.8%), Columbia Model (W. Lee & Stolfo, 2000) (81.8%), LCRF (Gupta et al., 2008) (86.30%) and MECF-STV (89.7%). For the R2L category, only MECF-STV provides a significantly high DR (48.2%) and lowest FAR (0.19%).

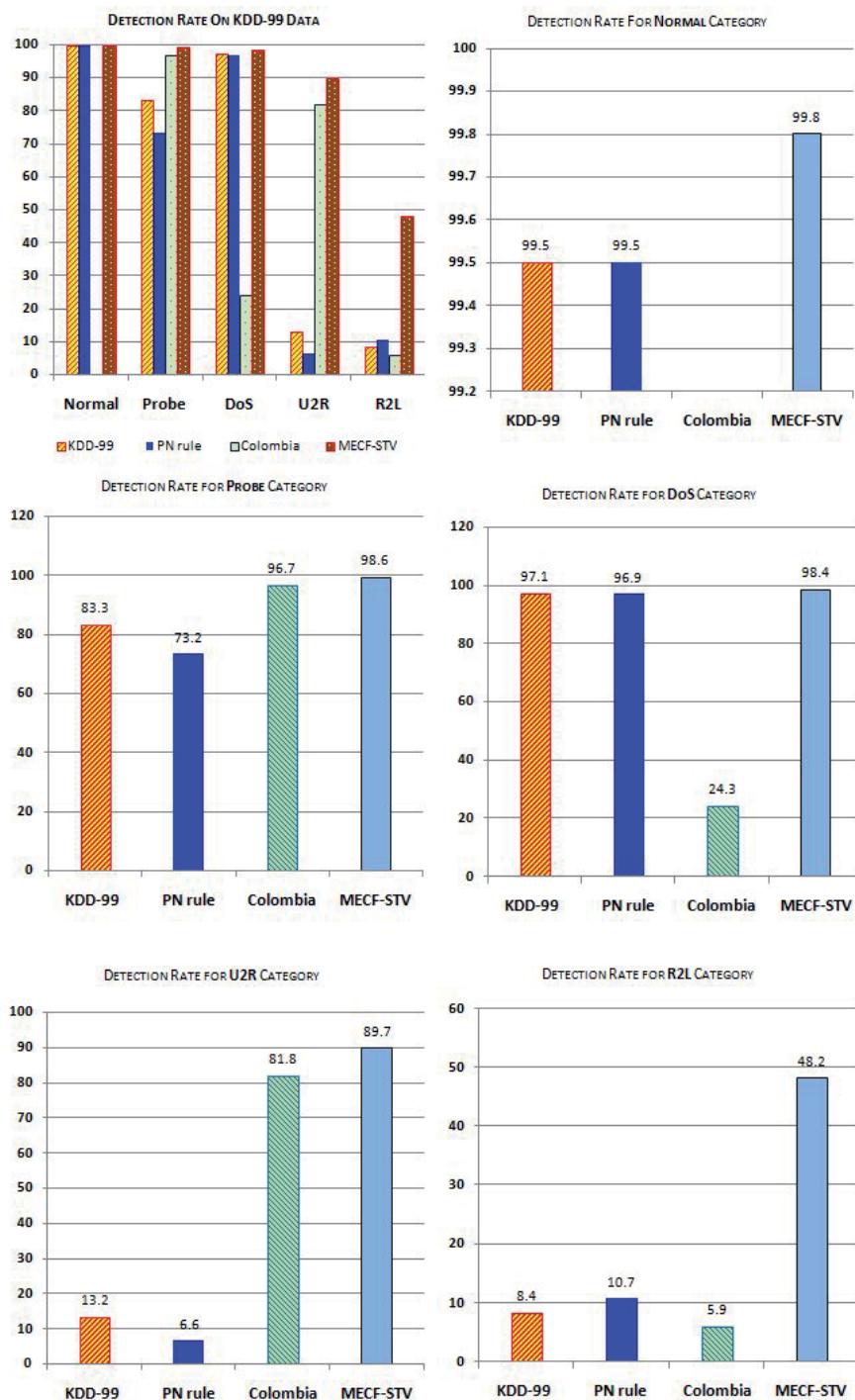


Fig. 3. Detection Rate comparision

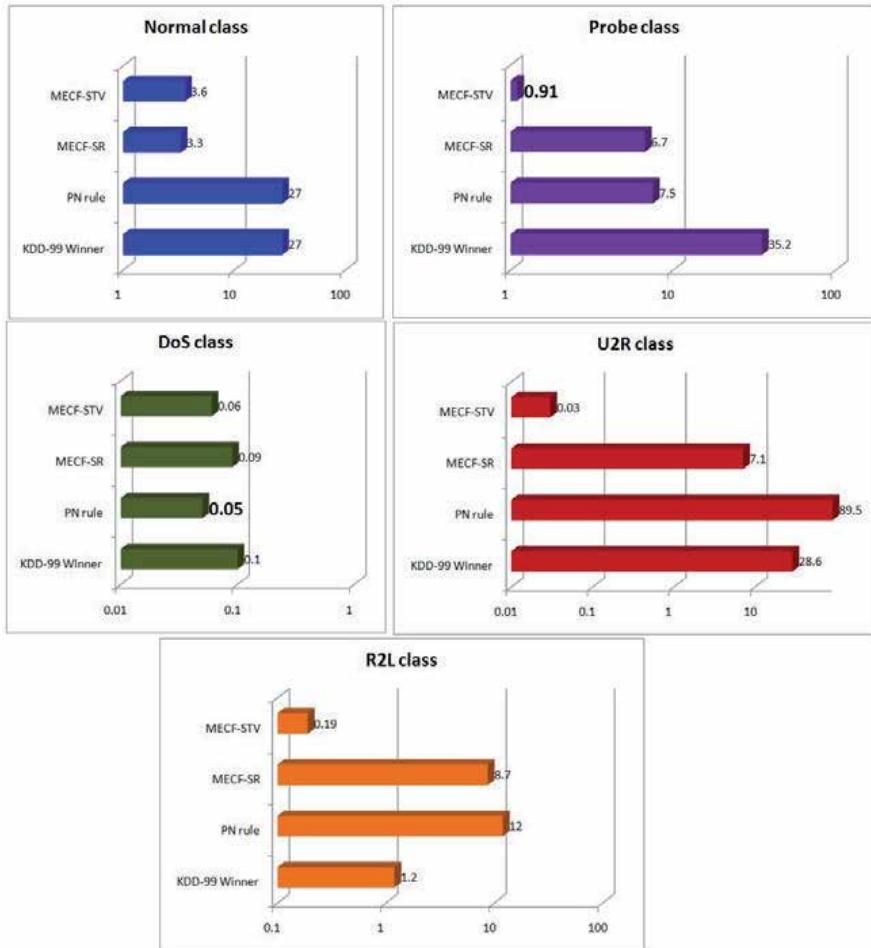


Fig. 4. False Alarm comparision

Though no model can provide both highest DR and lowest FAR for all the classes, our MECF-STV is the most promising model which makes the best combination of detection capability (DR) and system robustness (FAR). That is, MECF-STV can achieve highest DR for all the classes and its FAR for the two rare U2R and R2L categories are the lowest. In the case that other models obtain lower FAR, the performance difference between that model and MECF-STV is very small. Moreover, the average cost of MECF-STV is 0.1089 per test sample, which is much lower than the KDD winner (0.2332). It is also important to note that the test data used in our experiments follows a different distribution than in the training data and contains an additional 14 attack types not included in the training data. Therefore, achieving high DR on this test dataset suggests that our model is robust to data distribution changes and is able to detect unseen attacks. Fig. 3 and Fig. 4 visualize DR and FAR of the classifiers on the KDD-99 dataset.

In summary, our proposed MECF-STV (multi-expert classification framework using single transferable voting) can significantly reduce the total misclassification cost compared with KDD-99 winner. Its detection rates are the highest for Normal, U2R and R2L categories and

very close to that of the best performing classifiers for Probe and DoS categories. Finally, False alarm rates obtained by MECF-STV are often the lowest compared to other methods.

6. Conclusions and Future work

6.1. Conclusion remarks

Motivated by the low detection rates on rare and complicated attacks in the KDD-99 benchmark, we develop the Multi-Expert Classification Framework (MECF) in which Vector Quantized Generalized Regression Neural Network (VQ-GRNN) and Adaptive Boosting (AdaBoost) are deployed. It is shown that some learning algorithms that use certain sets of features and class-specific encoding schemes can achieve superior detection capability for a given attack category. Consequently, MECF aims to capture the characteristics of different intrusive classes and normal instances by constructing a set of five local classifiers (experts) to classify data into five different classes including Normal, Probe, DoS, U2R and R2L. For each of these classes, a tailored learning strategy (an expert) is employed. The outcomes of these experts will then be combined using high performance voting methods. Experimental results indicate that the weighted voting strategies outperform simple majority voting. Using the transferable voting approach, our Multi-Expert Classification Framework using Single Transferable Voting (MECF-STV) model can significantly improve the detection rates of not only minority and distributed U2R and R2L attacks but also majority classes compared with other techniques. Moreover, this model achieves a low detection cost.

In conclusion, the empirical analysis from this research suggests that our proposed framework performs very well in the intrusion detection problem in terms of accuracy and system robustness while offering “affordable” computation compared with existing state-of-the-art techniques. However, no system is absolutely secure given the best possible detection algorithms. That is true as long as the system is connected to other networks. The absolute security can only be achieved by disconnecting the system from the outside world which is against the principal benefits of internetworking – accessibility of information. This means that protecting our resources from network attacks is an ongoing task and computer security is always an active and challenging research area.

6.2. Future works

Increasingly-intense distributed denial-of-service (DDoS) attacks on Internet Service Provider (ISP) backbones are surpassing providers' capacity and knocking customers offline (Kemmerer & Vigna, 2002). Such attacks are more dangerous than traditional DoS due to its complex and distributed nature. To fight these attacks, the generic IDS examined here can be extended to a multi-level agent detection system for distributed networks. Literature in this area has highlighted several generic limitations associated with Distributed Intrusion Detection System (DIDS) such as inability to cope with huge amount of data in different formats and ineffective coordination between distributed sensors and agents (Kemmerer & Vigna, 2002). Some of these problems were outlined and different approaches have been implemented to solve those problems. It is interesting to explore the applicability of our framework in such distributed context, i.e. using the MECF-STV to develop a multi-level agent framework to construct a robust, distributed, error tolerant and self protecting DIDS. To design such DIDS, an appropriate design of centralization and decentralization of data processing and detection capabilities needs to be considered.

7. References

- Agarwal, R., Joshi, M. V. . (2000). PNrule: A New Framework for Learning Classifier Models in Data Mining. Paper presented at the Technical Report
- Ambwani, T. (2003). Multi class support vector machine implementation to intrusion detection. Paper presented at the Proceedings of the International Joint Conference of Neural Networks.
- Amor, N. B., Benferhat, S., & Elouedi, Z. (2004). Naive Bayes vs. Decision Trees in Intrusion Detection Systems. Proc. ACM Symp. Applied Computing, 420-424.
- Bauer, D. S., & Koblentz, M. E. (1988). NIDX - an expert system for real-time network intrusion detection. Paper presented at the Proceeding of the Computer Networking Symposium.
- Cannady, J. (1998). Artificial neural networks for misuse detection. Paper presented at the In Proceedings of the National Information Systems Security Conference.
- Costa, M., Filippi, E., & Pasero, E. (Eds.). (1995). Artificial neural network ensembles: a bayesian standpoint. : World Scientific.
- Doron, G., Kronick, R. (1977). Single transferable vote: an example of a perverse social choice function. American Journal of Political Science, 21(2), 303-311.
- Gupta, K. K., Nath, B., & Kotagiri, R. (2008). Layered Approach using Conditional Random Fields for Intrusion Detection. IEEE Transactions on Dependable and Secure Computing, 5(4).
- Huang, J., Ertekin, S., Song, Y., Zha, H., & Giles, C. L. (2007). Efficient Multiclass Boosting Classification with Active Learning. SIAM International Conference on Data Mining.
- Ilgun, K., Kemmerer, R. & Porras, P. (1995). State transition analysis: a rule-based intrusion detection approach. IEEE Transactions on Software Engineering, 181-199.
- Kemmerer, R. A., & Vigna, G. (2002). Intrusion detection: A brief history and overview. IEEE Security and Privacy.
- Kononenko, I., & Kukar, M. (2007). Machine Learning and Data Mining: Introduction to Principles and Algorithms Horwood Publishing Limited.
- Kruegel, C., Mutz, D., Robertson, W., & Valeur, F. (2003). Bayesian Event Classification for Intrusion Detection. Proc. 19th Annual Computer Security Applications Conference, 14-23.
- Kuncheva, L. I. (2002.). A theoretical study on six classifier fusion strategies. Paper presented at the IEEE Transactions on Pattern Analysis and Machine Intelligence.
- Lee, J.-H., Lee, J.-H., Sohn, S.-G., Ryu, J.-H., & Chung, T.-M. (2008). Effective Value of Decision Tree with KDD 99 Intrusion Detection Datasets for Intrusion Detection System. Paper presented at the 10th International Conference on Advanced Communication Technology.
- Lee, W., & Stolfo, S. (2000). A Framework for Constructing Features and Models for Intrusion Detection Systems. Information and System Security, 4, 227-261.
- Lee, W., Stolfo, S., & Mok, K. (1999a). A Data Mining Framework for Building Intrusion Detection Model. Proc. IEEE Symp. Security and Privacy, 120-132.
- Lee, W., Stolfo, S., & Mok, K. (1999b). Mining Audit Data to Build Intrusion Detection Models. Proc. Fourth International Conference Knowledge Discovery and Data Mining 66-72.

- Levin, I. (2000a). KDD-99 Classifier Learning Contest: LLSoft's Results Overview. SIGKDD Explorations, 1, 67-75.
- Levin, I. (2000b). KDD-99 Classifier Learning Contest: LLSoft's Results Overview Paper presented at the SIGKDD Explorations.
- McHugh, J., Christie, A., & Allen, J. (2000). Defending Yourself: The Role of Intrusion Detection Systems. Software, IEEE, 17(5), 42-51.
- Miheev, V., Vopilov, A., & Shabalin, I. (2000). The MP13 Approach to the KDD'99 Classifier Learning Contest. SIGKDD Explorations, 1, 76-77.
- Miheev, V., Vopilov, A. Shabalin, I. (2000). The MP13 Approach to the KDD'99 Classifier Learning Contest. Paper presented at the SIGKDD Explorations.
- Mitchell, T. (1997). Machine Learning. New York: McGraw-Hill.
- Mukkamala, S., Janoski, G., & Sung , A. (2002). Intrusion detection using neural networks and support vector machines. Paper presented at the International Joint Conference on Neural Networks (IJCNN).
- Optiz, D., & Maclin, R. (1999). Popular ensemble methods: An empirical study. Journal of Artificial Research, 11, 169-198.
- Parhami, B. (1994). Voting Algorithms. Paper presented at the IEEE Transactions of Reliability
- Pfahringer, B. (2000a). Winning the KDD99 Classification Cup: Bagged Boosting. SIGKDD Explorations, 1, 65-66.
- Pfahringer, B. (2000b). Winning the KDD99 Classification Cup: Bagged Boosting. Paper presented at the SIGKDD Explorations.
- Schapire, R. E. (1999). A brief introduction to boosting. Paper presented at the Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, San Francisco, CA.
- Shah, H., Undercoffer, J., & Joshi, A. (2003). Fuzzy Clustering for Intrusion Detection. Proc. 12th IEEE International Conference Fuzzy Systems (FUZZ-IEEE '03), 2, 1274-1278.
- Sommer, R. (2008). Viable Network Intrusion Detection: Trade-Offs in High-Performance Environments: VDM Verlag.
- Spetch, D. F. (1991). A general regression neural network. IEEE Transactions on Neural Networks, 2(6), 568-576.
- Tumer, K., & Ghosh, J. (1995). Order statistics combiners for neural classifiers. World Congress on Neural Networks, 1, 31-34.
- Vuurpijl, L., Schomaker, L. (2000). Two-stage character classification: a combined approach of clustering and support vector classifiers. Paper presented at the In International Workshop on Frontiers in Handwriting Recognition (IWFHR).
- Windleatt, T., & Roli, F. (Eds.). (2003). Multiple Classifier Systems, 4th International Workshop, MCS 2003. Guilford, UK: Springer.
- Yu, K.-M., Wu, M.-F., & Wong, W.-T. (2008). Protocol-based classification for intrusion detection. Paper presented at the Proceedings of the 7th WSEAS International Conference on Applied Computer and Applied Computational Science.
- Zaknich, A. (1998). Introduction to the modified probabilistic neural network for general signal processing applications. IEEE Transactions on Signal Processing, 46, 1980-1990.
- Zaknich, A. (2003). Neural Networks for Intelligent Signal Processing. Sydney: World Scientific Publishing.

Zhang, Z., Li, J., Manikopoulos, C. N., Jorgenson, J., & Ucles, J. (2001). HIDE: A Hierarchical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification. Proc. IEEE Workshop Information Assurance and Security, 85-90.

Artificial Immune Network: Classification on Heterogeneous Data

Mazidah Puteh¹, Abdul Razak Hamdan², Khairuddin Omar²
and Mohd Tajul Hasnan Mohd Tajuddin¹

¹*Universiti Teknologi MARA*

²*Universiti Kebangsaan Malaysia
Malaysia*

1. Introduction

Classification is one of the important tasks in data mining that can extract knowledge from real world data sets. It helps in forecasting the future knowledge from the available knowledge or information. It also helps people in making better decision in the future based on the history and existing knowledge. With the classification algorithm, people can repeatedly make a forecast on the accumulated knowledge in new situations.

2. Immune System

2.1 Natural Immune System

A biological immune system has two broad response systems. One is innate immunity, which is general and exists in our body since we are born. The other one is an adaptive immunity that is based on two kinds of antibody cells in the body: T-cells, so named because they originate in the thymus gland and B-cells originate in bone marrow (de Castro & Timmis, 2002). When a pathogen invades the body, special cells called antigen are available. An individual T-cell or B-cell responds to the antigens by cloning and mutating to match the antigen. This is the concept of clonal selection theory (Burnet, 1959) where the binding of antibody with the antigen will activate the antibody and the clonal expansion of the antibody occurs. The closer the match, the affinity of that T-cell or B-cell from the antigen (Hunt & Cook, 1996) becomes stronger. B-cells that do not match any antigens will be eliminated. From immune network theory, (Jerne, 1974) antibody also interacts with the neighbour antibodies to form a network. If the antibody do not stimulate with the neighbour antibodies, it eventually die. After the process of generating antibodies and combating the antigens and a body has successfully defended against a pathogen, a comparatively small number of memory cells remain in the body for very long time. These memory cells recognize antigens similar to those that originally cause the immune response, so that the body's response to a future and very similar invader is much faster and powerful than to a never-before-seen invader.

2.2 Artificial Immune System

An artificial immune system is a bio-inspired computational model that uses idea and concepts from the natural immune system. Although there are about four concepts that are explored in the immune system (de Castro & Timmis, 2002), the concepts that are discussed in this paper are the interaction between antigen and B-cells (stimulation and suppression) as in the clonal selection theory (Burnet, 1959) and also the interaction between antibody and antibody as in the immune network theory (Jerne, 1974). Both theories involve cloning and mutating process (de Castro & Von Zuben, 2000). It can offer strong and robust information processing capabilities for solving complex problems. Applications of AIS include supervised and unsupervised machine learning, pattern recognition, intrusion detection and security (Dasgupta, 2006). Among the early models on supervised machine learning is Immunos81 (Carter, 2000) and AIRS (Watkins, 2001; Watkins et al., 2004). However, the former model uses significantly different and complex approach. The later model is the first straightforward immune-inspired supervised learning algorithm and has subsequently gone through a period of study and refinements (Watkins & Timmis, 2002; 2004; Hamaker & Boggess, 2004). However, many of these studied classification models concentrate on the population-based or clonal selection algorithm and ignore the important network feature (Timmis, 2001) of the immune system. The models also require numerical representation of data and mostly are tested only on numerical dataset. Some of the applications on classification with AIS models are summarized in Table 1.

Concept	Objective	References
Immune Network	DNA Classification, Text Classification	Hunt & Cook, 1996; Secker et al, 2004
Clonal Selection	Numeric Data Classification	Carter 2000; Leandro 2000; Sahan et al 2005; Leung et al, 2006; Peng et al 2007
Clonal Selection with resource limited	Numeric Data classification, Text Classification, Heterogeneous data classification	Watkins 2001;2002;2004; Hamaker 2004; Secker 2007; Puteh et al 2008
Clonal Selection with resource limited and parallel	Numeric data classification	Watkins 2004
Negative Selection	Binary classification	Igawa et al 2005

Table 1. Classification applications with AIS models

As suggested in (Watkins, 2001; Freitas & Timmis, 2007; Hart & Timmis, 2008; Timmis, 2006), methods of using other types of data need to be explored to allow for greater applicability of this learning paradigm. (Hamaker & Boggess, 2004) has explored variety of similarity measurements in generating classifiers with clonal selection concept or population-based AIS algorithm but a more comprehensive experiment on many problems with heterogeneous types is required in order to prove a high quality classification technique for heterogeneous data types. (Puteh et al., 2008) has introduced a classification

model using clonal selection for heterogeneous data that is called Flexible Artificial Immune Recognition System (FAIRS) to experiment the heterogeneous data in its original types. FAIRS has shown some improvement in the accuracy compared to the existing AIS classification models. To further experiment on AIS algorithm and to overcome the limitation mentioned in the previous research, there is a need for developing the AIS classifier with the network feature and be able to accept heterogeneous data without the need for the data transformation. In order to accept various types of data, all processes involving these data must consider appropriate and suitable affinity measurement, mutation method and the correct data structure implementation.

2.2 Distance Metrics

There are many learning systems depend on good distance function to be successful such as the nearest neighbour techniques (Cover & Hart, 1967; Hart, 1968; Dasarathy & Belur, 1991), and memory-based reasoning methods (Stanfill & Waltz, 1986). Such algorithms have had much success on a wide variety of applications (real-world classification tasks). Many of these metrics work well for numerical attributes but do not appropriately handle nominal attributes (Wilson & Martinez, 1997). The common distance metrics that are used for numerical attributes and binary attributes are the Euclidean metric and the Hamming metrics as shown in equation 1 and 2.

$$\text{Euclidean}(x,y) = \sqrt{\sum_{a=1}^m (x_a - y_a)^2} \quad (1)$$

$$\text{Hamming}(x,y) = \sum_{i=1}^L \delta_i \text{ where } \delta_i = \begin{cases} 1 & \text{if } x_i \neq y_i \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The value difference Metric (VDM) (Stanfill & Waltz, 1986) was introduced to define an appropriate distance function for nominal attributes as shown in equation 3.

$$vdm_a(x,y) = \sum_{c=1}^C \left| \frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right|^q \quad (3)$$

where $N_{a,x}$ is the number of training records in T that has the value x for an attribute a; $N_{a,x,c}$ is the number of records in T that has the value x for attribute a and class c; C is the number of classes in the problem domain; q is a constant, usually value 1 or 2.

This distance metric work well in many nominal domains, but they do not handle continuous attributes directly. Instead, they rely upon process of discretization which can degrade generalization accuracy (Ventura et al., 1995). Many real-world applications have both nominal and numeric attribute as shown in the UCI MLR (Merz & Murphy, 1998). The distance function that is used in the proposed model is Heterogeneous Value Difference Metric (HVDM). It can take heterogeneous data where it uses normalized VDM for nominal data and normalized difference for linear data. HVDM has shown a good potential to be the distance metric for heterogeneous data without the need for any transformation of data into any specific type. HVDM has become the choice for the algorithm in this research. The discussion of the distance metrics can be found in (Wilson & Martinez, 1997). As mentioned in the previous section, the Euclidean distance function is inappropriate for nominal

attributes, and VDM is inappropriate for continuous attribute, so neither is sufficient on its own for use on a heterogeneous application, i.e. one with both nominal and continuous attributes. So, HVDM is used as shown in equation 4,5,6,7.

$$\text{HVDM}(x, y) = \sqrt{\sum_{a=1}^m d_a^2(x_a, y_a)} \quad (4)$$

where m is the number of attributes. The function $d_a(x, y)$ returns a distance between the two values x and y for attribute a and it is defined as:

$$d_a(x, y) = \begin{cases} 1, & \text{if } x \text{ or } y \text{ is unknown; otherwise} \\ \text{normalized_vdm}_a(x, y), & \text{if } a \text{ is nominal} \\ \text{normalized_diff}_a(x, y), & \text{if } a \text{ is linear} \end{cases} \quad (5)$$

where normalized vdm and normalized diff are defined as follows:

$$\text{normalized_vdm}_a(x, y) = \sqrt{\sum_{c=1}^C \left| \frac{N_{a,x,c} - N_{q,y,c}}{N_{a,x} - N_{a,y}} \right|^2} \quad (6)$$

and

$$\text{normalized_diff}_a(x, y) = \frac{|x - y|}{4\sigma_a} \quad (7)$$

where x and y are 2 input vectors for attribute a and σ is a standard deviation value for a.

Distances are often normalized by dividing the distance for each variable by the range of that attribute, so that the distance for each input variable is in the range 0..1 and this is employed by algorithm in (Hamaker & Boggess, 2004). However, dividing by the range allows outliers (extreme values) to have a profound effect on the contribution of an attribute. A more robust alternative in the presence of outliers is to divide the values by the standard deviation to reduce the effect of extreme values on the typical cases. The situation for HVDM is more complicated because the nominal and numeric distance values come from different types of measurements: numeric distances are computed from the differences between two linear values, normalized by standard deviation, while nominal attributes are computed from a sum of C differences of probability values (where C is the number of output classes). It is therefore necessary to find a way to scale these two different kinds of measurements into approximately the same range to give each variable a similar influence on the overall distance measurement (Wilson & Martinez, 1997).

3. Proposed Algorithm (FINERS)

In the real world situation, there are many data set comprise both numerical and nominal data types. This paper investigates the use of HVDM distance metric for heterogeneous datasets that are composed of nominal, discrete or continuous data types or the combination of them without the need for the transformation of the data into any specific type. The algorithm in the proposed model considers an appropriate data structures to suit the complexity of recognizing heterogeneous data in its original types.

The FINERS algorithm works as follows:

1st stage:

- Calculate Affinity Threshold (AT) by calculating average affinity (distance) between all pairs among antigens
- MemoryCell (MC) initialization, usually starts with null

For each antigen do

2nd stage:

- Search for mcmatch from MC, if unavailable, antigen as mcmatch
- Clone and mutate mcmatch
- Generate first generation antibodies (AB)
- Create a network among antibodies with affinity greater than network affinity threshold (NAT)

3rd stage:

- Clone and mutate antibody from AB randomly until average stimulation is greater than stimulation threshold.
- Generate the final AB
- Create a network among antibodies with affinity greater than network affinity threshold (NAT)

4th stage:

- Search for mccandidate (most stimulated) from AB
- Compare mccandidate to mcmatch, if mccandidate is more stimulated, it is added to MC. If affinity between mccandidate and mcmatch is less than affinity threshold scalar times affinity threshold then mccandidate replaces mcmatch inside MC
- Create a network among antibodies with affinity greater than network affinity threshold (NAT)

Basically, FINERS is a one shot algorithm where each antigen is processed only in one generation. At the end of the algorithm, set of rules or classifier and output of accuracy and number of rules are generated.

4. Experiments and Discussions

Experiment on FINERS is carried out on 8 datasets from UCI MLR (Merz & Murphy, 1998). The datasets are carefully selected to represent heterogeneous data types and non-heterogeneous data types. The heterogeneous data sets are Australian Credit (CRX), German Credit (GC), Hepatitis (HP), Cleveland Heart Disease (HD) and Ljubljana Breast Cancer (BC), the non-heterogeneous data sets are Iris Plant (IRIS), Zoo Animals (ZOO), Wisconsin Breast Cancer (WBC) (Zwitter & Milan Zoklic, 1998). The description of each data set is shown in Table 1.

	CRX	GC	BC	HD	HP	IRIS	ZOO	WBC
Continuous	6	7	0	2	6	4	0	0
Nominal	9	11	6	8	13	0	16	0
Discrete	0	2	3	3	0	0	0	9
Class	2	2	2	2	2	3	7	2
Training	562	900	249	267	132	135	91	629
Testing	62	100	28	30	15	15	10	70

Table 2. Heterogeneous and Non-heterogeneous Data Set

The dataset is distributed into 10 fold cross validation with 90% data for training and 10% data for testing with no overlapping. The data is tested in its original types as provided in the databases. For a consistent condition and comparison on FINERS and FAIRS (Puteh et al., 2008) and other immune algorithms from WEKA toolbox (Witten & Frank, 2005), they are tested using the same sets of 10-fold CV data. The selected immune classifiers from WEKA toolbox are AIRS1 (Watkins, 2001; Watkins et al., 2004; Brownlee, 2005), AIRS2 (Watkins & Timmis, 2002; Brownlee, 2005), AIRS2Parallel (AIRS2P) (Watkins & Timmis, 2004; Brwonlee, 2005), IMMUNOS1 (Brownlee, 2005; Carter, 2000) and CLONALG (Brownlee, 2005; de Castro & Von Zuben, 2000). The average accuracy is calculated from the 10 sets for each dataset and the significant difference is analyzed using paired T-Test using standard statistical package. Table 2 shows the comparison of the accuracy rates and Table 3 shows the comparisons of the rules reduction between FINERS and the other immune algorithms on heterogeneous data. Sig value in 2nd column shows the statistically significant value in differences. The value in bold is the highest accuracy in the table for each data set. The difference is significant if the significant value is less than 0.05 with 95% confidence (Coakes & Steed, 2003). NA is not applicable which means that these classification models do not test the value.

	Sig	ACCURACY				
		CRX	GC	BC	HD	HP
FINERS		87	75	73	89	88
FAIRS	0.070	87	74	72	88	88
AIRS1	0.000	80	67	68	82	83
AIRS2	0.001	83	71	68	82	84
AIRS2P	0.006	81	71	67	80	85
IMMUNOS1	0.027	85	68	71	86	80
CLONALG	0.025	63	70	68	71	75

Table 3. Accuracy (%) of heterogeneous data

		RULES REDUCTION				
	Sig	CRX	GC	BC	HD	HP
FINERS		30	35	71	43	30
FAIRS	0.006	11	28	50	34	12
AIRS1	0.900	62	20	45	42	34
AIRS2	0.045	29	13	33	18	22
AIRS2P	0.021	22	11	23	14	16
IMMUNOS1		NA	NA	NA	NA	NA
CLONALG		NA	NA	NA	NA	NA

Table 4. Rules Reduction (%) of heterogeneous data

The result shows that FINERS gives higher accuracy rate and higher rules reduction percentage in most of the heterogeneous datasets compared to other immune algorithms with statistically significant differences.

Table 4 shows the comparison of the accuracy rates and Table 5 shows the comparisons of the rules reduction between FINERS and the other immune algorithms on non-heterogeneous data. Sig value in 2nd column shows the statistically significant value in differences. The value in bold is the highest accuracy in the table for each data set. The difference is significant if the significant value is less than 0.05 with 95% confidence (Coakes & Steed, 2003). NA is not applicable which means that these classification models do not test the value.

		ACCURACY		
	Sig	IRIS	ZOO	WBC
FINERS		97	89	98
FAIRS	0.681	97	95	97
AIRS1	0.644	96	98	97
AIRS2	0.057	94	89	96
AIRS2P	0.381	94	98	96
IMMUNOS1	0.181	98	96	85
CLONALG	0.240	92	94	94

Table 5. Accuracy (%) of non-heterogeneous data

The result shows that the differences are not statistically significant which means no improvement in accuracy rates by FINERS compared to the previous classification models on non-heterogeneous data. But, for rules reduction, FINERS shows an improvement compare to FAIRS.

		RULES REDUCTION		
	Sig	IRIS	ZOO	WBC
FINERS		39	61	74
FAIRS	0.044	35	57	53
AIRS1	0.166	65	47	46
AIRS2	0.337	65	81	52
AIRS2P	0.133	53	72	48
IMMUNOS1		NA	NA	NA
CLONALG		NA	NA	NA

Table 6. Rule reduction (%) of non-heterogeneous data

5. Conclusion

This paper has proposed a new AIS immune network classifier called Flexible Immune Network Recognition System (FINERS) that uses HVDM as a distance metric for heterogeneous data type without the need for the discretization or transformation of the data into specific type. The experimental results show that the immune network model produces a better accuracy in most of the heterogeneous datasets and it also generates less rules compared to previous immune classification models. Comparing FINERS to FAIRS, although there are no differences in the accuracy for the heterogeneous data, using network feature from the immune system decreases the number of rules in the classifiers. The study solves some limitation shown in (Watkins, 2001; Freitas & Timmis, 2007; Hart & Timmis, 2008; Timmis, 2006). However, FINERS does not show a significant different or improvement on the accuracy and rules reduction on non-heterogeneous data compared to the previous AIS classification models. In conclusion, the results suggest that the use of network feature and to process data in its original types can increase accuracy performance while reducing the number of rules in heterogeneous data. Furthermore, it is significant to process the data in its original types to avoid degradation of data accuracy and it decreases the time in pre processing of data. For the future investigation, other AIS algorithm can employ HVDM function for other tasks such as optimization and clustering. FINERS could also be further refined to make it dynamic and be able to process dynamic data such as time series data. With the result, we hope to derive a more stable and flexible AIS classifier.

6. References

- Brownlee, J. (2005). Artificial Immune Recognition System (AIRS) A review and Analysis, CISCP, Faculty of ICT, Swinburne University of Technology, Australia, Technical Report 1-02
- Brownlee, J. (2005). Clonal Selection Theory & ClonalG and The Clonal Selection Classification Algorithm (CSCA), CISCP, Faculty of ICT, Swinburne University of Technology, Australia, Technical Report 1-02

- Brownlee, J. (2005). Immunos-81 The Misunderstood Artificial Immune System, CISCP, Faculty of ICT, Swinburne University of Technology, Australia, Technical Report 1-02
- Burnet, F. M. (1959). The Clonal Selection Theory of Acquired Immunity, Cambridge University Press
- Carter, J. H. (2000). The Immune Systems as a Model for Pattern Recognition and Classification, Journal of the American Medical Informatics Association 7 (1)
- Coakes, S. J. & Steed, L.G. (2003). SPSS Analysis without Anguish Version 11.0 for Windows, John Wiley & Sons Australia, Ltd
- Cover, T. M. & Hart, P. E. (1967). Nearest Neighbor Pattern Classification, IEEE Transactions on Information Theory, Vol 13, No. 1, pp. 21-27
- Dasarathy & Belur, V. (1991). Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques , IEEE Computer Society Press Los Alamitos, CA
- Dasgupta, D. (2006). Advances in Artificial Immune Systems, IEEE Computational Intelligence Magazine
- de Castro, L. N. & Von Zuben, F. (2000). The Clonal Selection Algorithm with Engineering Applications. *Proceedings of GEC2000*, Las Vegas, pp.36-37
- de Castro, L. N. & Timmis, J. (2002). Artificial Immune Systems: A New Computational Intelligence Approach, Springer-Verlag
- Freitas, A. & Timmis, J. (2007). Revisiting the Foundations of Artificial Immune Systems for Data Mining, IEEE Transactions on Evolutionary Computation, 11:4
- Hamaker, J. & Boggess, L. (2004). Non-Euclidean Distance Measures in AIRS, an Artificial Immune Classification System, *Proceedings of CEC2004*
- Hart, E. & Timmis, J. (2008). Applications of Artificial Immune Systems: The Past, the Present and the Future, Journal of Soft Computing, vol 8(1) 191-201
- Hart, P. E. (1968). The Condensed Nearest Neighbor Rule, IEEE Transactions on Information Theory, Vol 14, pp.515-516
- Hunt, J. E. & Cooke, D. E. (1996). Learning Using an Artificial Immune System, Journal of Network Computer Applications, 19:189-212
- Jerne, N. K. (1974). Towards a Network Theory of the Immune System, Ann. Immunol. (Inst. Pasteur) 125C, 373 - 389
- Merz C. J. & Murphy, P. M. (1998). UCI Machine Learning Repository, University of California, Irvine, USA, <http://www.ics.uci.edu/~mlearn/MLRepository.html>
- Puteh, M. ; Hamdan, A. R. ; Omar, K. & Abu Bakar, A. (2008). Classifying Heterogeneous Data with Artificial Immune System, *Proceedings of IEEE International Symposium on Information Technology (ITSIM2008)*
- Stanfill, C. & Waltz, D. (1986). Towards Memory-based Reasoning, ACM, Vol 29, pp. 1213-1228
- Timmis, J. (2001). Artificial Immune Systems: A Novel Data Analysis Technique Inspired by the Immune Network Theory, Ph. D. thesis, Department of Computer Science, University of Wales, Aberystwyth
- Timmis, J. (2006). Challenges for Artificial Immune System, LNCS 3931, pp355-367, Springer Verlag
- Ventura ; Dan & Martinez, T. R. (1995). An Empirical Comparison of Discretization Methods, *Proceedings of the 10th International Symposium on Computer and Information Sciences*, pp. 443-450

- Watkins, A. (2001). A Resource Limited Artificial Immune Classifier, Master's Thesis, Mississippi State University
- Watkins, A.; Timmis, J. & L. Boggess, L. (2004). Artificial Immune Recognition System (AIRS): An Immune-Inspired Supervised Learning Algorithm, Genetic Programming and Evolvable Machines, vol 5, pp291-317
- Watkins, A. & Timmis, J. (2002). Artificial Immune Recognition System (AIRS): Revisions and Refinements, *Proceeding of ICARIS2002*, pp173-181, Springer Verlag
- Watkins, A. & Timmis, J. (2004). Exploiting the Parallelism Inherent in AIRS, an Artificial Immune Classifier, *Proceeding of ICARIS2004*, pp427-438, Springer Verlag
- Wilson, D. R. & Martinez, T. R. (1997). Improved Heterogeneous Distance Functions, *Journal of Artificial Intelligence Research* 6, 1- 34
- Witten, I. H. & Frank, E. (2005). Data Mining, Morgan Kaufmann Publishers, USA
- Zwitter, M. & Milan Soklic, M. (1998). Institute of Oncology, University Medical Center, Ljubljana, Yugoslavia

Modified Cascade Correlation Neural Network and its Applications to Multidisciplinary Analysis Design and Optimization in Ship Design

Adeline Schmitz, Frederick Courouble,

Hamid Hefazi and Eric Besnard

California State University, Long Beach

USA

1. Introduction

Artificial Neural Networks (NN) basically attempt to replicate functions of the human brain by connecting neurons to each other in specific manners. In most cases, the number of neurons and connections has been limited to tens or hundreds of neurons with a similar order of magnitude of connections between them. This contrasts with the human brain which has many orders of magnitude more neurons and also many more connections between them. The work done so far in this field can therefore be categorized as precursory and the potential of this technology has yet to be fully realized. Jain & Deo (2005) present a survey of applications of NNs in ocean engineering. They have been used for predicting environmental parameters (wave heights, sea level, wind speeds, tides, etc.), forces and damage on ocean-going structures, and ship and barge motions, in various ship design applications and more. Gouguolidis (2008) presents an overview of utilization of neural networks in many other marine applications including structures, stability, propulsion and seakeeping. All of the applications that are reviewed, except one, are for predictions of characteristics of ships.

In the area of ship design, NNs have been employed for various purposes. For example, Koushan (2003) presents a hull form optimization employing NNs in which eight hull form parameters are varied in order to minimize resistance. Similarly, the use of NNs in hull shape optimization by Danışman *et al.*(2002) and Koh *et al.* (2005) allows for a more advanced flow model (panel method instead of thin-ship flow theory) and thus leads to improved shapes. Koh (2004) present a similar approach for investigating resistance, manoeuvrability and seakeeping characteristics of a high speed hull form. In Mesbahi & Bertram (2000), and Bertram & Mesbahi (2004), NNs are used to derive functional relations between design parameters for cargo ships as an alternative to using charts. In Maisonneuve (2003), several NN application examples from the industry are discussed, showing state-of-the-art of European research in marine design: integrating real calculations (using CAD

model) and artificial calculations (using NNs as response surface methods) to perform single- or multi-objective optimizations.

In most of the applications above, a single hidden layer feed-forward type of network and back-propagation are used. Also, the number of input nodes used by the different investigators was relatively small, on the order of one to ten. Very few applications used a large number of inputs to utilize the real power of the neural networks. One exception is Danışman *et al.* (2002) which used a single, hidden layer NN with 40 input and four output parameters. The inputs represent a series of half breadths of the aft end of a catamaran and the output parameters are related to wave resistance, wave elevation and displacement. The network used back-propagation with a training set of 300 and a validation set of 50 points. The number of hidden units used and the errors produced by the neural network are not discussed in the paper, however.

As the number of parameters to be varied increases, training the network becomes more and more challenging. It is demonstrated that feedforward NNs have universal approximation ability for a wide variety of functions classes, provided that a sufficient number of hidden units are available (Cybenko, 1989, Hornik, 1991). The approximation ability of a particular network depends on the numbers of input and output units, the number of training cases, the amount of noise in the targets, the complexity of the function to be learned, the actual architecture of the network, the type of hidden unit activation function and the training algorithm. This often leaves NN users having to determine the network size by trial and error. Also, back-propagation, the most commonly used algorithm to train single hidden layer feedforward NNs, is known to be very slow for large input spaces. Other network structures and training algorithms should be investigated.

This chapter presents an alternative NN structure based on a constructive network topology and a corresponding training algorithm suitable for large number of input/outputs to address the problems where the number of design parameters is fairly large, say up to 30 or even more.

The chapter is divided into four sections. First, the use of NNs as advanced regression models in the ship design cycle is reviewed and the choice of the particular topology (constructive network) and training algorithm (modified cascade correlation, or "MCC") of the NN is justified.

The next section describes in detail the MCC. This algorithm is an improvement from the original cascade algorithm introduced by Fahlman and Lebiere (1990). Improvements include altering the weight initialization, modifying the candidate hidden unit training, and introducing normalized inputs, "early stopping" and "ensemble averaging."

Next, the NN approach is applied to the design/optimization of an underwater hull configuration using a genetic algorithm search method. Results are compared with those obtained with a classical optimization approach in which the CFD code is directly coupled with the optimizer.

The last section presents a NN-based performance analysis and optimization of sailing configurations of an America's Cup class yacht. The objective is to maximize boat speed (objective function) by varying the sailing setups (design variables). In the approach, the experimental data from the sailing records provided by sensors is used to train an MCC neural network. The network is coupled with a genetic algorithm to determine the maximum boat speed and corresponding yacht settings at various wind speeds. Because the majority of the data is gathered in a small region of the search space corresponding to a

valid set of sailing configurations, the remaining regions of the domain are not well populated and can lead to training errors for the neural network. The chapter presents an automatic method to fill the domain of investigation by adding artificial points to the database in regions without sufficient experimental data.

2. Neural Networks as Response Surface Methods in the Design Cycle

2.1 NNs in the Design Cycle

The systems engineering approach, originated and widely used in the aerospace industry, consists of decomposing a system into subsystems. For a ship, those would correspond to the hull forms definition, propulsion, structure, payload, etc. This system decomposition approach is described, for example, in Blanchard & Fabrycky (1997), and can be applied at the ship level as well as at subsystem levels in the systems architecture. For example, this systems approach may also be used to design a propulsion subsystem which will be integrated into the ship, based on requirements established at higher levels. In other words, every *component* may be looked as a system which gets integrated into a *system of systems*.

The analyses performed at each subsystem level rely, in general, on a combination of semi-analytical models, advanced numerical methods such as finite element analysis, and use of existing databases. The modern approach used in the design of such systems usually includes optimization at some level.

Neural networks may be inserted directly at all levels of the system design process and on a broad basis. Specialists who use advanced computational tools for detailed analyses are often remotely connected to the design loop. The use of NN allows for them to be indirectly integrated very early into the design cycle by generating a computational database representative of the problem at hand over the desired design space. For example, the database might consist of a few hundred CFD analyses performed for a configuration represented by tens of widely varying design parameters. This database can then be used to train –hence its name, “training set”– a neural network which is then inserted in the design loop (Fig. 1). At this point, the designer (not the analyst) can use the NN and get a solution for a variety of designs in a fraction of a second. For example, if a network has been trained for estimating the ship resistance, the latter can be obtained by the designer for any desired point in the design space with minimal computational time.

Similar uses of neural networks can be made when dealing with available large databases. Such databases may be from one or more sources, numerical and/or experimental. In this case, the database can be used directly to train the NN and the latter can also be integrated into the design loop (Fig. 1).

The result is a design approach in which the function, such as ship resistance, corresponding to a particular set of design variables either selected by the designer or by the computer (“design tool”), can be obtained instantaneously.

In practical terms, the introduction of NN allows for extraction of time-consuming or difficult operations (performing an advanced numerical analysis or extracting information from a large and evolving database) from the design loop while still keeping their influence on the outcome of the design process via the NN. The cost has thus been moved (and possibly reduced in the process) to the training set generation (if it was not already available) and to the training of the network.

The result is a NN which can estimate the function or functions over the design space it has been trained on. This ability to quickly evaluate new designs allows in turn for the use of global optimization tools such as genetic algorithms instead of having to rely on local optimization methods or exploring a restricted part of the design space.

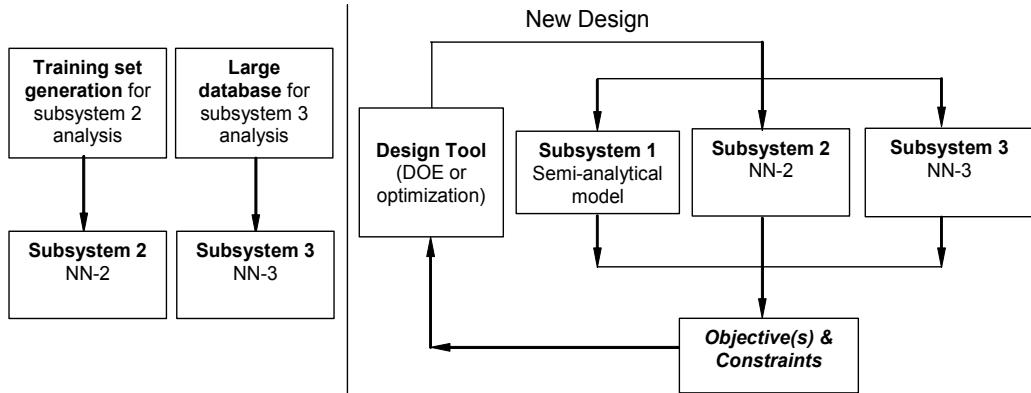


Fig. 1. System design loop utilizing NNs. The NNs are generated outside the design loop based on computationally extensive models and/or large databases.

2.2 Advantages Offered by Constructive Neural Networks

NNs have been investigated for many applications outside the field of ocean engineering that require a large number of function analyses, ranging from chemistry (Agatonovic-Kustrin *et al.*, 1998, and Takayama *et al.*, 2003) to structural analysis (Deng *et al.*, 2005). Research clearly indicates that NNs compare favourably with classical RSM (Gouguolidis, 2008, Todoroki *et al.*, 2004, Bourquin *et al.*, 1998, Gomes & Awruch 2004, Dutt *et al.*, 2004 and Lee & Hajel, 2001), in particular when the function is non-convex over the desired domain and the function may be highly nonlinear. In addition, for problems with a large number of inputs (design variables), the size of the dataset required for the classical RSM rapidly grows. Schmitz (2007) and Besnard *et al.* (2007) present a survey of the different RSM available and demonstrate the clear advantage in using NNs in this type of application. As discussed above, for most marine applications reported to date, however, NNs employed either a fixed topology and/or back propagation for training. The former implies that one needs to have some information about the function to approximate and the latter leads to increasingly large CPU time requirements for training in the case of a large number of inputs.

Methods which use a fixed network topology involve evaluating in advance (before training) the type of network that would best suit the application (how many neurons, how many hidden-layers) to match the complexity of the NN to that of the function. This point is best illustrated by Kwok & Yeung (1997a): *“Consider a data set generated from a smooth underlying function with additive noise on the outputs. A polynomial with too few coefficients will be unable to capture the underlying function from which the data was generated, while a polynomial with too many coefficients will fit the noise in the data and again result in a poor representation of the underlying function. For an optimal number of coefficients the fitted polynomial will give the best representation of the function and also the best predictions for new data. A similar situation arises in the application of NN, where it is again necessary to match the network complexity to the problem*

being solved. Algorithms that can find an appropriate network architecture automatically are thus highly desirable."

There are essentially two approaches for training multilayer feedforward networks for function approximation which can lead to variable networks (Kwok and Yeung 1997a). The Pruning Algorithms start with a large network, trains the network weights until an acceptable solution is found, and then uses a pruning method to remove unnecessary units or weights (units connected with very small weights). On the other hand, Constructive Algorithms start with a minimal network, and then grow additional hidden units as needed. The primary advantage of constructive algorithms versus pruning algorithms is that the NN size is automatically determined. Constructive algorithms are computationally economical compared to pruning algorithms which spend most time training on networks larger than necessary. Also, they are likely to find smaller network solutions, thus requiring less training data for good generalization. They also require a small amount of memory because they usually use a "greedy" approach where only part of the weights is trained at once, whereas the remaining part is kept constant (Schmitz 2007).

Cascade-Correlation, first introduced by Fahlman & Lebiere (1990), is one such supervised learning algorithm for NNs. Instead of just adjusting the weights in a network of fixed topology, Cascade-Correlation begins with a minimal network, then automatically trains and adds new hidden units one-by-one in a cascading manner. This architecture has several advantages over other algorithms: it learns very quickly; the network determines its own size and topology; it retains the structure it has built even if the training set changes; and it requires no back-propagation of error signals through the connections of the network. In addition, for a large number of inputs (design variables), the most widely used learning algorithm, back-propagation, is known to be very slow. Cascade-Correlation does not exhibit this limitation (Fahlman & Lebiere 1990).

3. Modified Cascade Correlation Neural Networks

As mentioned above, the constructive Cascade-Correlation algorithm begins with a minimal network consisting of the input and output layers and no hidden unit (neurons), then automatically trains and adds one hidden unit at a time until the error (E_p) between the targets (f_p) of the training set and the outputs from the network ($f_{NN,p}$) reaches a desired minimal value. Thus, it self-determines the number of neurons needed as well as their connectivity or weights.

The original algorithm of Fahlman & Lebiere (1990) was geared towards pattern recognition and has been improved to make it a robust and accurate method for function approximation (Schmitz *et al.*, 2002, Schmitz, 2007). Although the definitions used here assume that the NN has a single output, i.e. that the NN represents a single scalar function such as ship resistance, the process is easily extended to networks with multiple outputs (Schmitz *et al.*, 2002). In the applications considered in this paper, multiple functions, such as ship resistance and ship displacement, are each represented by a separate network, i.e. each uses multiple single-output networks rather than a single multiple-output network. It was found during the course of the study that it was more advantageous to train multiple single-output networks than one large multiple output network in terms of error on an unseen dataset (generalization error) and computing time (since multiple "single output" networks could be trained simultaneously faster than one single "multiple output" network).

3.1 Base Algorithm as Introduced by Fahlman and Lebiere

The basic algorithm as introduced by Fahlman and Lebiere (1990) includes the following 11 steps:

Step 1: Start with the required input and output units; both layers are fully connected. The number of inputs and outputs is dictated by the problem.

Step 2: Train all connections ending at an output unit with a common learning algorithm until the squared error E_s of the NN no longer decreases.

$$E_s = \frac{1}{2} \sum_{p=1}^{N_p} \|y_p - t_p\|^2 = \frac{1}{2} \sum_{p=1}^{N_p} \sum_{i=1}^m [y_{i,p} - t_{i,p}]^2 \quad (1)$$

Here m is the size of the outputs (or number of outputs), N_p is the size of the training set, $y_{i,p}$ is the i^{th} output from NN, and $t_{i,p}$ is the corresponding target.

Step 3: Generate a candidate unit that receives trainable input connections from all of the network's external inputs and from all pre-existing hidden units (if any). The output of this candidate unit is not yet connected to the active network (output).

Step 4: Train the unit (its weight) to maximize the correlation referred to as S_C (Eq. 2). Learning takes place with an ordinary learning algorithm; training is stopped when the correlation score no longer improves. The correlation formula is given by

$$S_C = \left| \sum_{i=1}^m \left(\sum_{p=1}^{P_{\max}} (z_{o,p} - \bar{z}_o)(E_{i,p} - \bar{E}_i) \right) \right| \quad (2)$$

Here $z_{o,p}$ is the output of the candidate hidden unit and $E_{i,p}$ is the residual error of the outputs calculated at Step 2, $E_{i,p} = y_{i,p} - t_{i,p}$. The bar above a quantity denotes the average over the training set.

Step 5: Connect the candidate unit with the outputs and freeze its input weights. The candidate unit acts now as an additional input unit.

Step 6: Train again the input-outputs connections by minimizing the squared error E_s as defined in Step 2.

Steps 7 to 10: Repeat Steps three to six adding one hidden unit at a time.

Step 11: Stop training when the error E of the net falls below a given value, ε .

Instead of a single candidate unit, it is possible to use a pool of candidate units, each with a different set of random initial weights. All receive the same input signals and see the same residual error for each training pattern. Because they do not interact with one another, or affect the active network, they can be trained simultaneously. Only the candidate whose correlation score is the best is installed. The use of a pool of candidates greatly reduces the chances that a useless unit will be permanently installed because an individual candidate got stuck during training. Fahlman and Lebiere (1990) typically show that four to eight candidate units are enough to ensure good candidates in each pool.

Steps 2 and 4 require the use of an optimization routine. Fahlman uses the so-called Quickprop algorithm. Quickprop computes the derivative of the error with respect to the weights as in standard back-propagation, but instead of simple gradient descent, Quickprop uses a second order method, related to Newton's method, to update the weights (Fahlman, 1988).

3.2 Overview of Algorithm Modifications for Function Approximation

While the basic CC algorithm described in the previous section provides a good foundation for regression applications, it also has areas which can be improved. This section presents the modifications that were developed and implemented to the CC algorithm described above. Fahlman introduced this algorithm for classification tasks which typically use a large number of inputs. The network is thus well suited for the application in mind in this research, i.e. approximation of functions with a large number of variables. Practical optimization problems require a large number of design variables which define the configuration to be optimized. CC algorithm learns very quickly and uses minimal computer memory as it only trains some of the network weights while others are frozen. It has been shown to work well for regression tasks (Kwok & Yeung, 1993, 1997a and 1997b, Prechelt, 1997, Treagold & Gedeon, 1999, Lehtokangas, 1999, Lahnajärvi *et al.*, 2002). Each author, however, points out some potential downfalls of the algorithm and proposes some possible fixes. These problems are primarily:

- Maximizing Fahlman's correlation formula trains candidate neurons to have a large activation (weight) whenever the error at their output is not equal to the average error. Cascade correlation has a tendency to overcompensate errors. (Prechelt, 1997)
- The candidate unit weight optimization might get stuck in a local maximum, and thus units which are not correlating well with the error are installed on the NN, leading to more units than necessary to reach the desired level of accuracy (deeper network) (Lehtokangas, 1999, Kwok & Yeung, 1997b, Lahnajärvi *et al.*, 2002).
- Cascading units can result in a network that can exhibit very strong non-linearities, thus affecting generalization (Kwok & Yeung 1993, 1997a).

The critical issue in developing a neural network for regression tasks is generalization: how well will the network make predictions for cases that are not in the training set? Neural networks, like other nonlinear estimation methods such as kernel regression and even linear methods like polynomial regression, can suffer from either underfitting or overfitting. As stated in Sarle (2002): "*A network that is not sufficiently complex can fail to detect fully the signal in a complicated data set, leading to underfitting. A network that is too complex may fit the noise, not just the signal, leading to overfitting. Overfitting is especially dangerous because it can easily lead to predictions that are far beyond the range of the training data with many of the common types of NNs.*"

Model selection plays an important role in the generalization ability of the network. As explained above, constructive methods, like cascade correlation, are usually better methods than fixed network topologies trained with back-propagation or pruning methods because they automatically find the number of hidden units that matches the complexity of the problem. They also find smaller network solutions. Smaller networks mean less connections or weights to adjust and thus usually require smaller training sets for similar generalization ability (Kwok & Yeung, 1997b).

The generalization ability of the NN has been addressed in the Modified Cascade Correlation algorithm. Various improvements to the original CC based on some of the solutions proposed by the above-referenced works have been implemented. Extensive research has been conducted that demonstrates the clear advantages of using the MCC on a test function for dimensions varying from 2 to 30 inputs (Schmitz 2007).

In the MCC, inputs and outputs to the NN are non-dimensionalized, weights are constrained to a maximum value in order to limit strong non-linearities of the response

surface, training of the input-to-hidden-unit weights and hidden-to-output weights is performed with a second order optimization method (Sequential Quadratic Programming) instead of the Quickprop algorithm introduced by the original authors. Several stopping criteria are also available to limit overfitting of the network; they all continue training slightly past the minimum validation error (error measured on the Validation Set) and the resulting network is that which has the smallest squared error on the VS, whereas the original authors stop when the error on the training set reaches a predetermined value. Also, ensemble averaging, a well known technique for reducing overfitting is available when training with the MCC. These improvements are described in more detail in the following sections.

3.2.1 Normalization of Inputs

The activation function of the hidden units is usually highly nonlinear. In this research, the activation function chosen is the sigmoid function which varies between zero and one. During training, weights are initialized with small random values. It is commonly known that optimization algorithms will perform faster if optimization is started in an area where the objective function varies rapidly. It is thus better to ensure that the hidden units are not in their saturated portion but rather in the area of the sigmoid which is quasi-linearly varying when optimization is started. Along with using small initial weights, it was also decided to normalize the inputs to the NN. The training set minimum and maximum values are first calculated for each input i .

$$\text{MinInput}_i = \min_{1 \leq p \leq Np} (z_{ip}) \quad (3)$$

$$\text{MaxInput}_i = \max_{1 \leq p \leq Np} (z_{ip}) \quad (4)$$

where Np is the size of the training set and z_{ip} is the i^{th} input for the p^{th} point of the training set.

The training set is next rescaled from zero to one according to the equation below.

$$\begin{bmatrix} z_{1p} \\ \vdots \\ z_{np} \end{bmatrix} = \begin{bmatrix} \frac{z_{1p} - \text{MinInput}_1}{\text{MaxInput}_1 - \text{MinInput}_1} \\ \vdots \\ \frac{z_{np} - \text{MinInput}_n}{\text{MaxInput}_n - \text{MinInput}_n} \end{bmatrix} \quad (\text{for } p \in \{1 \dots Np\}) \quad (5)$$

Also the validation and generalization sets are rescaled using the same minimum and maximum values found for the training set, so they will vary from around zero to one (but not exactly between 0 and 1).

3.2.2 Weights Initialization

In any nonlinear optimization problem, the initialization of the parameters has an important influence on the ability of the training program to converge and the speed of that

convergence. The training of weights in NNs can be viewed as a nonlinear optimization problem in which the goal is to find a set of network weights that optimizes a cost function. In the MCC algorithm, there are two separate optimization problems. The first one is to maximize a correlation function to train the candidate hidden unit newly added to the network; the other is to minimize the error on the training set. Both describe a surface in the weight space. Training algorithms are simply methods used to find the minimum of this surface. The complexity of the search is governed by the nature of this surface. Error surfaces for multilayer NNs have typically many flat regions where learning is slow and long narrow “canyons” that are flat in one direction and steep in the other directions. This makes it very difficult to search the surface efficiently using gradient-based routines. In addition, the cost function is characterized by a large number of local minima with values in the vicinity of the best global minimum. The efficiency of the search method depends much on the initial weight distribution. The simplest category among the weight initialization methods is random weight initialization. It is commonly known that if all the weights of an NN are initialized with a zero, they cannot change to any other value during training if some simple training algorithms are used. Random initialization has been proposed to avoid this undesired situation and its ability to break the symmetry. Very little research has been reported on weight initialization in the literature (Lehtokangas, 1999 and Lahnajärvi *et al.*, 2002).

In the cascade correlation algorithm, there are three separate weight optimization problems to investigate:

1. The first optimization problem is the squared error minimization between input and output units before any hidden unit is added to the network. A previous study showed that the weights between the inputs and the outputs should be initialized randomly between -0.5 and +0.5 (Hefazi *et al.*, 2003).
2. The second optimization problem consists of finding the best candidate hidden unit by maximizing the correlation formula. The weight initialization consists in calculating the norm of the input vector $Z_p, \|Z_p\|$, for each training set point p (also called pattern), and then initializing the weights for the new candidate unit w_j so that:

3.

$$\|\mathbf{w}\| = \sum_{j=1}^{n+h+1} w_j^2 \leq 4 * \max_{p=1,\dots,N_p} (\|Z_p\|). \quad (6)$$

Also during the optimization, the weights values are limited between -10 and +10, as lower values lead to very deep networks and higher values lead to severe overfitting of the data.

4. A third optimization consists of minimizing again the squared error after a new hidden unit (h^{th} hidden unit) has been added to the network and connected to the outputs. Hefazi *et al.* (2003) showed that the weights v_{ij} found at the previous step (unit $h-1$) are already close to the optimal value with this new hidden unit added to the network. For this weight initialization problem, it is then best to use, as initial weights, the ones found at the previous iteration (weights between the inputs and previous hidden units to hidden unit $h-1$) and to initialize at zero the new weights between the inputs and previous hidden unit to hidden unit h . This method leads to the fastest search for the

optimum as well as the smallest overfitting. Similarly, the weights are allowed to vary only between -10 and +10 during optimization.

3.2.3 Choice of optimization routine

The weight optimization must be solved by using some optimization software. Kwok & Yeung (1993) demonstrate that the CC algorithm can always reach $Es < \varepsilon$ for a given $\varepsilon > 0$ for L² functions, even when using a local optimizer. Fahlman (1988) uses its own optimization routine to update the weights; the Quickprop algorithm, a second order local search method related to Newton's method. In this research, a commercially available software DOT, developed by Vanderplaats (1995) was chosen. This software contains a choice of the latest state-of-the-art optimization methods. Therefore the Broydon-Fletcher-Goldfarb-Shanno (BFGS) method from DOT software was chosen for its proven efficiency and accuracy for unconstrained optimization problems. It is also a quasi-Newtonian method because it creates an approximation of the inverse of the Hessian matrix. The details of the method are explained in Vanderplaats (1995). One advantage of using this software is that the gradient of the squared error and the correlation formula can be supplied directly to DOT and, thus, considerably speed up the weight optimization.

3.2.4 Candidate Hidden Unit Training

Fahlman's original algorithm calls for randomly initializing a pool of four to eight candidate units and then maximizing all candidate units. The candidate whose correlation score is the highest is then added to the network. This is done because, as explained in 0, the weight surface has many local maxima. And since the method used for finding the best weights is a gradient search, i.e. local search, the optimization may get stuck in a local maxima and fail to find the global one. So doing several searches starting with different initial weight values increases the chance of finding the "global" optimum. One might want to use a global search method, but this becomes prohibitive in terms of computer time requirements. Another idea is to use a much larger pool, of the order of 100-500 candidates, initialized at random and then only optimizing the one whose correlation after random initialization is best. Random initialization is very fast, and increases the chance of starting the optimization with a unit close to the global optimum and only one candidate is trained using the time consuming optimization algorithm. Lehtokangas (1999) has applied this method successfully to his constructive algorithm and found it beneficial in terms of time requirements and performance of the NN. Both options are implemented in the algorithm. A study in Schmitz (2007) shows that for a number of inputs greater than 5 or 10, it is advantageous to use the method using a large pool of candidate units.

3.2.5 Stopping Criterion

When training a NN, one is usually interested in obtaining a network with optimal generalization performance. Generalization performance means small errors on examples not seen during training. As hidden units are added to the network, the error on the TS decreases, i.e. the network is able to fit the training data better. However, when looking at the error on an unseen data set, the error initially decreases but at some point during training it increases. The network starts to overfit the training data and the generalization ability of the network gets worse. This is even more pronounced when the data is noisy

(Bishop, 1995). This phenomenon is called the bias variance tradeoff; underfitting produces excessive bias in the outputs, whereas overfitting produces excessive variance. To our knowledge, Fahlman and Lebiere did not study the generalization properties of their CC network and looked only at the convergence of their network on the training data.

An easy way to find a network having the best performance on new data is to evaluate the error function using data which is independent of that used for training, i.e. on the validation set (VS) and to stop training when the error is minimum on the VS. This method is called *early stopping* or stopping the learning procedure before full convergence of the network on the training set (TS) to obtain optimal generalization properties. It is widely used in all feedforward NN architectures. Because of the stochastic nature of the CC algorithm, the minimum validation error might exhibit several local minima as hidden units are added one by one before the global minimum can be attained. This implies that one must continue training the NN past each local minimum to make sure that the global minimum has been found and then choose the network with the number of hidden units which correspond to this minimum validation error. Prechelt (1998a & 1998b) has derived several classes of stopping criteria which may be used to determine how long training should be continued to make sure that the global minimum has been found. These criteria are described in detail in Schmitz (2007). Only the criterion used in the applications presented in this chapter is described in Section 0.

3.2.6 Ensemble Averaging

Because of the stochastic nature of the process in building NNs, it is a common practice to train many different candidate networks and then to keep only the one with best performance. Each network leads to different weight values, different numbers of HUs and different errors. Usually, the one with the best performance is chosen. Performance is usually measured by how the network predicts data on an independent validation set. There are two disadvantages in this approach. First, the effort involved in training the remaining networks is wasted. Second, the generalization performance on the validation set has a random component since it is a relatively small set and so the network which had best performance on the validation set might not be the one with the best generalization, i.e. performance on the rest of the computational domain. These drawbacks can be overcome by combining the networks together by forming a committee. There are several ways of combining networks; one simple way is to take the output of the committee to be the average output of each individual network. This method, called *ensemble averaging*, appears to be a very simple way to limit the overfitting of the network. According to Bishop (1995), the error on the committee is always less than the average error calculated by averaging the error on each individual network. Also, networks trained with the CC algorithm usually show strong non-linearities in their response because the hidden units are added in cascade, making a network with many layers and one hidden unit in each layer. In the application in mind involving approximation of smooth functions, the idea of using all the networks constructed, and average them out to "smooth out" the response surface, appears promising. Tekto & Villa (1997) have done some preliminary research on ensemble averaging combined with early stopping (ESE) for NNs trained according to the cascade correlation algorithm for simple single input/single output functions. Their work shows that the technique they call ESE provides an improvement in the generalization ability of the network for those test cases. An extensive study in Schmitz (2007) has shown that ensemble

averaging always improves the generalization ability of the NN and should indeed be used each time an NN is constructed with the MCC algorithm.

3.3 Equations/Mathematical Formulation/Algorithm

This section describes the mathematical formulation the modified CC algorithm for function approximation. The training algorithm was programmed in C++ language and coupled with the DOT software which uses FORTRAN language.

3.3.1 Step 1: No Hidden Units/Linear Inputs to Outputs Connection

In the first step of building the network with the cascade correlation algorithm, there are no hidden units. Inputs and outputs are fully connected, the weights, v_{ij} , determine the strength of the connection from the i^{th} input to the j^{th} output. These will need to be adjusted to minimize the squared error E_s . Fig. 2 shows a schematic of the input-outputs connections without hidden units. The vertical lines sum all incoming activation. X connections correspond to weights to be trained. Square boxes represent neurons; input-output neurons have a linear activation function.

The bias term can be modelled in the equations as an additional input unit of value one and with weighted connections to the outputs. Without loss of generality, the output neurons can have a linear activation function of slope one (i.e. $\varphi(z)=z$) because there is always a linear component to a nonlinear function, thus a linear link between inputs and outputs. This greatly simplifies the equations for training the NN. The bias parameter is useful to compensate for the difference between the mean (over the training set) of the output vector and the corresponding mean of the target data.

Based on this NN, the relationship between input and output is given by

$$\left(\forall p \in \{1 \dots N_p\} \right) \begin{bmatrix} y_{1p} \\ \vdots \\ y_{mp} \end{bmatrix} = \begin{bmatrix} v_{11} & \dots & \dots & v_{1,n+1} \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ v_{m1} & \dots & \dots & v_{m,n+1} \end{bmatrix} \begin{bmatrix} z_{1p} \\ \vdots \\ z_{np} \\ +1 \end{bmatrix} \quad (7)$$

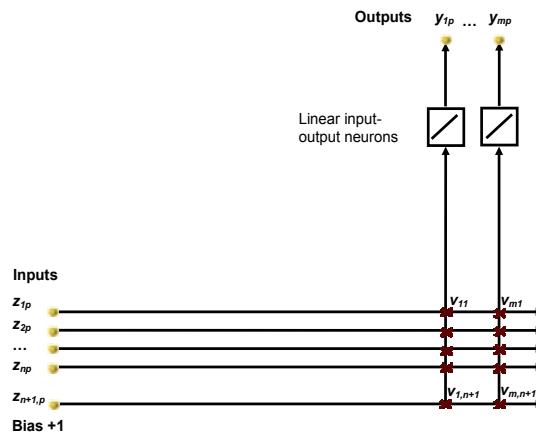


Fig. 2. Schematic of input-output connections without hidden units. The vertical lines sum all incoming activation. X connections or weights must be trained.

3.3.2 Step 2: Minimize Squared Error without Hidden Unit

The next step consists of adjusting the weights to fit the training data. The squared error between the targets and the outputs is used as the standard error measure and must be minimized. The squared error for the neural network without hidden unit is given by the following equation:

$$Es = \frac{1}{2} \sum_{i=1}^m \sum_{p=1}^{Np} \|y_{ip} - t_{ip}\|^2 = \frac{1}{2} \sum_{i=1}^m \sum_{p=1}^{Np} \left[\sum_{j=1}^n v_{ij} z_{jp} + v_{i,n+1} - t_{ip} \right]^2 \quad (8)$$

The weights v_{ij} are initialized randomly between [-0.5, +0.5] as discussed in Section 0. The error is then minimized by adjusting the weights using the BFGS method from DOT optimization software (Vanderplaats, 1995). DOT allows the user to directly input the gradient of the function to optimize, if known, and to speed up the optimization process. The gradient of squared error with respect to the weights $\partial Es / \partial v_{kl}$ can be calculated analytically as follow:

$$\frac{\partial Es}{\partial v_{kl}} = \sum_{p=1}^{Np} \left[\sum_{j=1}^n v_{kj} z_{jp} + v_{k,n+1} - t_{kp} \right] z_{lp} \quad (9)$$

with $k \in \{1 \dots m\}$, $l \in \{1 \dots n+1\}$ and $j \in \{1 \dots n\}$.

It is noteworthy to point out here that a pseudo inverse method could also be used to find the minimum error since it is a linear system. However BFGS works fast on linear systems and is subsequently used to find the candidate units weights once hidden units have been added and the system is no longer linear. This approach was used here instead of calculating the pseudo inverse matrix.

3.3.3 Step 3: Adding a First Hidden Unit Connected to Inputs only

After optimizing the matrix of weights, V, a first hidden unit is connected to the inputs as shown in Fig. 3. Its output is noted $z_{n+2,p}$.

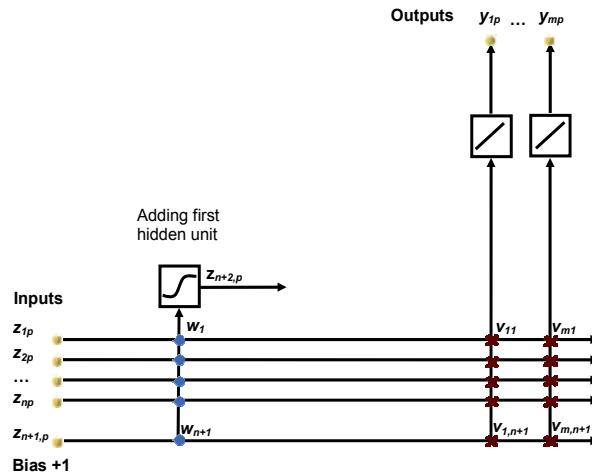


Fig. 3. Schematic of input to first hidden unit connections. Hidden unit not yet connected to outputs. Diamond connections to be trained.

The connections (weights) between inputs and the hidden unit are noted w_j in

$$z_{n+2,p} = \sigma \left(\begin{bmatrix} w_1 \cdots w_n w_{n+1} \\ \vdots \\ z_{n+1,p} \end{bmatrix} \right) = \sigma \left(\sum_{j=1}^{n+1} w_j z_{jp} \right) \quad (10)$$

where $j = 1, \dots, n+1$ and σ is the sigmoid function. Also, to simplify the notation, the +1 of the bias is replaced by the notation $z_{n+1,p}$ in the equations.

3.3.4 Step 4: Maximize Correlation Formula for First Hidden Unit

The next step in the CC algorithm is to maximize the correlation for the new hidden unit installed on the network. Optimization is performed with the BFGS method from DOT (Vanderplaats, 1995).

For this weight initialization, the norm of the input vector Z_p , $\|Z_p\|$, is calculated for each training set point p (also called pattern). And the weights for the new candidate unit w_j are initialized so that:

$$\sum_{j=1}^{n+1} w_j^2 \leq 4 * \max_{p=1, \dots, N_p} (\|Z_p\|) \quad (11)$$

to avoid starting the optimization in the highly saturated part of the sigmoid, and thus getting a null derivative of the correlation formula with respect to the weights w_j . In fact, a pool of candidate hidden units is generated with different random initial weights, and two options are available to train the candidates depending on the size of the pool chosen. If this number is less than 10, the algorithm is programmed so that all candidate units in the pool are trained to maximize the chosen correlation formula using the BFGS algorithm. Only the unit with the largest correlation value after training is next installed on the network. This method is the same as Fahlman's algorithm except for the use of the BFGS instead of the Quickprop (Fahlman, 1988) algorithm. If the size of the pool is greater than 10, then only the candidate unit which exhibits the largest correlation value after random initialization is trained with the BFGS method and next permanently installed on the NN. This builds the network faster since the time-consuming operation of optimizing the weights is done only once. The other option implies optimizing several candidates. For this method to work well, it is recommended to use a large pool, say 100 to 500 candidates. Only the candidate whose correlation is the highest is kept in memory. Its weights are saved in a separate matrix WH :

$$WH = \begin{bmatrix} w_1^1 & \dots & w_{n+1}^1 \end{bmatrix} \quad (12)$$

Those connections are now permanently frozen.

The following equation describes the correlation formula, denoted S_C , between the candidate unit's value and the residual output error observed at the first unit.

$$S_C = \sum_{i=1}^m \left| \sum_{p=1}^{Np} (z_{n+2,p} - \bar{z}_{n+2}) (E_{ip} - \bar{E}_i) \right| \quad (13)$$

where E_{ip} is the residual error $E_{ip} = y_{ip} - t_{ip}$ calculated with the outputs y_{ip} from the previous step. Strictly speaking, S_C , is actually a covariance, not a true correlation because the formula leaves out some of the normalization terms.

The gradient of S_C with respect to the w_l can again be calculated analytically and is supplied to the optimizer to speed up the process.

$$\frac{\partial S_C}{\partial w_l} = \sum_{i=1}^m \left[\text{sgn} \left(\sum_{p=1}^{Np} (z_{n+2,p} - \bar{z}_{n+2}) (E_{ip} - \bar{E}_i) \right) \times \left(\sum_{p=1}^{Np} (E_{ip} - \bar{E}_i) \left[\frac{\partial z_{n+2,p}}{\partial w_l} - \frac{\partial \bar{z}_{n+2}}{\partial w_l} \right] \right) \right] \quad (14)$$

where

$$\frac{\partial z_{n+2,p}}{\partial w_l} = \sigma' \left(\sum_{i=1}^{n+1} w_i z_{ip} \right) z_{lp} \quad (15)$$

$$\frac{\partial \bar{z}_{n+2}}{\partial w_l} = \frac{1}{Np} \sum_{k=1}^{Np} \frac{\partial z_{n+2,k}}{\partial w_l} \quad (16)$$

and

$$\sigma'(x) = \partial \sigma / \partial x \quad (17)$$

is the derivative of the activation function (sigmoid).

3.3.5 Step 5: Connect First Hidden Unit to Outputs

Once trained, the new hidden unit is connected to the outputs with the weights saved in matrix WH. The output $z_{n+2,p}$ is now fixed; it acts as an additional input to the NN. The NN equation can be written as:

$$\begin{bmatrix} y_{1p} \\ \vdots \\ y_{mp} \end{bmatrix} = \begin{bmatrix} v_{11} & \cdots & v_{1,n+2} \\ \vdots & & \vdots \\ v_{m1} & \cdots & v_{m,n+2} \end{bmatrix} \begin{bmatrix} z_{1p} \\ \vdots \\ z_{n+2,p} \end{bmatrix} \quad (18)$$

A schematic of the connections is represented in Fig. 4, the weights that connect the input-to-outputs weights and the first HU-to-output weights are still unknown and must be trained in Step 6.

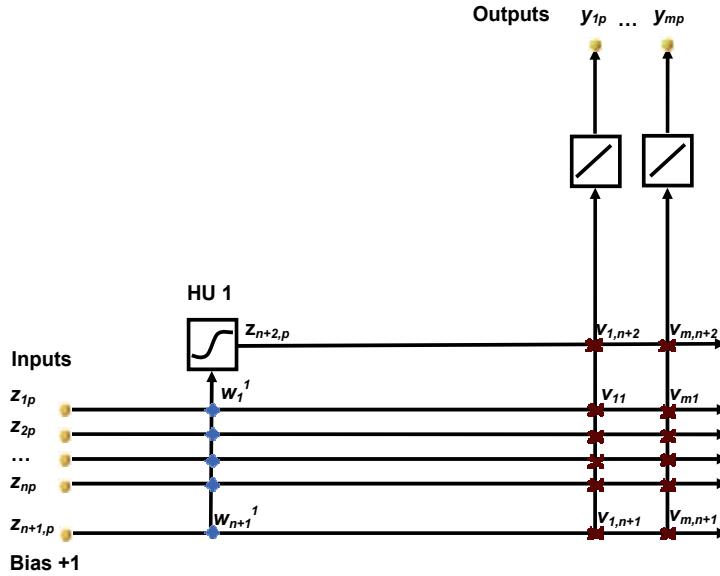


Fig. 4. Schematic of first hidden unit connected to the outputs. Diamond connections are permanently frozen. X connections must be trained again.

3.3.6 Step 6: Minimize Squared Error with first Hidden Unit

Once the new HU is installed on the network, the matrix V must be optimized to minimize the squared error on the TS using the BFGS algorithm from DOT.

The squared error is calculated as:

$$Es = \frac{1}{2} \sum_{i=1}^m \sum_{p=1}^{Np} \|y_{ip} - t_{ip}\|^2 = \frac{1}{2} \sum_{i=1}^m \sum_{p=1}^{Np} \left[\sum_{j=1}^{n+2} v_{ij} z_{jp} - t_{ip} \right]^2 \quad (19)$$

The gradient, which is also supplied to the optimizer, is given by:

$$\frac{\partial Es}{\partial v_{kl}} = \sum_{p=1}^{Np} \left[\sum_{j=1}^{n+2} v_{kj} z_{jp} - t_{kp} \right] z_{lp} \quad (20)$$

for \$k \in \{1 \dots m\}\$ and \$l \in \{1 \dots n+2\}\$.

Weights are initialized by taking former weights calculated above for \$j \in \{1 \dots n+1\}\$ and set to zero for \$j = n+2\$. Indeed, the first \$n+1\$ columns of the V matrix represent the input-to-output weights. Those have already been adjusted at Step 2 to minimize the squared error. It is therefore expected that a good initial guess for the solution with the additional HU in the network is to take the former weights calculated at Step 2 and to set to zero the weights that connect the new hidden unit to the outputs. These weights correspond to column \$n+2\$ in the weights matrix V .

After the squared error is minimized for the training set, it is next evaluated on the validation and the generalization set if they have been specified. Either the epsilon stopping

criterion or one of the early stopping criteria can be chosen. (see Section 0). The chosen criterion for stopping is checked. If it is met the program stops. If it is not met, the program continues adding hidden units one at a time.

3.3.7 Step 7: Connect h^{th} Hidden Unit to Inputs

Each time a new hidden unit is added, a link from this neuron to all the inputs (and bias) and the former hidden units is created (see Fig. 5).

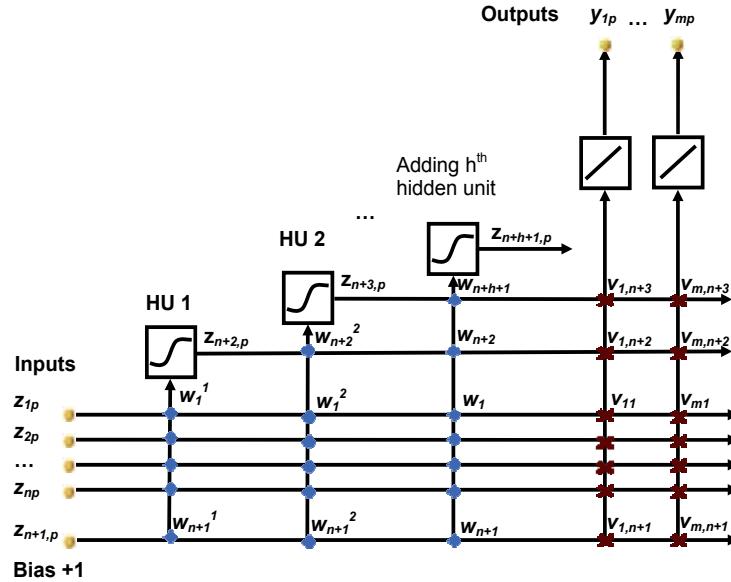


Fig. 5. Schematic after adding h^{th} hidden unit. Hidden unit connected to former HUs and inputs only. Diamond connections to the h^{th} hidden unit only must be trained (w_j weights)

The equation for the output of the h^{th} hidden unit added on to the NN, $z_{n+1+h,p}$, is:

$$z_{n+1+h,p} = \sigma \left(\sum_{j=1}^{n+h} w_j z_{jp} \right) \quad (21)$$

The z_{jp} are the inputs to the network for $j=1 \dots n+1$ and the outputs from the $h-1$ previous hidden units for $j=n+2, \dots, n+h$. Since the input-to-hidden unit weights are frozen for the $h-1$ previous HUs, they can be viewed as additional inputs to the network. The weights w_j from the inputs and the previous HUs to the h^{th} hidden unit are unknown and must be adjusted.

3.3.8 Step 8: Maximize Correlation Formula for h^{th} Hidden Unit

Next, the w_j weights are adjusted to maximize the correlation formula. Again a pool of candidate units is created by initializing the weights w_j for each candidate at random and Step 4 is repeated. The candidate whose correlation is the highest is kept in memory as w_j^h . Those weights are saved on row h of the matrix \mathbf{WH} . Each line of this matrix contains the

weights w_j^i saved for each HU. Its dimension is increased by one row and one column each time a new HU is added.

$$\mathbf{WH} = \begin{bmatrix} w_1^1 & \cdots & w_{n+1}^1 & 0 & \cdots & 0 \\ w_1^2 & \cdots & & w_{n+2}^2 & \ddots & \vdots \\ \vdots & & & \ddots & & 0 \\ w_1^h & \cdots & & & \cdots & w_{n+h}^h \end{bmatrix} \quad (22)$$

Again, the equation for the correlation can be written as:

$$S_c = \sum_{i=1}^m \left| \sum_{p=1}^{N_p} (z_{o,p} - \bar{z}_o) (E_{ip} - \bar{E}_i) \right| \quad (23)$$

and

$$z_{o,p} = z_{n+1+h,p} \quad (24)$$

For simplicity, in the equations the output to the h^{th} hidden unit is denoted $z_{o,p}$ instead of $z_{n+h+1,p}$. Also as before, E_{ip} is the residual error $E_{ip} = |y_{ip} - t_{ip}|$ calculated with the outputs y_{ip} for the network with $h-1$ hidden units.

The gradient of S_C is given by

$$\frac{\partial S_C}{\partial w_l} = \sum_{i=1}^m \left[\operatorname{sgn} \left(\sum_{p=1}^{N_p} (z_{o,p} - \bar{z}_o) (E_{ip} - \bar{E}_i) \right) \times \left(\sum_{p=1}^{N_p} (E_{ip} - \bar{E}_i) \left[\frac{\partial z_{o,p}}{\partial w_l} - \frac{\partial \bar{z}_o}{\partial w_l} \right] \right) \right] \quad (25)$$

where

$$\frac{\partial z_{o,p}}{\partial w_l} = \sigma' \left(\sum_{i=1}^{n+h+1} w_i z_{ip} \right) z_{lp} \quad (26)$$

$$\frac{\partial \bar{z}_o}{\partial w_l} = \frac{1}{N_p} \sum_{k=1}^{N_p} \frac{\partial z_{o,k}}{\partial w_l} \quad (27)$$

3.3.9 Step 9: Connect h^{th} Hidden Unit to Outputs

The candidate HU with the highest correlation is added on to the network and connected to the output (see Fig. 6). The matrix \mathbf{V} connects the inputs and all hidden units installed to the network to the outputs. The outputs can be calculated with the following equation.

$$\begin{bmatrix} y_{1p} \\ \vdots \\ y_{mp} \end{bmatrix} = \begin{bmatrix} v_{11} & \cdots & v_{1,n+1+h} \\ \vdots & & \vdots \\ v_{m1} & \cdots & v_{m,n+1+h} \end{bmatrix} \begin{bmatrix} z_{1p} \\ \vdots \\ z_{n+1+h,p} \end{bmatrix} \quad (28)$$

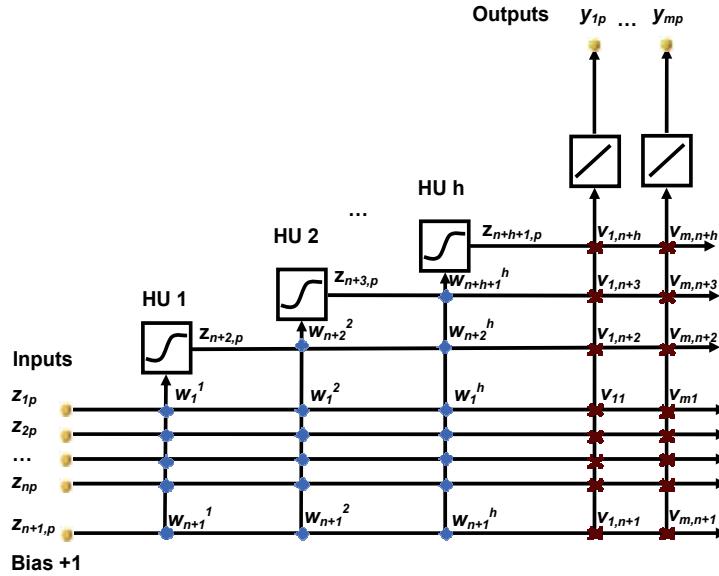


Fig. 6. Connecting h^{th} hidden unit to the outputs. X connections must be trained again.

3.3.10 Step 10: Minimize Squared Error for h^{th} Hidden Unit

Similarly to Step 6, weights v_{ij} are initialized by taking former weights calculated for the NN with $h-1$ HUs for columns $j \in \{1, \dots, n+h\}$ and set to zero for column $j = n+1+h$. The squared error is calculated over the training set and is minimized using the BFGS algorithm.

The squared error and its gradient are both supplied to the optimizer, their equations are:

$$Es = \frac{1}{2} \sum_{i=1}^m \sum_{p=1}^{N_p} \|y_{ip} - t_{ip}\|^2 = \frac{1}{2} \sum_{i=1}^m \sum_{p=1}^{N_p} \left[\sum_{j=1}^{n+1+h} v_{ij} z_{jp} - t_{ip} \right]^2 \quad (29)$$

and

$$\frac{\partial Es}{\partial v_{kl}} = \sum_{p=1}^{N_p} \left[\sum_{j=1}^{n+1+h} v_{kj} z_{jp} - t_{kp} \right] z_{lp} \quad (30)$$

for $k \in \{1, \dots, m\}$ and $l \in \{1, \dots, n+1+h\}$.

3.3.11 Step 11: Stop Training when Stopping Criterion is met

Steps seven through ten are repeated and the cascade correlation algorithm stops when the stopping criterion is met. Several stopping criteria are available to the user in the modified algorithm. The first one, the epsilon stopping criterion, is used in Fahlman's original algorithm (Fahlman & Lebiere, 1990). However, this criterion does not prevent overfitting and therefore, was later replaced by the early stopping criteria as described below.

The Epsilon Stopping Criterion

The epsilon stopping criterion stops the algorithm when the square error on the training set has reached a predetermined ϵ value. The problem with this criterion is that the user must determine in advance which ϵ value to use. Also the squared error can vary significantly

from one function to another. The average value of the outputs changes the error as defined by Eq. 1. Also the number of points used in the training will change the value of the error. So, this ϵ value should be adjusted by the user manually every time a network needs to be trained. Also this stopping criterion does not give any information on the generalization ability of the network, i.e. how the network performs for points not in the training set. This criterion is thus replaced by the early stopping criteria described in the next subsection, which monitor the error on an unseen dataset, the validation set.

The Early Stopping Criteria

As alluded to earlier, these criteria allow to limit overfitting of the network by checking how the error decreases on the validation set. It is commonly known that as more units are added, the network is able to fit training data better since additional degrees of freedom are added. However, the error on data not used during training decreases at first but then later increases, showing signs of overfitting or overtraining. The idea of early stopping is to stop training early, before full convergence of the network on the training set, or when the error on the unseen dataset, the VS, is minimum. However in order to find the minimum of the error on the VS, one must continue training the network some time past this minimum and then stop and choose the network with the number of hidden units which correspond to that minimum error. This leads to several stopping criteria to decide how long to continue training after a minimum of the error is found.

Three classes of stopping criteria are available in the MCC. They are described in detail in Schmitz (2007) and are not repeated here. The criterion used in both applications described in this chapter is the PQ0.75 criterion. This criterion is relatively efficient and accurate in finding the true minimum error on the VS.

A few definitions are required before the PQ criterion can be derived. The squared error calculated on the TS for h hidden units added on the network will be noted $Es_{TS}(h)$ and called training set error at epoch h , or training error for short. The epoch h corresponds to a network trained with h hidden units, with h varying from 0 to the maximum number of hidden units (namely 70 in the MCC). $Es_{VS}(h)$, the validation error, is the corresponding error on the VS. Let $Es_{OPT-VS}(h)$ be the lowest validation error obtained in epochs up to h :

$$Es_{OPT-VS}(h) = \min_{h' \leq h} Es_{VS}(h') \quad (31)$$

The generalization loss ($GL(h)$) at epoch h is defined as the relative increase of the validation error over the minimum so far, in percent:

$$GL(h) = 100 \cdot \left(\frac{Es_{VS}(h)}{Es_{OPT-VS}(h)} - 1 \right) \quad (32)$$

The training progress, denoted $P_k(h)$, measures how large the average training error is during a training strip of length k (epochs from $h-k+1$ to h) with respect to the minimum error during that same strip.

$$P_k(h) = 1000 \cdot \left(\frac{\sum_{h'=h-k+1}^h Es_{TS}(h')}{k \cdot \min_{h-k+1 \leq h' \leq h} Es_{TS}(h')} - 1 \right) \quad (33)$$

The $PQ0.75$ criterion can be defined as following: training stops when the quotient of generalization loss and progress exceeds a threshold $\alpha=0.75$, such that:

$$PQ(h) = \frac{GL(h)}{P_k(h)} > 0.75. \quad (34)$$

Note that this criterion is only checked after every end-of-strip epoch h , where the length of the strip is k epochs. In the following study we will always assume length of strips $k=5$.

When the criterion is met, the training is stopped at some value of h and the resulting set of weights is the one that corresponds to the lowest validation error $E_{SOPT-VS}(h)$. So, the corresponding network usually has a number of hidden units $h' < h$. Note that the criterion does not ensure stopping, so a large maximum number of hidden units ($h_{max} = 70$) was chosen to avoid training indefinitely.

3.3.12 Resulting Single Network

Once the stopping criterion is met, the algorithm stops. If the epsilon criterion is used, the resulting network is the last trained. If one of the early stopping criteria is used, the program chooses the network with the number of hidden units corresponding to the minimum validation error. The program keeps several matrices in memory: $WH=[w_{ij}^h]$, the weights between inputs and each hidden unit saved after adding each unit is added to the NN. Also, it keeps two sets of v_{ij} weights; the one between inputs, plus all hidden units and outputs after the last hidden unit has been added to the network and the set of v_{ij} weights which correspond to the minimum validation error (if needed).

The function, approximating f is thus given by the equations below and can be evaluated by recurrence, so that:

$$\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} v_{11} & \cdots & v_{1,n+1+h} \\ \vdots & \ddots & \vdots \\ v_{m1} & \cdots & v_{m,n+1+h} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \\ +1 \\ u_1 \\ \vdots \\ u_h \end{bmatrix} \quad (35)$$

$$\begin{bmatrix} u_1 \\ \vdots \\ u_h \end{bmatrix} = \sigma \begin{bmatrix} w^1_1 & \cdots & w^1_{n+1} & 0 & \cdots & 0 \\ \vdots & & \ddots & & & \vdots \\ w^h_1 & \cdots & \cdots & \cdots & w^h_{n+h} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \\ +1 \\ u_1 \\ \vdots \\ u_h \end{bmatrix} \quad (36)$$

where x_l ($l \in \{1, \dots, n\}$) are inputs to the neural network, y_i ($i \in \{1, \dots, m\}$) outputs and u_k ($k \in \{1, \dots, h\}$) intermediate states, calculated by recurrence. Also, h is the number of hidden

units corresponding to the last hidden unit added to the network, if the epsilon stopping criterion is used, or the one corresponding to the minimum validation error.

Note that the network has been trained with inputs normalized according to the TS and outputs rescaled so that the TS average is one. Therefore, the datapoint \mathbf{X} , must be linearly transformed before evaluating the output of the network according to the following equation.

$$\mathbf{X} = \begin{bmatrix} x_{1p} \\ \vdots \\ x_{np} \end{bmatrix} = \begin{bmatrix} \frac{z_{1p} - \text{MinInput}_1}{\text{MaxInput}_1 - \text{MinInput}_1} \\ \vdots \\ \frac{z_{np} - \text{MinInput}_n}{\text{MaxInput}_n - \text{MinInput}_n} \end{bmatrix} \quad (37)$$

where MinInput_i and MaxInput_i are the minimum and maximum values obtained from Eq. 3 and 4.

When using the early stopping criterion, several networks are trained sequentially using the same TS and VS. Because the weight surfaces have many local minima and maxima and the weights are initialized at random, the networks will all be different. Only the network which leads to the smaller validation error is retained.

3.3.13 Resulting Ensemble Network

If ensemble averaging is chosen, all networks built are kept in memory and the output of the committee network is taken as the average output of each individual network. A simple average according to

$$\mathbf{Y}_{\text{Ensemble_NN}} = \frac{1}{M} \sum_{k=1}^M \mathbf{Y}^{(k)} \quad (38)$$

is used to calculate the prediction ability of the ensemble. The output $\mathbf{Y}_{\text{Ensemble_NN}}$ is average of output values of each individual network $\mathbf{Y}^{(k)}$ where $k=1,\dots,M$ is the index of the network belonging to the ensemble.

4. Application to Fast Ship Multi Disciplinary Design Optimization

This section describes an MDO optimization of an underwater hull configuration. The optimization is performed using both a “classical approach”, in which the CFD analysis is integrated directly *inside* the optimization loop, and an “NN approach”, in which the CFD analyses are used for TS generation, i.e. *outside* of the optimization loop.

4.1 Design Problem Description

The problem consists of optimizing one of Pacific Marine’s advanced lifting bodies. This patented underwater hull is made of two displacement bodies, called *H-bodies*, linked with a thin foil, referred to as *cross-foil*, and attached to the ship by two struts as shown in Fig. 7. The entire arrangement is referred to as the *twin H-body* configuration and can be fitted to

catamaran or pentamaran hull forms. The displacement bodies are designed to provide good sea-keeping properties at lower speeds when the hulls of the catamaran or pentamaran are partially submerged, while the cross-foil is designed to provide additional lift at higher speed when the multiple hulls are lifted out of the water in order to reduce drag.

A very similar configuration was optimized under a previous work reported by Hefazi *et al.* (2002), using the classical optimization process. Sea trials of a half-scale replica of this configuration on a 44 ft test platform were conducted to provide data for the validation and demonstrate the application of lifting body technology on a real test platform (Hefazi *et al.*, 2003). The resulting optimized geometry was integrated into the first US-built "fast ship", the HDV-100 technology demonstrator (Fig. 8).

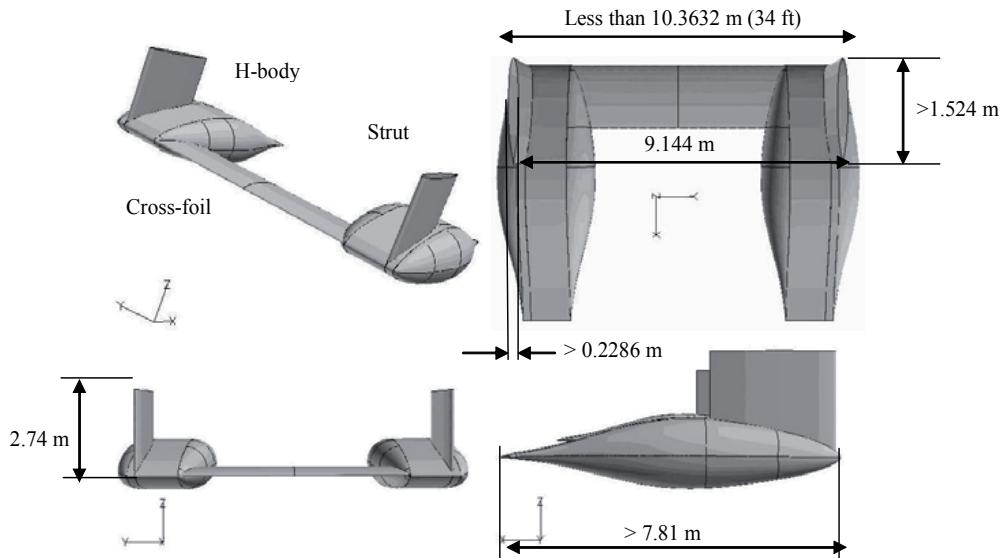


Fig. 7. Twin H-body configuration, baseline



Fig. 8. Blended-wing-body configuration sketch and picture of the HDV-100 technology demonstrator (shown here at low speed)

The optimization is performed for a boat speed of 47 knots (24.2 m/s). At this velocity, the parent hull will run dry. The whole configuration must be able to generate a total lift of 80 LT, including a minimum of 16 LT by displacement effect.

The objective is to maximize range, i.e. lift-to-drag ratio, which corresponds to minimizing drag at constant lift. In addition, the configuration is to be designed such that it can operate cavitation free at 52 knots. Additionally, the operating draft is to be 2.74 m (waterline to lowest point - 9 ft) and a structural constraint is to be imposed to prevent yield of the cross-foil. Also, the maximum width of configuration shall not exceed 10.36 m (34 ft) and the strut-to-strut distance is fixed by requirements for mating to the upper hull.

From an optimization problem point of view, the objective function and design constraints can be summarized as:

- Objective: Maximize lift-to-drag (LOD)
- Constraints:
 - Operational speed of 47 knots (i.e. Reynolds number is 211.63×10^6 based on a reference chord of 8.69 m in 75 F Hawaii waters)
 - Cavitation free at 52 knots, i.e. $C_p > -0.269$ (Hawaii water at 75 F)
 - Total lift (TL) equal to 80 LT
 - Displacement or buoyant lift (BL) greater than 16 LT
 - 2.74 m (9 ft) operating draft
 - Strut centerline to strut centerline distance is fixed at 9.14 m (30 ft) for mating with upper hull
 - Overall beam length should not exceed 10.36 m (34 ft)
 - Minimum strut thickness 0.2286 m (0.75 ft)
 - Minimum strut chord of 1.524 m (5 ft)
 - The cross foil should have the structural integrity, not to yield using a material of yield strength of 344.7379 Mpa (50 ksi), assuming solid section.
 - Displacement pod length should not exceed by more than 20% the parent body (defined as optimized configuration (Hefazi, 2002)), i.e. 7.81 m.

4.2 Classical Optimization

4.2.1 Objectives, Constraints and Optimization Implementation

Nowadays, all design engineers are accustomed to designing their vehicles using some computer aided design (CAD) or solid modeling package, such as Pro-Engineer, CATIA, UniGraphics, IDEAS. In addition, the designer usually represents the configuration by a set of parameters, or design variables, which can be varied to improve the design by linking the CAD software with an appropriate analysis module. This approach is routinely used for structural design, for example, by linking the CAD software with a finite element (FE) method. Such an approach is not yet routinely used, however, in the case of hydrodynamic shape optimization, because additional challenges face the designer. Among these issues are

- The cost and accuracies associated with flow analysis using CFD
- The grid requirements for CFD methods

In order to address the issue of CFD shape optimization, one must be able to automatically vary the shape of various elements of the configuration in the CAD method, generate a mesh of sufficient quality for the CFD method, and use an efficient and accurate CFD method to obtain the hydrodynamic performance of the configuration being analyzed. The

driver in the selection of the components of the CFD optimization method is the ability to link these tools together in an automated fashion, without user intervention, while ensuring that the flow analysis is both efficient and accurate for the problem at hand. For this reason, several options for each tool were considered and the following set of tools was selected:

- CAD software: *Pro-Engineer* (Parametric Technology Corporation, 2009)
- Grid generation software: *ICEM CFD* (ANSYS, Inc., 2009)
- CFD software: CSULB-developed interactive boundary layer (IBL) approach with free surface modeled by negative images (Besnard, 1998, and Hefazi *et al.*, 2002)
- Optimization software: *iSIGHT* (Dassault Systèmes SIMULIA, 2009)

Pro-Engineer and *ICEM CFD* were selected because of the existence of a module which does allow for automatic data transfer between the CAD and grid generation package. The IBL approach was chosen because at high Reynolds numbers and low angles of attack, it is a very accurate and efficient approach. In addition, because of the large Froude number for the case at hand, the free surface can be modelled with negative images. Finally, the numerical optimizer *iSIGHT* offers an easy to use platform for the optimization and/or design of experiments. The method, controlled by *iSIGHT*, integrates the different software packages with several scripts, which, once set up, performs all tasks automatically, without user intervention. This automatic setup is critical in the optimization process. *iSIGHT* controls the process and calls the various scripts;

- Define and generate the new geometry (*Pro-Engineer*);
- Check for any constraint violation (from output of *Pro-Engineer*);
- Generates a mesh suitable for the CFD method (*ICEM CFD*);
- Executes the CFD method (IBL code); and
- Extracts the data needed by *iSIGHT* (objective function and constraint values) and calculate the constrained objective function for the next iteration.

The process implemented for the Twin-H body optimization is shown in Fig. 9. The configuration is represented by a total of 28 design variables which control the size and shape of each component, which, once assembled, describe the entire configuration. The first step involves generating a geometrically feasible configuration from the selected set of design variables. Simple geometrical parameters, such as width, length, chord, etc., are used to characterize the elements. Foil cross-sections are defined by their mean camber line and thickness distributions. Camber line is parameterized by the classical NACA two-parameter set. The thickness distribution, y_{th} , is represented by a 6-deg. polynomial, with an additional term, a_0 , for controlling the leading edge radius:

$$y_{th}(x) = a_0 \sqrt{x} + \sum_{i=1}^6 a_i x^i, \quad 0 \leq x \leq 1 \quad (39)$$

This configuration is automatically generated by the CAD-based solid modeler, *Pro-Engineer*, based on the 28 design variables using scripts. This process involves two parts:

- Airfoil shape definitions ("shape.in" files)
- Configuration parameterization ("ptr" files)

Several independent scripts using the different "shape.in" files are used to regenerate updated *Pro-Engineer* files ("ptr" files). Then, *Pro-Engineer* automatically updates the geometry based on the revised data.

In the second step, constraints which may be determined from the newly generated solid model, such as volume, structural constraint, etc., are evaluated. In third step, a suitable mesh is automatically generated using *ICEM CFD*. This mesh is then used along with the CFD input data file to execute the CFD code. The CFD tool used here makes use of the high Froude and Reynolds number approximations by employing a viscous-inviscid approach. The inviscid flow is solved by a higher order panel method with the free surface effects modeled by negative images and the viscous flow is solved using an inverse boundary layer approach which can treat large regions of flow separation. Viscous and inviscid methods are coupled using the blowing velocity/displacement concept and leads to accurate pressure, lift and drag predictions at minimal costs for this type of configuration and flow conditions (see, e.g., Hefazi *et al.* 2002). Hence, only a surface mesh is needed. The use of a Reynolds averaged Navier-Stokes (RANS) method would require the use of a volume mesh which could also be implemented with the tools used here (*ICEM CFD*).

The last step involves the use of a constrained objective function, f_c . Because a global optimization method is used (genetic algorithm), the objective function and constraints are integrated into a single “constrained objective function” which is to be minimized. The constrained objective function is such that, when at least one constraint is strongly violated, f_c is set close to f_{max} . f_{max} is typically on the order of one and corresponds to a normalized value of the maximum objective function. The normalization value is given by the user and is typically chosen at the higher values of expected f over the search space. For example, with an optimization where L/D is the objective function on the order of 10-15, a normalization value of 10 to 20 can be chosen. Choosing 10 would mean that f_c might reach 1.5.

Two positive parameters ε_1 and ε_2 are now defined. If $\forall i, g_i \leq -\varepsilon_1$, then the constrained objective function is f . If $\exists i \mid g_i \geq \varepsilon_2$, then the penalized cost function gets close to f_{max} depending on how the constraints are violated. In other words, ε_1 decides when constraints become active, and ε_2 when they become prohibitive.

The generalized constraint G is defined as

$$G(x) = \frac{1}{n_{con}} \left(\sum_{g_i \geq -\varepsilon_1} g_i(x) + \varepsilon_1 \right) \quad (40)$$

and the constrained objective function becomes

$$f_c(x) = [1 - \varphi(y)] \cdot f(x) + \varphi(y) \cdot f_{max} \cdot \left(1 - \frac{e^{-y}}{2} \right) \quad (41)$$

where

$$\varphi(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ e^{-1/x^2} & \text{if } x > 0 \end{cases} \quad (42)$$

and

$$y = \frac{10G(x)}{(\varepsilon_1 + \varepsilon_2)} \quad (43)$$

For the NN based optimization, the generation of the training and validation sets is performed using the same approach, except that instead of using iSIGHT setup to run an optimization (genetic algorithms in the present case), it is designed to run Latin Hypercube samplings of the desired TS, VS and GS sizes.

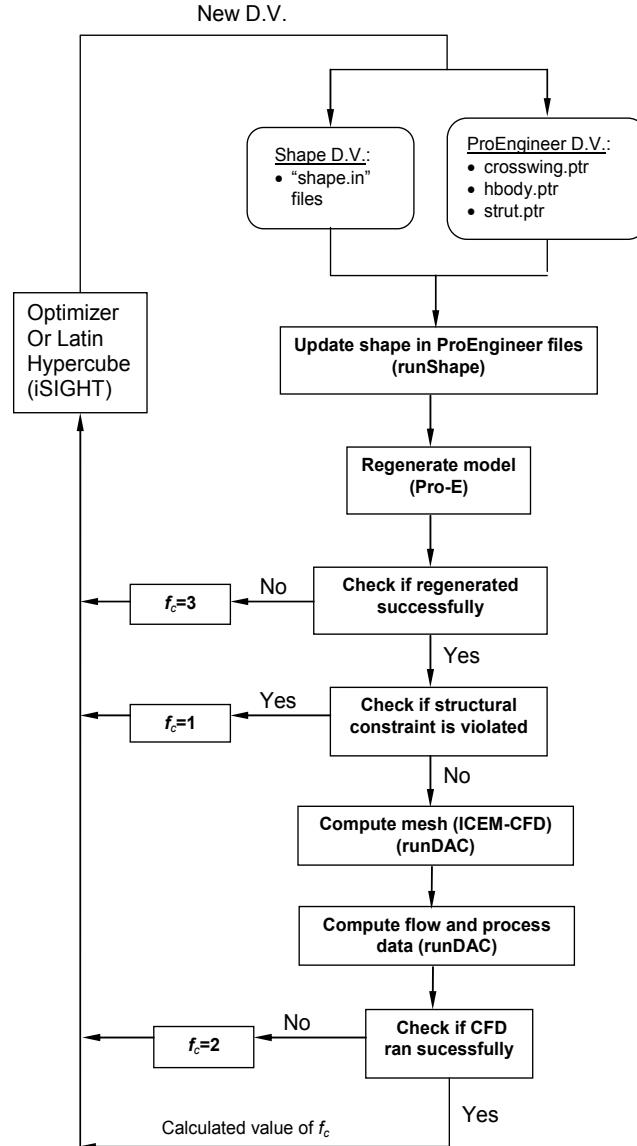


Fig. 9. Optimization with the classical approach. This same loop is used for training/validation set generation but with Latin Hypercube sampling instead of optimization.

4.2.2 Classical Optimization Results

A genetic algorithm optimization was run over the 28-dimension design space for 5000 iterations. The outcome was a configuration with an L/D of 12.92, which represents a 26 percent improvement in L/D over the baseline design. A comparison of the performance of the baseline and optimum configurations is shown in Table 1. Note that the number of iterations used is small for a global search problem with 28 design variables, but it was limited to 5000 because of time requirements. One iteration takes about 10 min. to run on an Origin 3200 server. 5000 runs correspond to over a month of CPU time.

	Baseline	Optimum	Objective
Buoyant lift	18.4 LT	16.1 LT	> 16 LT
Total lift	80.2 LT	81.6 LT	= 80 LT
Cpmin	-0.263	-0.250	> - 0.269
LOD	10.23	12.92	Maximum

Table 1. Performance of optimized vs. baseline configuration (28 design variables)

4.3 Neural Network Optimization Approach

The use of the neural network (NN) approach encompasses several steps:

- Generation of the training set (TS) & validation set (VS)
- NN training to obtain a NN “evaluator(s)”
- Optimization with the NN evaluator(s)

The first two steps are explained in detail in the next subsections. The third step is essentially the same as the classical optimization with the CFD code replaced by the neural network, and thus is not repeated here.

The optimization approach specific to the twin H-body optimization problem calls for the use of five single output neural networks as shown in Fig. 10, one for the objective function and the others for the constraints: lift-to-drag ratio (LOD), minimum pressure (Cpmin), dynamic lift (DL), buoyant lift (BL) and maximum stress value (Struc). Alternatively, a single NN with five outputs could have been used, but typically, the resulting network is much more complex than five individual networks (has more weights). Hence, it takes more time to generate, i.e. train, and usually needs a larger training set than five single output networks to generalize well. Also the five networks can be trained in parallel on a multiprocessor machine, reducing the training time even more.

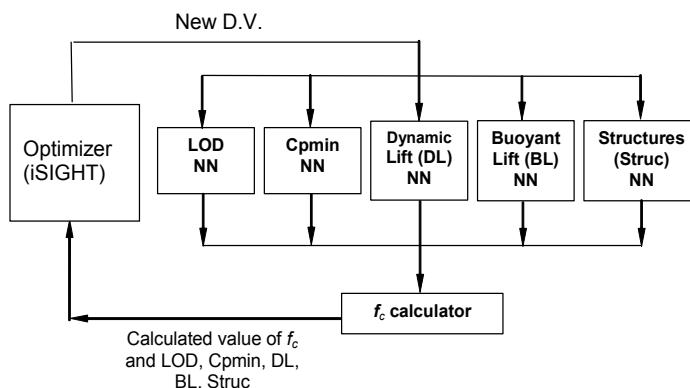


Fig. 10. Optimization process with NN approach

Also, although the optimization is performed on the constrained objective function, f_c , it was not represented directly by a single NN because it would have required a very good training set definition in the small regions of the design space where the design was feasible. Instead, using five networks allows for accurate representation of each function over the design space and thus improved predictions for f_c .

4.3.1 Training and Validation Sets

For the present analysis, a validation set (VS) of 300 points was generated with the iSIGHT setup of Fig. 9 and using the same 28 design variables as for the classical approach, but using a Latin Hypercube instead of an optimization algorithm. Two training sets (TS) were also generated using the same process, one with 1000 points, the second with 2000 points. A third set with 5000 points was also generated to be used as the Generalization Set (GS). Although a practical application may not involve the generation of multiple TS and certainly not a GS, this was done here to analyze the effect of TS size on the result quality. For each size of Latin Hypercube sampling in the design space, approximately 20 to 25 percent of the points were geometrically unfeasible (i.e. the model could not be successfully reconstructed) and had to be removed from the training sets. Nevertheless, the VS, TS and GS will be referred to as 300-VS, 1000-TS, 2000-TS and 5000-GS subsequently. The exact sizes for each set are respectively, 228, 808, 1587, and 3852.

4.3.2 Neural Network Training

Each neural network (NN) was trained with 28 inputs (or design variables) and one output, i.e. one for each function: buoyant lift (BL), Cpmmin, dynamic lift (DL), lift-to-drag ratio (LOD), structural constraint (Struc) and total lift (TL). For each function, 10 NNs were built and the one that had the best validation error was chosen. Typically, training each network takes from about an hour (for 1000-TS) to a day for the more complete data sets (5000-GS). Unlike CFD methods which are demanding in computing power (I/O, memory), training demands relatively little other than CPU time and thus, all training can be done in parallel on the same server without interfering with other ongoing computations. This feature is yet another advantage compared to training a single multiple-output network.

The results for the different TS did not vary much from one to the other and are presented here for the 1000-TS. Also, in order to evaluate the quality of the training and compare the error on the 300 points VS. The GS of 5000 points was evaluated on the trained NN. Table 2 shows the average errors and standard deviations over the TS, VS and GS. Errors and standard deviations are adequate for the problem at hand. For example, an average error of 0.04 is expected for the displacement (BL) which has a value on the order of 18 and the corresponding standard deviation is also 0.04.

	Typical	\bar{E}_{TS}	\bar{E}_{VS}	\bar{E}_{GS}	$std(E)_{TS}$	$std(E)_{VS}$	$std(E)_{GS}$
BL (LT)	18	0.0275	0.0381	0.0419	0.0223	0.0314	0.0427
Cpmmin	-0.269	0.0055	0.0065	0.0068	0.0066	0.0071	0.0080
DL (LT)	60	0.1607	0.1847	0.1855	0.2002	0.1420	0.1874
LOD	12	0.2270	0.2602	0.2646	0.1775	0.2074	0.2054
Struc (MPa)	300	4.05	5.3	5.3	3.7	5.2	5.1

Table 2. Average errors and standard deviations on TS, VS and GS for a 1000-pt. TS and a 5000-pt. GS for the 28-design variable Twin H-body optimization

Also, average errors and standard deviations on the VS and GS are in excellent agreement, thus validating the VS approach despite the number of points used for the design space of 28 dimensions.

4.3.3 Neural Network Optimization Results

The neural networks generated in the form of five executables for Cpmmin, LOD, DL, BL and Struc using the different sizes of training sets (1000-TS and 2000-TS) were integrated in iSIGHT as shown in Fig. 10. A genetic algorithm (GA) optimization was used with an initial population of 50 and 35000 iterations were run based on the constrained objective function defined in Section 0.

The best 100 runs resulting from the optimization with the five NNs (best f_c) were then run using the Pro-Engineer model and the CFD code to compute the objective function and constraints and compare them with those of the NN near the optimum. Also, a few results from the optimization (i.e. as determined by the NN) with a slightly higher LOD than that of the best f_c but with constraints closer to their limit (therefore resulting in a lower f_c) were chosen and also run through the CFD package. A summary of the results is shown in Table 3. For each size of training set, the table shows:

- The best f_c as determined by the optimizer (i.e. after 35000 GA iterations using the NN)
- The best LOD based on the NN with minimal constraint violations: corresponds to some hand-picked results from the optimization showing a slightly better LOD but with constraints close to the acceptable limits resulting in a higher f_c
- The best f_c based on CFD results from the 100 best NN points (as determined by the GA)
- The best LOD based on CFD results from the 100 best NN points (as determined by the GA)

In each case, buoyant lift (BL), total lift (TL), lift-to-drag ratio (LOD) and minimum pressure (Cpmmin) values computed by the NN and the CFD method are shown. The stress value (Struc) is not shown because it exhibited little variations between points and because the constraint was not violated. Also, the dynamic lift (DL) can be directly calculated from total and buoyant lift values (the optimization actually uses DL and BL to determine TL, but the latter is presented because it corresponds to a primitive twin H-body requirement). It should also be noted that the constraints are not implemented as step functions but rather very steep functions which do vary near the constraint border. For example, the primitive requirement calls for a displacement or buoyant lift greater than 16 LT, but as implemented here, a value close to 15 is acceptable. For this reason, f_c may vary in a counter-intuitive fashion, thus rendering the analysis of its values difficult. It is therefore not shown.

The differences between using the NN and the direct CFD computation are within acceptable limits. They are very close in many instances. Also, differences between the values resulting from different selection processes for a particular TS are rather small. Finally, while one observes that the LOD is over-predicted by the NN, the differences between points are about the same for a given TS. Regardless of which TS is chosen, however, LOD is greatly improved from the 12.9 result obtained with the direct CFD method.

TS	Results	Output from NN				Same DV but with CFD			
		BL	TL	LOD	Cpmin	BL	TL	LOD	Cpmin
1000	Best f_c optimized from NN	15.23	80.59	14.39	-0.265	15.24	81.59	13.76	-0.266
	Best LOD optimized from NN	14.75	79.94	14.44	-0.267	14.78	81.04	13.85	-0.268
	Best f_c (using CFD) out of 100 best runs	15.49	80.81	14.30	-0.264	15.51	81.67	13.70	-0.266
	Best LOD (using CFD) out of 100 best	15.25	80.60	14.38	-0.265	15.25	81.64	13.78	-0.265
2000	Best f_c optimized from NN	15.20	80.67	14.49	-0.265	15.13	80.86	13.70	-0.271
	Best LOD Optimized from NN	15.07	80.66	14.51	-0.265	15.01	80.78	13.81	-0.269
	Best f_c (using CFD) out of 100 best runs	15.30	80.74	14.42	-0.263	15.23	80.96	13.71	-0.265
	Best LOD (using CFD) out of 100 best	15.17	80.59	14.47	-0.265	15.11	80.75	13.79	-0.270

Table 3. Results from NN optimization and comparison with CFD

4.4 Comparison between Classical and NN-based Methods

Depending on which design is selected, L/D (LOD) ranges from 13.70 to 13.85, which is a definite improvement from the classical method which lead to 12.92. This improvement is due to the ability to increase the exploration of the design space within the time available in a given design project. For the classical optimization, the genetic algorithm was allowed to run for 5000 iterations, which corresponds to approximately one month of constant calculations on an Origin 3200 server. Because of the use of the CFD tool inside the design loop, CPU time available limited the design space exploration and did not lead to a true optimum. On the other hand, a much larger number of iterations could be performed with the NN approach leading to a greater L/D improvement. In this case, most of the CPU time is taken by the training set generation, with all other computations (training and GA iterations) representing a small fraction of the total CPU time. With as low as 1000 points generated to approximate the various functions over a design space with 28 design variables, an improvement of about 34 percent in L/D is achieved with the NN approach compared with the original baseline twin H-body, this at one fourth the cost needed to get a 26 percent improvement when using the classical approach (with iterations limited because of CPU time constraints).

The results also point to a few improvements which would need to be implemented to address non-differentiable functions (to improve Cpmin predictions, for example), the selection of constrained objective function, and the selection of mathematical optimization method. The latter comment is particularly pertinent for problems in which the optimizer (GA here) would focus its attention in a region of the design space where one (or more) function (objective or constraint) is not approximated as well as might be desired, thus potentially leading to unusable optimization results. One could benefit from having an optimization method which explores several regions of the design space, as illustrated in Lin and Wu (2002).

Results do show, however, that the method can provide its user with a valuable tool for improving designs within a limited time frame and possibly at a lower cost than using

conventional analysis tools integrated in the optimization loop. In this latter case, similarly to the twin H-body configuration optimization presented here, the cost of the analyses limits the number of iterations which can be performed in a reasonable time leading likely to sub-optimal solutions. On the other hand, with the NN approach, a large number of designs can be investigated quickly –almost instantaneously– once a training set has been made available and the NN has been trained.

5. Application to an America's Cup Class Yacht Analysis and Design

In this section, we apply the NN approach to a case where one wishes to use large experimental datasets for detailed analysis and optimization of the performance of a system, here an America's Cup class yacht. This case presents unique challenges associated with the fact that some data is not usable and that the data is usually not uniformly distributed over the design space. The objective here is to use an experimental database obtained with a yacht in at-sea trials to determine the fastest upwind speed the boat can have under prevalent wind conditions from the corresponding boat settings (including keel, rudder, sail, etc.).

5.1 Parameters

During trials, America's Cup teams record their boat performance with very high accuracy to create a true dynamic picture of the boat response under various sailing conditions. This sailing data file recorded by the Wave Technology Processor (WTP) is used here as our experimental database. In our example, each sailing “point” is a vector with a total of 40 parameters. Since the objective here is to optimize the upwind performance of the boat, the projected speed over the course is to be maximized:

$$VMG = Vs \cdot \cos(TWA) \quad (44)$$

where Vs is the measured boat speed, VMG is the Velocity Made Good, and TWA is the true wind angle. This VMG becomes the objective function, and will be the output of the NN.

Of the remaining 39 parameters, eight independent variables have been selected for the training based on their accuracy and their major influence on boat speed. Other dependent variables, parameters with calibration problems or with marginal impact on boat speed, have been discarded. The eight remaining independent variables to be used for the NN training are (with their variable name written in parenthesis):

- Heel Angle (Heel)
- Leeway Angle (Leeway)
- True Wind Angle (TWA)
- True Wind Speed (TWS)
- Trim Tab Angle (Trim Tab)
- Rudder Angle (Rudder)
- Forestay Tension (Forestay)
- Main Traveler Position (Main Traveler)

The sailing database represents several hours of sailing and about 50 percent of that sailing time is the upwind direction, which is our targeted condition. This type of dataset based on

experiments requires using a process to identify the data corresponding to the desired conditions (here upwind) among the entire database and to discard the downwind sailing and all transitional moments like tacking, rounding marks.

5.2 Data filtering

To filter or process the large WTP database several software packages such as Microsoft Access (M.A.) are available. M.A. is able to manipulate large series of data, with a good Microsoft Excel interface to generate plots from the output dataset for verification purposes. The queries technique for M.A. simplifies the experimental dataset filtering process in manipulating with a logical operator the variables and generates a new dataset without altering the original database. Once the queries (point selection criteria) are written, the system is automated to produce a new dataset in an instant from any other experimental database of similar type. M.A. is used first to discard all transitional and non-upwind sailing points. In a second step, since the upwind sailing data is a mix of port and starboard sailing with design variables being either positives or negatives, M.A. is used to convert the points to a single condition, here starboard sailing.

Seven constraints capable of removing and altering the incorrect data are used:

1. $0 \leq \text{absolute [Leeway]} \leq 2$
2. $0 \leq \text{absolute [Trim tab]} \leq 9$
3. $0 \leq \text{absolute [Rudder]} \leq 12$
4. if $[\text{AWA}] > 0$ then $[\text{Main Traveler}]$
5. if $[\text{AWA}] < 0$ then $[\text{Main Traveler}] \times (-1)$
6. $16 < \text{absolute [AWA]} < 40$
7. $[\text{Boatspeed}] / [\text{Vs_target}] \geq 0.6$

where AWA corresponds to the apparent wind angle and Vs_target is the expected boat speed at such conditions.

In our example, the number of points provided by the team was 21,200. The automated filtering system removed about 40% of points from the raw sailing database.

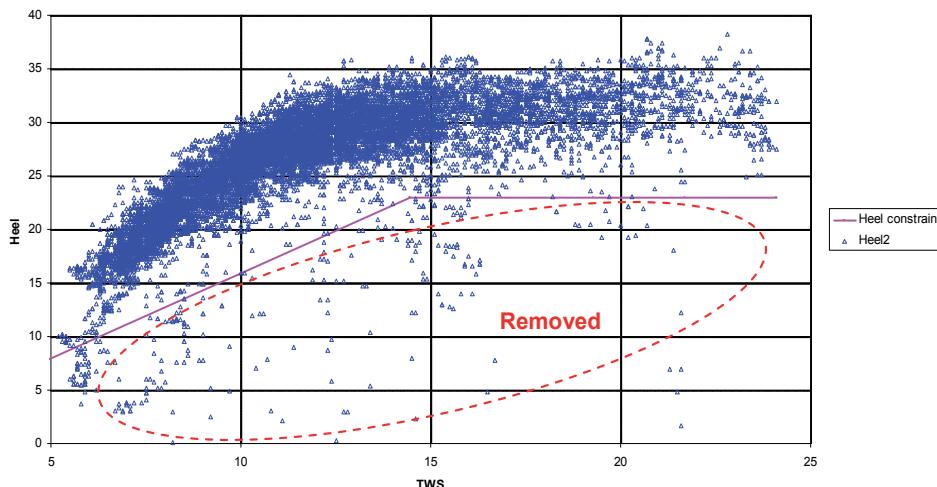


Fig 11. Filtered database shown in terms of Heel vs. TWS resulting from imposing 7 constraints

Fig 11 shows a sample point distribution (here Heel vs TWS) over the design space; the filtered points (shown in blue) are mainly gathered in a dense area but a few points are still “unrealistic” by their location in the plot, away from the main cloud area.

Based on experience, a sailing limit base line (in pink) has been drawn and all records located below the pink limit line are non-valid, non-representative sailing points. They represent 1 to 2 % of initially filtered points and can be easily removed “manually” (non-automated approach).

Similarly, two other verification plots (TWA vs TWS and Forestay vs TWS) were made. The “unrealistic” point percentage is similar to that above, with only about 1% of them needing to be removed manually. This low percentage validates our constraints and filtering procedure. The resulting database has 6,386 points.

5.3 Filling of the design/analysis space

In this analysis, the wind range selected is from 10 to 15 knots. As illustrated in Fig. 12, the experimental data, shown here in terms of Heel and Forestay settings from 10 to 11 knots, is primarily dependent on the TWS values with most data points located in a cluster or “cloud” and areas around the cloud with few or no points. The objective is to have the NN trained to provide the boat performance over a range of operating condition. Since the location of these clusters is not a priori known, it is necessary to have an approach where the NN gives adequate results in the “empty regions” of the space. This is accomplished by filling in the remainder of the space with other points. The technique consists of generating artificial points in areas of low point density (i.e. “fill-in” the space) but with their objective function value equal to some fixed “unattractive” value. Therefore, any NN function evaluation away from the “cloud” area(s) will generate a lower objective function value (if the goal is maximization of the said function, such as boat speed) in the vicinity of the points added to the dataset; preventing the optimizer to search for solutions in that area.

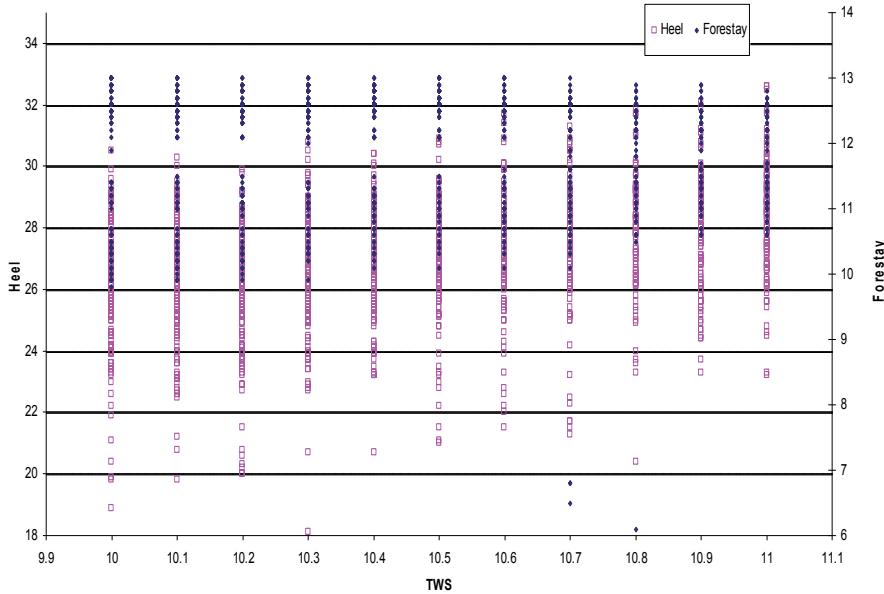


Fig. 12. Heel and Forestay settings variation with TWS between 10 and 11 knots.

The location and the quantity of these added points is the crucial element, as they need to surround the border zone of high density points without perturbing the well-defined zones of the domain, or clouds.

A data point enclosing a number “ n ” of variables is interpreted as a vector of dimension n . In a first step, the additional points are generated randomly over the *entire* domain. In a second step, those that end up *within* a cloud are removed. The selection (decision to keep) each of the randomly generated points is based on geometric considerations. The selection criterion is based on the largest of the distance between closest points of the experimental database, δ . In other words, if the experimental dataset is $(\vec{x}_i)_{1 \leq i \leq N}$ with \vec{x}_i a vector of dimension n , each vector or point has a closest neighbour and the distance between that point and its closest neighbour defined in Euclidian space is δ_i . Therefore, δ can be defined as:

$$\delta = \text{MAX}_{1 \leq i \leq N} (\delta_i) = \text{MAX}_{1 \leq i \leq N} \left(\text{MIN}_{1 \leq j \leq N} \|\vec{x}_i - \vec{x}_j\| \right) \quad (45)$$

Specifically, a randomly added point \vec{y}_i will be discarded if there exists a point \vec{x}_k such that the distance between these two is less than $k.\delta$, where k is a constant to be determined as discussed in the next section.

The result of this process is an experimental database which has been filled in regions away from the “cloud” and usable for generating the NN. The two questions which arise are first, what is the best value to give to k , and second, what value to give to the NN output, or objective function since we use the NN for optimization, at that additional point. Courouble *et al.* (2008) present a detailed analysis which answers these questions and only a summary of the results is presented here.

Since the NN is used for optimization (maximization of upwind boat speed and determination of the corresponding settings), ideally, the result obtained with the NN trained using the database including the automated filling process should be the same as that when an experienced yacht designer would restrict the sailing parameters to “appropriate ranges.” This latter case is used as a *reference case* for comparison with the results obtained from the automated filling process.

5.3.1 Reference case

To preset the correct search domain to be used in the optimization, one approach is to first represent the point locations graphically like in Fig 11 and then visually define the design space boundaries by selecting, for each variable, upper and lower values, values which may vary as functions of key parameters, such as TWS. This method of pre-restraining the domain is subjective as it relies on the reader’s ability to evaluate the boundary values, labor-intensive as it requires one plot for each design variable, and inadequate for complex cases where no single variable can be used to establish such bounds.

In the case of the yacht, TWS plays a key role in the actual boat speed so that by focusing on 1-knot increments in TWS, it becomes possible to define upper and lower value for the eight design variables by plotting seven similar two-dimensional graphs as functions of TWS, as in Fig.4. This approach enables us to define a reference case against which we can compare with the automated filling process where such parameter restrictions are not imposed.

5.3.2 Automated space filling

The first step is to convert the sailing database independent variables to non-dimensional values so that the database is contained in a unit hypercube of dimension n , with $n = 8$ in our example. Second, we generate a random set of non-dimensional points/vectors in the same hypercube. The randomly generated points are added throughout the design space and those “too close” to a valid point of the database are removed following the approach described above; only the points populating the voids are kept.

Courouble *et al.* (2008) show that 1,000 random points over the targeted space provide good accuracy. For the current dataset, the largest of the minimum distances expressed in terms of non-dimensional vectors is 0.698; for convenience we will choose $\delta=0.7$ as the distance criterion. Therefore, the randomly generated points are kept only if they are at least at a distance of $k \times 0.7$ from their neighbors. For example, for $k = 1$, about 40% of the 1000 random points are rejected with the criterion of 0.70.

5.4 Process overview

Fig. 13 presents an overview of the automated process, starting with the database containing the filtered sailing data.

Since the validation set (VS) is used for stopping the training and our interest is to train the function over the sailing telemetry, the VS will be composed of valid sailing data points only. In our case, the VS is about 300 points and all remaining points are used for the training set (TS) to which points are added following the approach described above (except in the *reference case*). The TS and VS are then used to train the NN, so that VMG can be determined instantaneously from given sailing parameters and wind conditions.

The global optimization method used here is a Genetic Algorithm (GA). GA is a search and optimization method based on the process of biological evolution, in that they involve a search from a population of solutions and not from a single point. Each iteration of a GA involves a competitive selection that penalizes poor solutions. The solutions with high fitness are recombined with others to produce members of the next generation. Reproduction and mutations are used to generate new solutions which are biased towards regions of the space for which good solutions have already been seen. The strength of the GA is that they perform well in spaces where there may be multiple local optima. The typical drawback of GA is the requirement for a rather large number of function evaluations, a requirement easily met here with the use of the NN since objective functions can be evaluated instantaneously.

The overall dataset is primarily based on the TWS going from 10 to 15 knots; consistent with sailing practice. In order to establish well defined *reference cases* to be used for evaluation of the automated approach (see above), the analysis is split in one-knot increments, starting at 10 knots. In the general case, however, such splitting would not be necessary since the NN would be capable of representing the dependency of the boat speed on TWS (as long as TWS is an independent parameter). A summary of the results is presented in the next section.

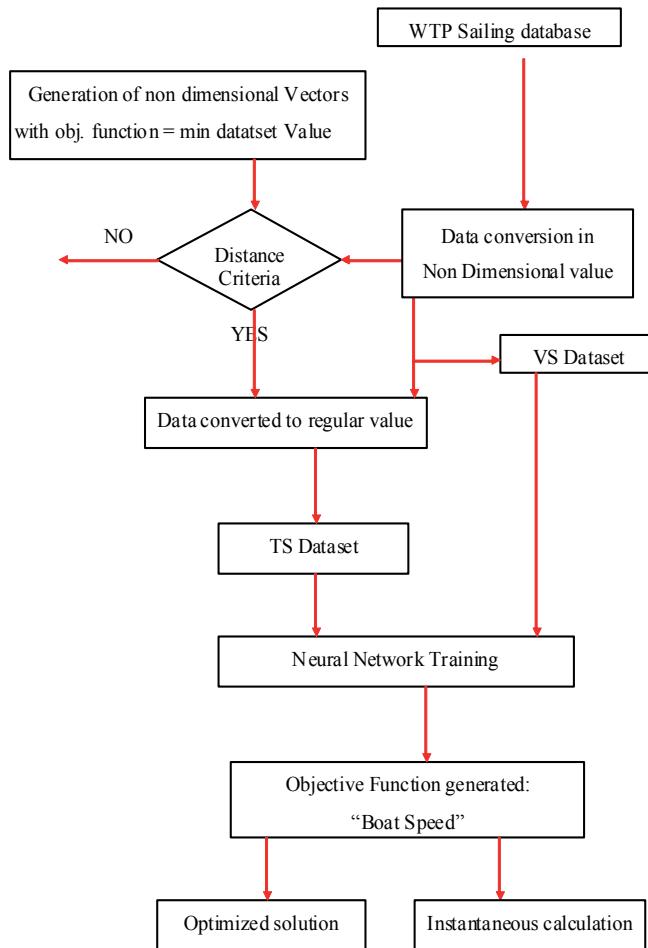


Fig. 13. Automated process starting with the WTP sailing database and leading to a NN capable of instantaneously calculating boat speed, or which can be used to generate optimum sailing setups under varying conditions.

5.5 Results and Discussion

5.5.1 Effects of Optimum Distance Criterion and Number of Added Points

All optimizations shown here are performed with a population size of 5000 with a stopping criterion of 100 generations. One optimum is obtained in about eight minutes on a regular personal computer, corresponding to half a million operations of the trained function. As noted above, the results obtained from the automated database processing are compared with a *reference case* where the search space has been greatly restricted to fit the available data. In this section, all added points are assigned a boat speed of 0. Table 4 presents the effect of the number of added points (before those too close to sailing data clusters are removed automatically) and distance criterion in comparison with the optimum values found in the reference case. With no added points, the optimum solution of run 1 is off by about 12% when comparing the value with the reference case. This large error is due to

"waves" in the network near the edges of data clusters because of lack of data outside these clusters and because some of the best sailing conditions are actually near some of these edges of data clusters.

Increasing gradually the number of points from one hundred to one thousand, the difference drops to 2.1%. For a margin within 2% accuracy, 1000 points added over the six thousand sailing data points, means a minimum of 16% of added points is required the whole dataset. Theoretically in training a NN there is no limitation in terms of size, so adding more points would then not be a problem. Therefore, to improve the non-valid domain coverage, a higher percentage of added points (such as up to 20% of the total original dataset size) appears to be a safe margin to start with.

The second element to investigate is the distance criterion. Our initial distance was based on $k = 1$ so as to insert additional points not too close to the valid sailing points. Results for two additional datasets are shown in Table 4, both with one thousand additional points but differing by their distance criterion (k coefficient). In the first dataset, the distance criterion is reduced to 75% ($k=0.75$); in the second dataset, the distance is reduced by 50% ($k=0.5$). For $k=0.5$, the dataset is within 1% of the target solution; $k=0.75$ shows a difference of 3.5%, which is still acceptable. This solution, however, is marginal, with parameters like heel angle or forestay tension being lower than typical values. This discrepancy is likely due to the fact that an objective function, boat speed, of zero was set of additional points which also creates "waves" in the NN near the edges of clusters. The value to use for these additional points is discussed below.

Case	# of added points	k	Error %*
1	0	1	12.5
2	100	1	7.2
3	500	1	5.5
4	1000	1	2.1
5	1000	0.75	3.5
6	1000	0.5	0.7

Table 4. Comparison of optimum obtained with the automated database processing with that obtained with the reference case over a more restrictive design space. TWS: 10-11 Knots.

5.5.2 Effect of Boat Speed for Added Points and Impact on Minimum Distance Criterion

To prevent the optimizer from searching in the low point density area, we assigned boat speed of zero for all the added points. Populating non-valid areas of the domain by non-sailing data points with speed value set to zero is radical but efficient, especially if the domain to be covered is very large for the amount of added points available. But as we get closer to the sailing data points, we risk that locally the function value be altered. Here, the sailing dataset shows boat speed values ranging from 7.1 to 11.3 knots, and inserting a point with zero value creates a radical damping of 170 % in the function value and may remove potential attractive solutions in the optimization process or create waves near the edges of sailing data clusters. To prevent that phenomenon near the edges, the function value for the added points is set to be equal to the overall minimum boat speed of the dataset (i.e. 7.1

* compared with reference case

knots) instead of 0.0 as set in earlier analyses. From that statement we created two new datasets with similar number of points added (1000) and $k=1$, but with function values equal to the lower boat speed value (7.1). The solutions when compared with the reference case show differences within 1.2 %, which is even better than the 3% obtained earlier.

In regard to the settings (eight sailing variables), there is also more consistency between the two solutions. On a sailboat many different set ups can provide nearly the same boat speed, as long as they are coherent. In this case, although the TWS are not exactly the same the setting numbers are globally in the same range which is an important factor to emphasize, meaning the solutions have been optimized for both domains in a similar zone. Courouble *et al.* (2008) compare these solutions in more detail and show that although the solutions are within few percent, a closer look at the optimum design variables for heel, forestay, rudder and trim tab show some differences between the two solutions, but similar trends, suggesting that several combinations of sailing parameters may be used for optimum boat speed. The results demonstrate that the method offers excellent potential for identifying the areas of interests for further investigation. Although not performed in the present study, the use of a multi-island genetic algorithm would allow a user to explore such areas systematically.

6. References

- Agatonovic-Kustrin, S.; Zecevic, M.; Zivanovic, L.; and Tucker, I.G. (1998) "Application of Neural Networks for Response Surface Modeling in HPLC Optimization," *Analytica Chimica Acta*, Vol. 364, pp. 265-273.
- ANSYS, Inc. (2009) "ANSYS ICEM CFD" [Online] available at <<http://www.ansys.com/products/icemcfd.asp>>, accessed 01 June 2009.
- Bertram, V.; Mesbahi, E. (2004) "Estimating Resistance and Power for fast Monohulls Employing Artificial Neural Nets," 4th Int. Conf. High-Performance Marine Vehicles (HIPER), Rome.
- Besnard, E.; Schmitz, A.; Kaups, K.; Tzong, G.; Hefazi, H.; Chen, H.H.; Kural, O.; and Cebeci, T (1998) "Hydrofoil Design and Optimization for Fast Ships," *Proceedings of the 1998 ASME International Congress and Exhibition*, Anaheim, CA.
- Besnard, E.; Schmitz, A.; Hefazi, H.; and Shinde, R. (2007) "Constructive Neural Networks and their Application to Ship Multi-disciplinary Design Optimization," *Journal of Ship Research*, Vol. 51, No. 4, pp. 297-312.
- Blanchard, B. and Fabrycky, W. (1997) *Systems Engineering and Analysis*, 3rd Ed., Prentice Hall.
- Bishop, C. (1995) *Neural Networks for Pattern Recognition*, Oxford University Press.
- Bourquin, J.; Schmidli, H.; van Hoogevest, P.; and Leuenberger, H. (1998) "Advantages of Artificial Neural Networks (ANNs) as Alternative Modeling Technique for Data Sets Showing Non-linear Relationships Using Data from a Galenical Study on a Solid Dosage Form," *European Journal of Pharmaceutical Sciences*, Vol. 7, pp. 5-16.
- Cybenko, G. (1989) "Approximation by Superpositions of a Sigmoidal Function, " *Mathematics of Control, Signal and Systems*, Vol. 2, Issue 4, pp. 303-314
- Courouble, F.; Besnard, E.; and Schmitz, A. (2008) "Application of Constructive Neural Networks to America's Cup Racing Yacht Performance Optimization," Paper

- presented at the MDY'08, 3rd Symposium on Yacht Design and Production, Madrid Spain.
- Danışman, D. B.; Mesbahi, E.; Atlar, M. and Goren, O. (2002) "A New Hull Form Optimization Technique for Minimum Wave Resistance," 10th International Maritime Association Mediterranean Congress (IMAM), Crete
- Dassault Systèmes SIMULIA (2009). "iSIGHT - Integrate, Automate, and Optimize your Manual Design Processes," [Online] available at <<http://www.simulia.com/products/isight.html>>, accessed 01 June 2009.
- Dutt, J. R.; Dutta, P. K.; and Banerjee, R. (2004) "Optimization of Culture Parameters for Extracellular Protease Production from a Newly Isolated *Pseudomonas* sp. using Response Surface and Artificial Neural Network Models," *Process Biochemistry*, Vol 39, pp. 2193-2198.
- Fahlman, S.E. (1988) "Faster-Learning Variations on Back-Propagation Learning: An Empirical Study," in *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann.
- Fahlman, S. E. and Lebiere, C. (1990) "The Cascade-Correlation Learning Architecture, " *Technical Report CMU-CS-90-100*, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA.
- Gomes, H. M.; and Awruch, A. M. (2004) "Comparison of Response Surface and Neural Network with other Methods for Structural Reliability Analysis," *Structural Safety*, Vol. 26, pp. 49-67.
- Gouguolidis, G. (2008), "The utilization of Artificial Neural Networks in Marine Applications: An Overview". *Journal of Naval Engineering*, Vol. 120, No.3, pp 19-26, 2008.
- Hefazi, H *et al.* (2002) "CFD Design Tool Development and Validation, CCDoTT FY00 Task 2.8," Center for the Commercial Deployment of Transportation Technologies, Long Beach, CA., [Online] available at <ftp://www.foundation.csulb.edu/CCDoTT/Deliverables/2000/Task%202.8/task2.8_1.pdf>, accessed 01 March 2009.
- Hefazi, H *et al.* (2003) "Computer Software Product End Item, Deliverable 2, Optimization Tool Development Based on Neural Networks Computer DCI-MCCR-80700 report," Center for the Commercial Deployment of Transportation Technologies, Long Beach, CA. [Online] available at <ftp://www.foundation.csulb.edu/CCDoTT/Deliverables/2002/task%202.20/task%202.20_opttools%20FY%2002.pdf>, accessed 01 March 2009.
- Hornik, K. (1991) "Approximation Capabilities of Multilayer Feedforward Networks, " *Neural Networks*, Vol. 4, Issue 2, pp. 251-257.
- Jain, P.; and Deo, M. C. (2005) "Neural Networks in Ocean Engineering," *Ships And Offshore Structures (SAOS 2006)*, Vol. 1, Issue 1, pp. 25-35.
- Koushan, K. (2003) "Automatic Hull Form Optimization Towards Lower Resistance and Wash using Artificial Intelligence," FAST 2003 Conference, Ischia, Italy.
- Koh, L.; Janson, C-E, Altar, M.; Larsson, L.; Mesbahi, E.; and Abt, C. (2005) "Novel Design and Hydrodynamic Optimization of a High Speed Hull Form," 5th International Conference on High Performance Marine Vessels, Shanghai.

- Kwok, T. Y. and Yeung, D. Y. (1993) "Theoretical Analysis of Constructive Neural Networks," *Technical Report HKUST-CS-93-12*, Hong Kong University of Science and Technology.
- Kwok, T. Y. and Yeung, D. Y. (1997a) "Constructive Algorithms for Structure Learning in Feedforward Neural Networks for Regression Problems," *IEEE Transactions on Neural Networks*, Vol. 8, Issue 3, pp. 630-645.
- Kwok, T. Y. and Yeung, D. Y. (1997b) "Objective Function for Training New Hidden Units in Constructive Neural Networks," *IEEE transactions on Neural Networks*, Vol. 8, Issue 5, pp 1131-1148.
- Lahnajärvi J.J.T. et al.(2002) "Evaluation of Constructive Neural Networks with Cascaded Architectures," *Neurocomputing*, Vol. 48, pp 573-607.
- Lee, J.; and Hajel, P. (2001) "Application of Classifier Systems in Improving Response Surface based Approximations for Design Optimization," *Computers and Structures*, Vol. 79, pp. 333-344.
- Lehtokangas, M. (1999) "Fast Initialization for Cascade-Correlation Learning," *IEEE Transactions on Neural Networks*, Vol. 10, nº 2.
- Lin, C.Y.; and Wu, W.H. (2002) "Niche Identification Techniques in Multimodal Genetic Search with Sharing Scheme," *Advances in Engineering Software*, Vol. 22, pp. 779-791.
- Maisonneuve, J.J. (2003) "Chapter 7: Applications Examples from Industry," in *Optimistic - Optimization in Marine Design*, 2nd edition, Birk, L., and Harries, S. (Editors), Mensch & Buch Verlag.
- Mesbahi, E.; Bertram, V. (2000) "Empirical Design Formulae Using Artificial Neural Nets," *COMPIT'2000*, Potsdam.
- Parametric Technology Corporation (2009) "PTC : Pro/ENGINEER," [Online] available at <<http://www.ptc.com/appserver/mkt/products/home.jsp?k=403>>, accessed 01 June 2009.
- Prechelt, L. (1997) "Investigation of the CasCor Family of Learning Algorithms," *Neural Networks*, Vol. 10, No. 5, pp 885-896.
- Prechelt, L. (1998a) "Automatic Early Stopping Using Cross-Validation: Quantifying the Criteria," *Neural Networks*, Vol. 11, Number 4, 761-767
- Prechelt, (1998b) "Early Stopping : But When?," in *Neural networks : Tricks of the Trade*, Lecture Notes In Computer Science, Vol. 1524, pp 55-69, Orr, G.B., and Mueller, K.-R., eds.
- Sarle, W.S., ed. (2002) "Neural Network FAQ, Usenet newsgroup comp.ai.neural-nets," [Online] available at <<ftp://ftp.sas.com/pub/neural/FAQ.html>>, accessed 01 June 2009.
- Schmitz, A. (2007) *Constructive Neural Networks for Function Approximation and their Application to CFD Shape Optimization*, Ph.D Thesis, Claremont Graduate University.
- Tekto, I.; Villa, A. (1997) "An Enhancement of Generalization Ability in Cascade Correlation Algorithm by Avoidance of Overfitting/Overtraining Problem", *Neural Porcesssing Letters*, Vol. 6, pp 43-50
- Takayama, K.; Fujikawa, M.; Obata, Y.; and Morishita, M. (2003) "Neural Network based Optimization of Drug Formulations," *Advanced Drug Delivery Reviews*, Vol. 55, pp. 1217-1231.

- Todoroki, A.; and Ishikawa, T. (2004) "Design of Experiments for Stacking Sequence Optimizations with Genetic Algorithm using Response Surface Approximation," *Composite Structures*, Vol. 64, pp. 349-357.
- Treagold, .K.; Gedeon,T.D. (1999) "Exploring Constructive Cascade Networks," *IEEE Transactions on Neural Networks*, Vol. 10, No. 6.
- Vanderplaats, Muira & Associates Inc. (1995), *DOT Users Manual, Version 4.20*, VMA Engineering.

Massive-Training Artificial Neural Networks (MTANN) in Computer-Aided Detection of Colorectal Polyps and Lung Nodules in CT

Kenji Suzuki, Ph.D.

*Department of Radiology, Division of Biological Sciences, The University of Chicago
USA*

1. Introduction

Computer-aided diagnosis (CAD) (Giger and Suzuki 2007) has been an active area of study in medical image analysis, because evidence suggests that CAD can help improve the diagnostic performance of radiologists in their image interpretations (Li, Aoyama et al. 2004; Li, Arimura et al. 2005; Dean and Ilvento 2006). Many investigators have participated in and developed CAD schemes for detection/diagnosis of lesions in medical images, such as detection of lung nodules in chest radiographs (Giger, Doi et al. 1988; van Ginneken, ter Haar Romeny et al. 2001; Suzuki, Shiraishi et al. 2005) and in thoracic CT (Armato, Giger et al. 1999; Armato, Li et al. 2002; Suzuki, Armato et al. 2003; Arimura, Katsuragawa et al. 2004), detection of microcalcifications/masses in mammography (Chan, Doi et al. 1987), breast MRI (Gilhuijs, Giger et al. 1998), breast US (Horsch, Giger et al. 2004; Drukker, Giger et al. 2005), and detection of polyps in CT colonography (Yoshida and Nappi 2001; Suzuki, Yoshida et al. 2006; Suzuki, Yoshida et al. 2008). Some advanced CAD schemes employ a filter for enhancement of lesions as a preprocessing step for improving sensitivity and specificity. The filter enhances objects similar to a model employed in the filter; e.g., a blob enhancement filter based on the Hessian matrix enhances sphere-like objects (Frangi, Niessen et al. 1999). Actual lesions, however, often differ from a simple model, e.g., a lung nodule is generally modeled as a solid sphere, but there are nodules of various shapes and inhomogeneous nodules such as a spiculated nodule and a ground-glass opacity. A colorectal polyp is often modeled as a cap structure by using a shape index filter, but a sessile polyp or a flat polyp cannot be characterized well as a cap structure of the shape index. Thus, conventional filters often fail to enhance actual lesions such as lung nodules with ground-glass opacity and sessile/flat polyps.

To address this issue, we developed a supervised filter for enhancement of actual lesions by use of a massive-training artificial neural network (MTANN) (Suzuki, Armato et al. 2003) filter in a CAD scheme. In this chapter, we introduce MTANN-based CAD schemes for detection of lung nodules in CT and for detection of polyps in CT colonography. To summarize, by extension of “neural filters” (Suzuki, Horiba et al. 2002) and “neural edge enhancers” (Suzuki, Horiba et al. 2003; Suzuki, Horiba et al. 2004), which are ANN-based

(Rumelhart, Hinton et al. 1986) supervised nonlinear image-processing techniques, MTANNs (Suzuki, Armato et al. 2003) have been developed for accommodating the task of distinguishing a specific opacity from other opacities in medical images. MTANNs have been applied to the reduction of false positives (FPs) in the computerized detection of lung nodules in low-dose CT (Suzuki, Armato et al. 2003; Arimura, Katsuragawa et al. 2004) and chest radiography (Suzuki, Shiraishi et al. 2005), for distinction between benign and malignant lung nodules in CT (Suzuki, Li et al. 2005), for suppression of ribs in chest radiographs (Suzuki, Abe et al. 2006), and for reduction of FPs in computerized detection of polyps in CT colonography (Suzuki, Yoshida et al. 2006; Suzuki, Yoshida et al. 2008). The MTANN filter is trained with actual lesions in CT images to enhance the actual patterns of the lesions. We evaluated the performance of our CAD schemes incorporating the MTANNs for detection of lung nodules in CT and for detection of polyps in CT colonography.

2. A MTANN Filter for Lesion Enhancement

2.1. An Architecture of an MTANN Filter

To enhance actual lesions in medical images, we developed an MTANN supervised filter. The architecture of an MTANN supervised filter is shown in Fig. 1. An MTANN filter consists of a linear-output regression artificial neural network (LOR-ANN) model (Suzuki, Horiba et al. 2003), which is a regression-type ANN capable of operating on pixel/volel data directly. The MTANN filter is trained with input CT images and the corresponding “teaching” images that contain a map for the “likelihood of being lesions.” The pixel values of the input images are linearly scaled such that -1,000 Hounsfield units (HU) corresponds to 0 and 1,000 HU corresponds to 1. The input to the MTANN filter consists of pixel values in a sub-region, R_s , extracted from an input image. The output of the MTANN filter is a continuous scalar value, which is associated with the center pixel in the sub-region, and is represented by

$$O(x, y) = \text{LORANN}\{I(x-i, y-j) | (i, j) \in R_s\}, \quad (1)$$

where x and y are the coordinate indices, $\text{LORANN}(\cdot)$ is the output of the LOR-ANN model, and $I(x, y)$ is a pixel value in the input image. The LOR-ANN employs a linear function, $f_L(u) = a \cdot u + 0.5$, instead of a sigmoid function, $f_S(u) = 1/\{1 + \exp(-u)\}$, as the activation function of the output layer unit because the characteristics and performance of an ANN are improved significantly with a linear function when applied to the continuous mapping of values in image processing (Suzuki, Horiba et al. 2003). Note that the activation function in the hidden layers is still a sigmoid function. The input vector can be rewritten as

$$\vec{I}_{x,y} = \{I_1, I_2, \dots, I_m, \dots, I_{N_I}\}, \quad (2)$$

where m is an input unit number, and N_I is the number of input units. The output of the n -th unit in the hidden layer is represented by

$$O_n^H = f_S \left\{ \sum_{m=1}^{N_I} w_{mn}^H \cdot I_m - w_{0n}^H \right\}, \quad (3)$$

where W^H_{mn} is a weight between the m -th unit in the input layer and the n -th unit in the hidden layer, and W^H_{0n} is an offset of the n -th unit in the hidden layer. The output of the output layer unit is represented by

$$O(x, y) = f_L \left\{ \sum_{m=1}^{N_H} w_m^O \cdot O_m^H - w_0^O \right\}, \quad (4)$$

where w_m^O is a weight between the m -th unit in the hidden layer and the unit in the output layer, N_H is the number of units in the hidden layer, and w_0^O is an offset of the unit in the output layer. For processing of the entire image, the scanning of an input CT image with the MTANN is performed pixel by pixel, as illustrated in Fig. 2(b).

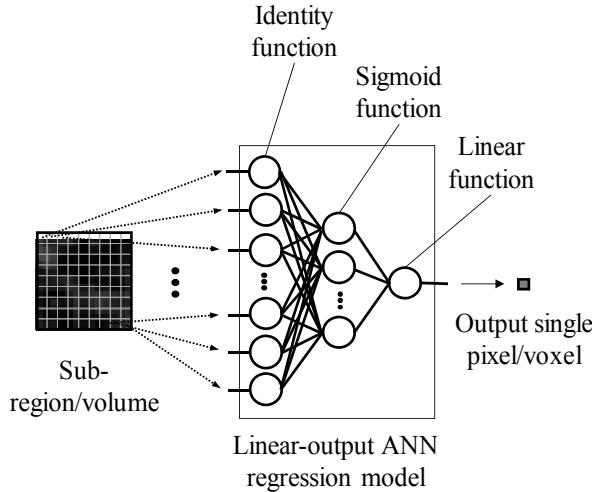


Fig. 1. Architecture of an MTANN supervised filter consisting of a LOR-ANN model with sub-region input and single-pixel output. All pixel values in a sub-region extracted from an input CT image are entered as input to the LOR-ANN. The LOR-ANN outputs a single pixel value for each sub-region, the location of which corresponds to the center pixel in the sub-region. Output pixel value is mapped back to the corresponding pixel in the output image.

2.2. Training of an MTANN Filter

For enhancement of lesions and suppression of non-lesions in CT images, the teaching image $T(x, y)$ contains a map for the "likelihood of being lesions," as illustrated in Fig. 2(a). To create the teaching image, we first segment lesions manually for obtaining a binary image with 1 being lesion pixels and 0 being non-lesion pixels. Then, Gaussian smoothing is applied to the binary image for smoothing down the edges of the segmented lesions, because the likelihood of being lesions should gradually be diminished as the distance from the boundary of the lesion decreases. Note that the ANN was not able to be trained when binary teaching images were used.

The MTANN filter involves training with a large number of pairs of sub-regions and pixels; we call it a massive-sub-region training scheme. For enrichment of the training samples, a training image, R_T , extracted from the input CT image is divided pixel by pixel into a large

number of sub-regions. Note that close sub-regions overlap each other. Single pixels are extracted from the corresponding teaching image as teaching values. The MTANN filter is massively trained by use of each of a large number of input sub-regions together with each of the corresponding teaching single pixels; hence the term "massive-training ANN." The error to be minimized by training of the MTANN filter is given by

$$E = \frac{1}{P} \sum_c \sum_{(x,y) \in R_T} \{T_c(x,y) - O_c(x,y)\}^2 \quad , \quad (5)$$

where c is a training case number, O_c is the output of the MTANN for the c -th case, T_c is the teaching value for the MTANN for the c -th case, and P is the number of total training pixels in the training images, R_T . The MTANN filter is trained by a linear-output back-propagation (BP) algorithm where the generalized delta rule (Rumelhart, Hinton et al. 1986) is applied to the LOR-ANN architecture (Suzuki, Horiba et al. 2003). After training, the MTANN filter is expected to output the highest value when a lesion is located at the center of the sub-region of the MTANN filter, a lower value as the distance from the sub-region center increases, and zero when the input sub-region contains a non-lesion.

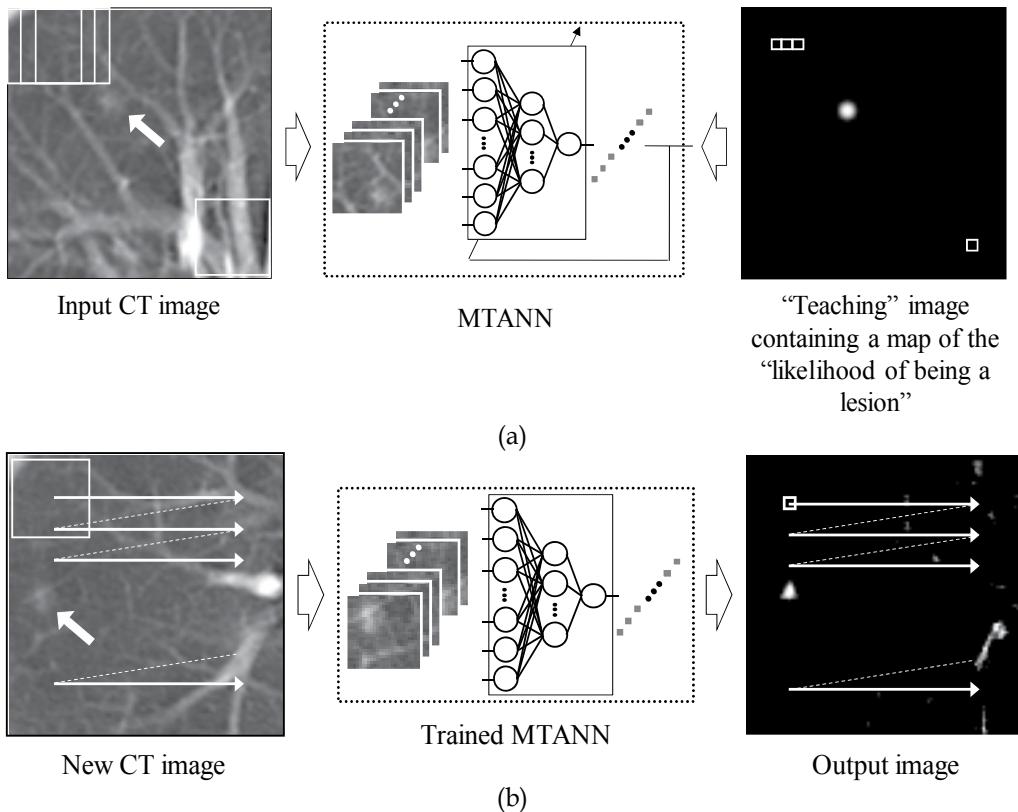


Fig. 2. Training and application of an MTANN filter for enhancement of lesions. (a) Training of an MTANN filter. The input CT image is divided pixel by pixel into a large number of overlapping sub-regions. The corresponding pixels are extracted from the "teaching" image containing a map for the "likelihood of being a lesion." The MTANN filter is trained with

pairs of the input sub-regions and the corresponding teaching pixels. (b) Application of the trained MTANN filter to a new CT image. Scanning with the trained MTANN filter is performed for obtaining pixel values in the entire output image.

3. An MTANN for Classification

3.1. A Training Method of an MTANN for Classification

For distinction between lesions and non-lesions in medical images, the teaching image contains a Gaussian distribution with standard deviation σ_T for a nodule and zero for a non-nodule (i.e., completely dark), as shown in Fig. 3. This distribution represents a map for the “likelihood of being a lesion”:

$$T(x, y) = \begin{cases} \frac{1}{\sqrt{2\pi}\sigma_T} \exp\left\{-\frac{(x^2 + y^2)}{2\sigma_T^2}\right\} & \text{for a lesion} \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

To enrich the training samples, a training region, R_T , extracted from the input image is divided pixel by pixel into a large number of overlapping sub-regions. Single pixels are extracted from the corresponding teaching region as teaching values. The MTANN is massively trained by use of each of a large number of the input sub-regions together with each of the corresponding teaching single pixels. After training, the MTANN is expected to output the highest value when a lesion is located at the center of the sub-region of the MTANN, a lower value as the distance from the sub-region center increases, and zero when the input sub-region contains a non-lesion.

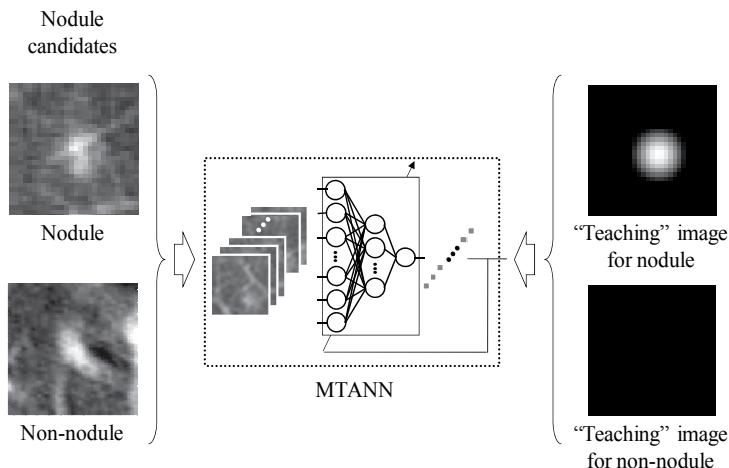


Fig. 3. Architecture and training of an MTANN for classification of candidates into a lesion (e.g., a nodule) or a non-lesion (e.g., a non-nodule). A teaching image for a nodule contains a Gaussian distribution at the center of the image, whereas that for a non-nodule contains zero (i.e., it is completely dark).

3.2. A Scoring Method for Combining Output Pixels

For combining output pixels from a trained MTANN, we developed a scoring method. A score for a given lesion candidate from the trained MTANN is defined as

$$S = \sum_{(x,y) \in R_E} f_G(\sigma; x, y) \times O(x, y) \quad (7)$$

where

$$f_G(\sigma; x, y) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{(x^2 + y^2)}{2\sigma^2}\right\} \quad (8)$$

is a Gaussian weighting function with standard deviation σ , and with its center corresponding to the center of the region for evaluation, R_E ; and $O(x, y)$ is the output region of the n -th trained MTANN, where its center corresponds to the center of R_E . The use of the Gaussian weighting function allows us to combine the responses (outputs) of a trained MTANN as a distribution. A Gaussian function is used for scoring, because the output of a trained MTANN is expected to be similar to the Gaussian distribution used in the teaching region. This score represents the weighted sum of the estimates for the likelihood that the region (lesion candidate) contains a lesion near the center, i.e., a higher score would indicate a lesion, and a lower score would indicate a non-lesion.

3.3. A Mixture of Expert MTANNs

To distinguish lesions from various types of non-lesions (or FPs), we have extended the capability of a single MTANN, and have developed a mixture of expert MTANNs. The architecture of the mixture of expert MTANNs is shown in Fig. 4(a). The mixture of expert MTANNs consists of several MTANNs that are arranged in parallel. Each MTANN is trained independently by use of the same nodules and a different set of non-nodules, as shown in Fig. 4(b). Each MTANN acts as an expert for distinction between lesions (e.g., nodules) and non-lesions (e.g., non-nodules) representing a specific non-lesion type. The scores from the expert MTANNs are combined by use of a mixing ANN such that different types of non-lesions can be distinguished from lesions. The mixing ANN consists of a linear-output multilayer ANN model with a linear-output BP training algorithm (Suzuki, Horiba et al. 2003) for processing of continuous output/teaching values; the activation functions of the units in the input, hidden, and output layers are an identity, a sigmoid, and a linear function, respectively. One unit is employed in the output layer for distinction between a lesion and a non-lesion. The scores of each expert MTANN are used for each input unit in the mixing ANN; thus, the number of input units corresponds to the number of expert MTANNs, N . The scores of each expert MTANN act as the features for distinguishing lesions from a specific type of non-lesion for which the expert MTANN is trained. The output of the mixing ANN for the c -th lesion candidate is represented by

$$M_c = NN[\{S_{n,c}\} \mid 1 \leq n \leq N], \quad (9)$$

where $NN(\cdot)$ is the output of the linear-output ANN model, and n is an MTANN number. The teaching values for lesions are assigned the value one, and those for non-lesions are

zero. Training of the mixing ANN may be performed by use of a leave-one-lesion-out cross-validation scheme (Mosier 1951). After training, the mixing ANN is expected to output a higher value for a lesion and a lower value for a non-lesion. Thus, the output can be considered to be a "likelihood of being a lesion." By thresholding the output, a distinction between lesions and non-lesions can be made. The balance between a true-positive rate and an FP rate is determined by the selected threshold value. If the scores of each expert MTANN properly characterize the specific type of non-lesion for which the expert MTANN is trained, the mixing ANN combining several expert MTANNs will be able to distinguish lesions from various types of non-lesions.

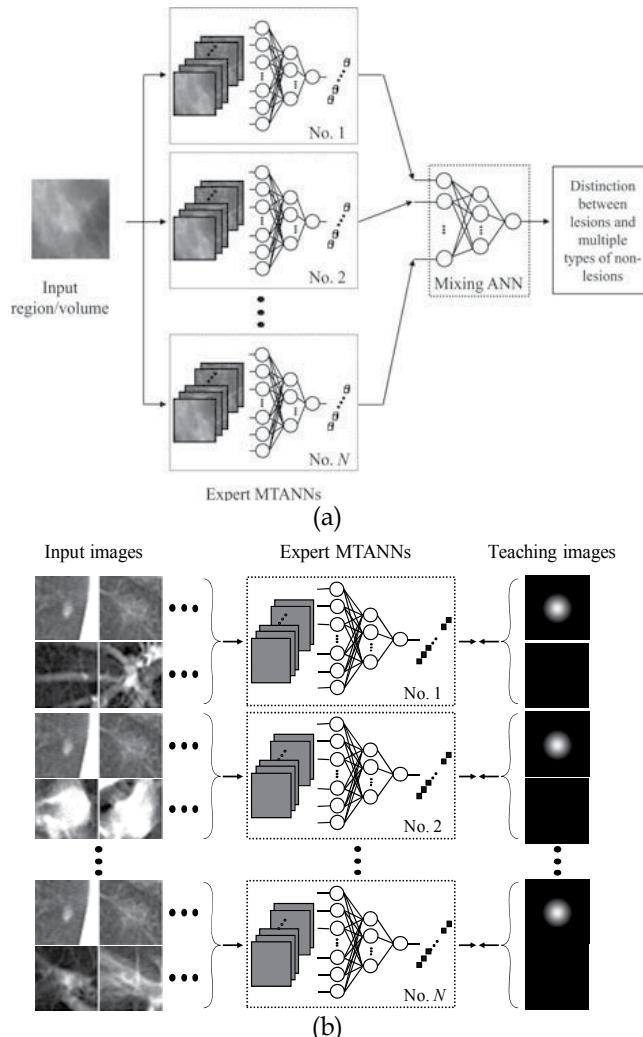


Fig. 4. Architecture (a) and training (b) of a mixture of expert MTANNs for classification of lesion candidates into lesions or multiple types of non-lesions.

4. A CAD Scheme for Detection of Lung Nodules on CT Images

4.1. Lung Cancer Detection in CT

Lung cancer continues to rank as the leading cause of cancer deaths among Americans (American Cancer Society 2005; Jemal, Murray et al. 2005); the number of lung cancer deaths each year is greater than the combined number of breast, colon, and prostate cancer deaths. Evidence suggests that early detection of lung cancer may allow more timely therapeutic intervention and thus a more favorable prognosis for the patient (Heelan, Flehinger et al. 1984; Flehinger, Kimmel et al. 1992; Sobue, Suzuki et al. 1992; Miettinen 2000). Therefore, in the 1970s, screening programs for early detection of lung cancer were carried out with chest radiography and cytologic examination of sputum in the United States (Flehinger, Melamed et al. 1984; Fontana, Sanderson et al. 1984; Frost, Ball et al. 1984) and in Europe (Kubik and Polak 1986). As the CT imaging techniques have advanced, screening with low-dose helical CT has been performed for early detection of lung cancer (Kaneko, Eguchi et al. 1996; Sone, Takashima et al. 1998; Henschke, McCauley et al. 1999; Henschke, Naidich et al. 2001; Miettinen and Henschke 2001; Sone, Li et al. 2001; Nawa, Nakagawa et al. 2002; Swensen, Jett et al. 2003) since early 1990.

Because CT is more sensitive than chest radiography in the detection of small non-calcified nodules due to lung carcinoma at an early stage (Sone, Takashima et al. 1998; Miettinen and Henschke 2001), lung cancer screening programs are being conducted in the United States (Henschke, McCauley et al. 1999; Henschke, Naidich et al. 2001; Miettinen and Henschke 2001; Swensen, Jett et al. 2003) and Japan (Kaneko, Eguchi et al. 1996; Sone, Takashima et al. 1998; Sone, Li et al. 2001; Nawa, Nakagawa et al. 2002) with low-dose single-detector CT as the screening modality. Recently, multi-detector-row CT (MDCT) has been used for lung cancer screening. Helical CT and MDCT, however, generate a large number of images that must be read by radiologists. This may lead to "information overload" for radiologists. Furthermore, radiologists may fail to detect some cancers, which are visible in retrospect, during the interpretation of CT images (Gurney 1996; Li, Sone et al. 2002). Therefore, a CAD scheme for detection of lung nodules in CT has been investigated as a tool for lung cancer screening, because the CAD scheme may detect some cancers that are "missed" by radiologists (Li, Sone et al. 2002), and provide quantitative detection results as a "second opinion" to assist radiologists in improving their detection accuracy (Kobayashi, Xu et al. 1996).

4.2. Database of Lung Nodules in CT

To test the performance of a CAD scheme utilizing the MTANN filters, we created a CT database consisting of 69 lung cancers in 69 patients (Li, Sone et al. 2002). The scans used for this study were acquired with a low-dose protocol of 120 kVp, 25 mA or 50 mA, 10-mm collimation, and a 10-mm reconstruction interval at a helical pitch of two. The reconstructed CT images were 512 × 512 pixels in size with a section thickness of 10 mm. The 69 CT scans consisted of 2,052 sections. All cancers were confirmed either by biopsy or surgically.

4.3. Detection of Nodule Candidates

The flowchart of our CAD scheme utilizing the MTANN supervised lesion enhancement filter and the MTANNs for classification is shown in Fig. 5. To limit processing area to the lungs, we segmented the lung regions in a CT image by use of thresholding based on Otsu's

threshold value determination (Otsu 1979). Then, we applied a rolling-ball technique along the outlines of the extracted lung regions to include a nodule attached to the pleura in the segmented lung regions (Armato, Giger et al. 2001).

To enhance lung nodules in CT images, we trained an MTANN filter with 13 lung nodules in a training database and the corresponding “teaching” images that contained maps for the “likelihood of being nodules,” as illustrated in Fig. 2(a). To obtain the training regions, R_T , we applied a mathematical morphology opening filter to the manually segmented lung nodules (i.e., binary regions) such that the training regions sufficiently covered nodules and surrounding normal structures (i.e., a 9 times larger area than the nodule region, on average). A three-layer structure was employed for the MTANN filter, because any continuous mapping can be approximated by a three-layer ANN (Funahashi 1989). The number of hidden units was selected to be 20 by use of a method for designing the structure of an ANN (Suzuki, Horiba et al. 2001; Suzuki 2004). The size of the input sub-region, R_S , was 9 by 9 pixels, which was determined experimentally in our previous studies (Suzuki, Armato et al. 2003; Arimura, Katsuragawa et al. 2004; Suzuki and Doi 2005). The slope of the linear function, a , was 0.01. With the parameters above, training of the MTANN filter was performed by 1,000,000 iterations. To test the performance, we applied the trained MTANN filter to the entire lungs. We applied thresholding to the output images of the trained MTANN filter to detect nodule candidates. We compared the results of nodule-candidate detection with and without the MTANN filter.

We applied the trained MTANN filter to original CT images. The result of enhancement of nodules in CT images by the trained MTANN filter is shown in Fig. 6. The MTANN filter enhances the nodule and suppresses most of the normal structures in the CT image. Although some medium-sized vessels and some of the large vessels in the hilum remain in the output image, the nodule with spiculation is enhanced well. We applied thresholding to the output images of the trained MTANN filter. There are a smaller number of candidates in the MTANN-based image, as illustrated in Fig. 6(c), whereas there are many nodule candidates in the binary image obtained by use of simple thresholding without the MTANN filter, as shown in Fig. 6(d). Note that the large vessels in the hilum can easily be separated from nodules by use of their area information.

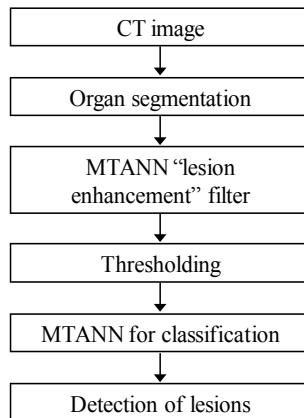


Fig. 5. Flowchart of our CAD scheme utilizing the MTANN supervised lesion enhancement filter and the mixture of expert MTANNs for classification.

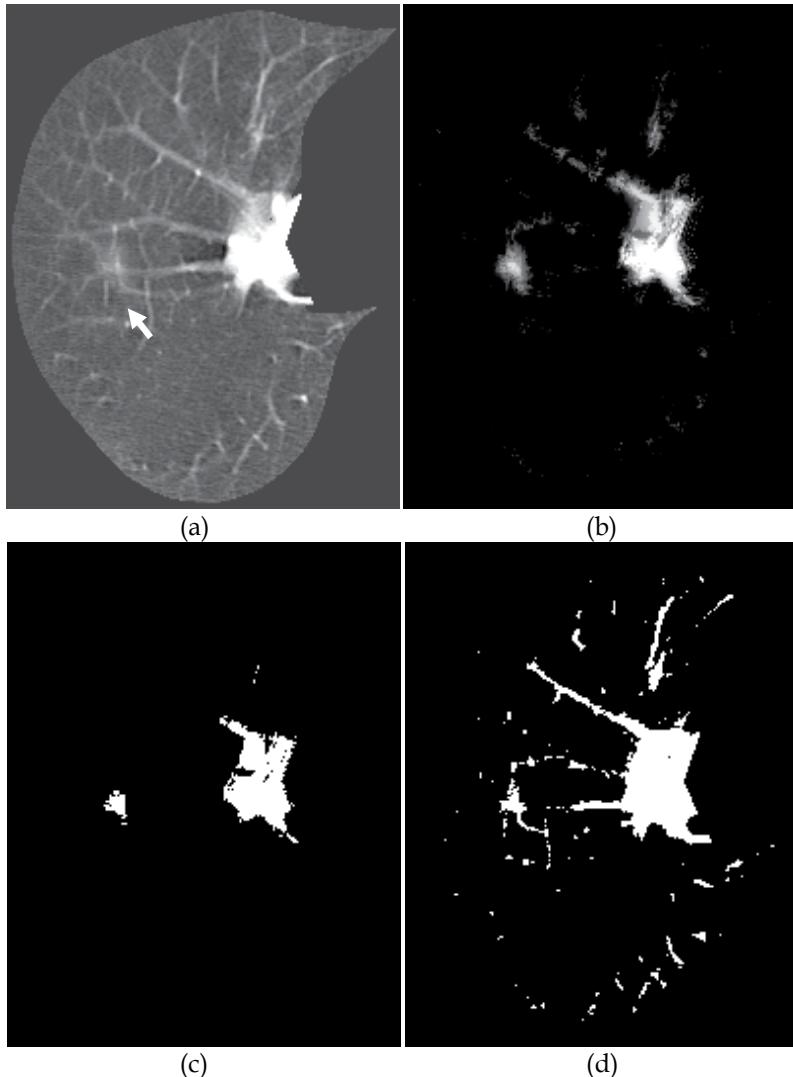


Fig. 6. Enhancement of a lesion by the trained lesion-enhancement MTANN filter for a non-training case. (a) Original image of the segmented lung with a nodule (indicated by an arrow). (b) Output image of the trained lesion-enhancement MTANN filter. The nodule is enhanced in the output image, whereas most of the normal structures are suppressed. (c) Nodule candidates detected by the trained lesion-enhancement MTANN followed by thresholding. (d) Nodule candidates detected by simple thresholding only.

4.4. Classification of Nodule Candidates

Nodule candidates generally include true positives and mostly FPs. For reduction of the FPs, we trained an MTANN for classification of nodule candidates into nodules or non-nodules (Suzuki, Armato et al. 2003; Suzuki, Yoshida et al. 2008). We used 10 different-sized nodules with various contrasts and 10 non-nodule images including medium-sized and small vessels as training cases for the MTANN, as shown in Fig. 7. Parameters such as the

size of the subregion of the MTANN, the standard deviation of the 2D Gaussian function in the teaching image, and the size of the teaching image were determined by experimental analysis (16) to be 9 × 9 pixels, 5.0 pixels, and 19 × 19 pixels, respectively. We employed a three-layer structure for the MTANN, because it has been proved theoretically that a three-layer ANN can approximate any continuous mapping (38,39). The number of hidden units in the MTANN was determined to be 20 by use of a method for determining the structure of an ANN (40,41). Thus, the numbers of input, hidden, and output units were 81, 20, and one, respectively. With the parameters above, the training of the MTANN was performed 500,000 times, and it converged with a mean absolute error of 0.112. Figure 7 shows the input images used for training the MTANN and the output images of the trained MTANN. It is apparent that the nodules are represented by light “fuzzy nodular” distributions in the output images, whereas the vessels are dark and thus “almost removed.”

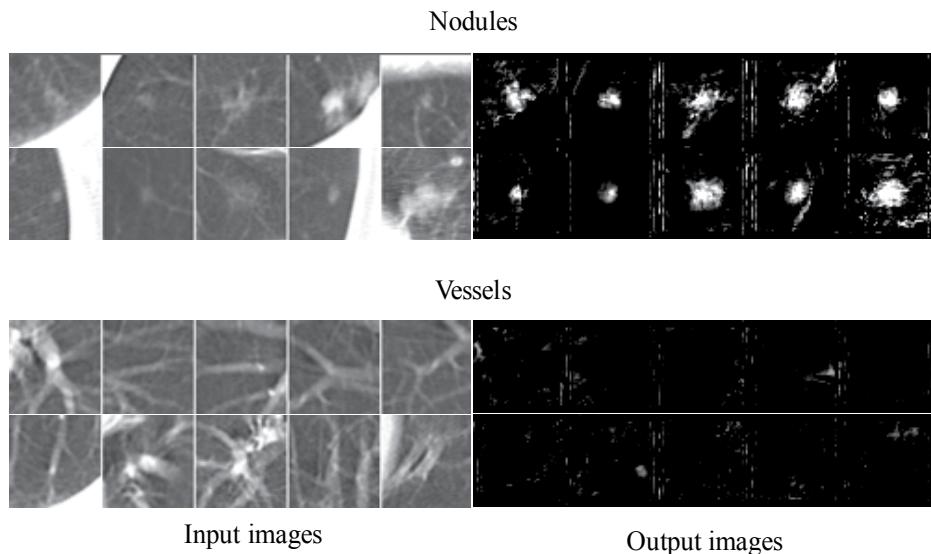


Fig. 7. Ten nodules and 10 non-nodule images including vessels used for training an MTANN, and the corresponding output images of the trained MTANN. The nodules are various-sized with different contrasts. The non-nodule images include medium-sized and small vessels with various orientations, which are the majority of non-nodules in the lungs.

4.5. Simulated CT Images

To investigate the basic characteristics of the trained MTANNs, we created simulated CT images that contained model nodules and model vessels. A nodule was modeled as a sphere, and a vessel as a cylinder. The simulated images included various-sized model nodules (8.0 mm, 14.0 mm, and 20.0 mm in diameter) with low, medium, and high contrast [200 Hounsfield units (HU), 400 HU, and 600 HU], various-sized model vessels (2.0 mm, 3.0 mm, and 4.0 mm in diameter) with different orientations such as horizontal, vertical, and diagonal, and model nodules overlapping with model vessels, as shown in Fig. 8(a). We created the same-sized model nodules with different contrasts, because solid opacity and

ground-glass opacity (GGO) of the same size have different contrasts. The background level was -900 HU, which corresponds to the average background level in the lungs.

Figure 8(b) shows the output image of the MTANN trained with actual nodules. In the output image, the various-sized model nodules with different contrasts are represented by light “nodular” distributions, whereas various-sized model vessels with different orientations are almost dark, and are thus removed. Therefore, it is apparent in the figure that model nodules can be distinguished from model vessels. This result indicates that the MTANN was able to learn from a very small number of training actual cases (10 actual nodules and 10 actual vessel images) to enhance sphere-like objects (model nodules) and suppress cylinder-like objects (model vessels), and that the trained MTANN would be robust against a change in scale and rotation.

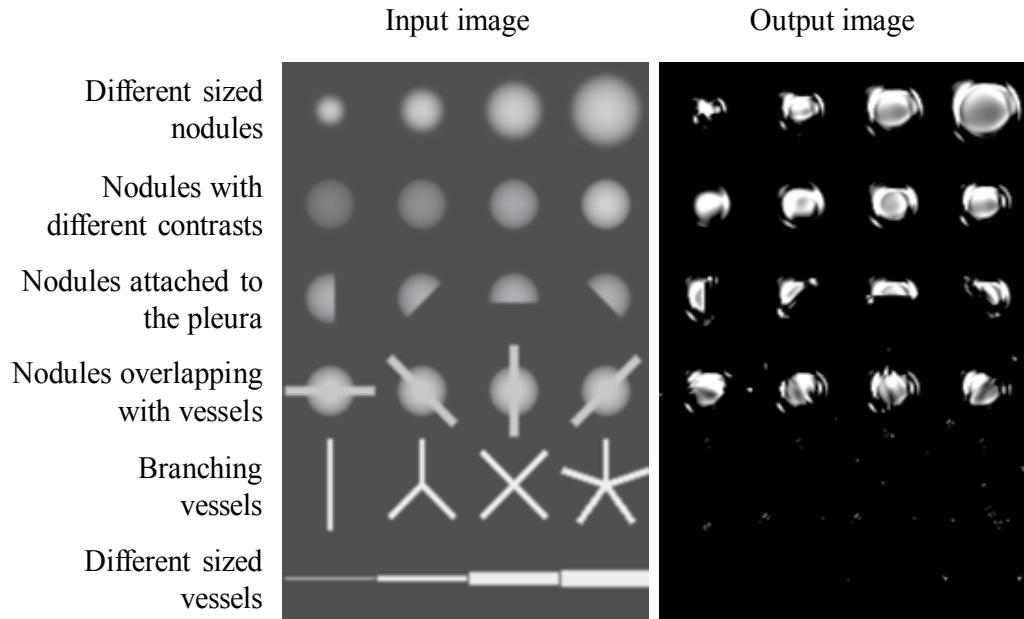
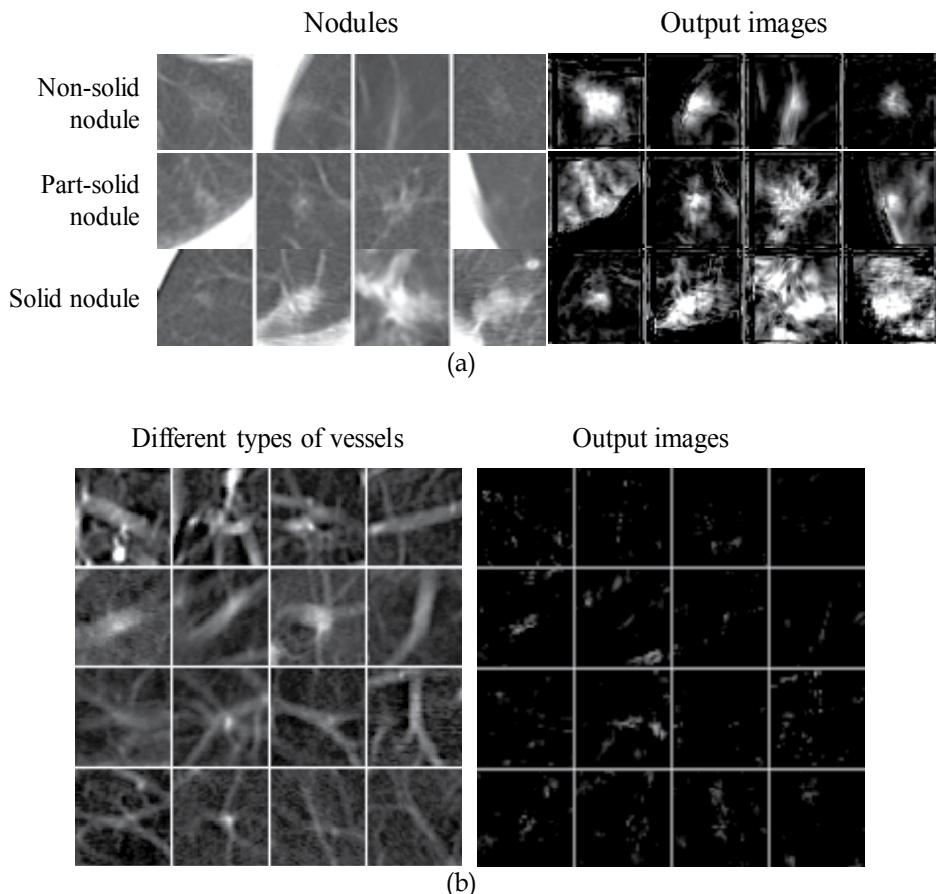


Fig. 8. Simulated CT image that contains various-sized model nodules with different contrasts and various-sized model vessels with different orientations, and the corresponding output images of the MTANNs trained with 10 actual nodules and 10 actual vessel images. (a) Input image for the MTANNs. (b) Output image of the trained MTANN.

4.6. Performance of a CAD Scheme

In order to investigate the performance for actual nodules and vessels, we applied the trained MTANN to non-training cases. Figures 9(a) and (b) show the output images of the trained MTANN, where various-sized actual nodules with different contrasts are represented by light “nodular” distributions, whereas medium-sized and small actual vessels with different orientations are almost eliminated. To distinguish nodules from various types of non-nodules, we trained 6 classification-MTANNs with 10 typical nodules and 6 different types of 10 non-nodules such as medium-sized vessels, small vessels, large vessels, soft-tissue opacity, and abnormal opacities from a training database. We applied the

trained classification-MTANNs to various types of nodules and non-nodules. The trained classification-MTANNs enhance nodules and suppress most of normal structures including various-sized lung vessels in CT images, as shown in Fig. 9. The scores indicating the likelihood of being a nodule from the 6 classification-MTANNs were combined with a mixing ANN to form a mixture of expert classification-MTANNs. We used a leave-one-out cross-validation test for testing the mixing ANN in the mixture of expert MTANNs. We evaluated the performance by using free-response receiver-operating-characteristic (FROC) analysis (Egan, Greenberg et al. 1961).



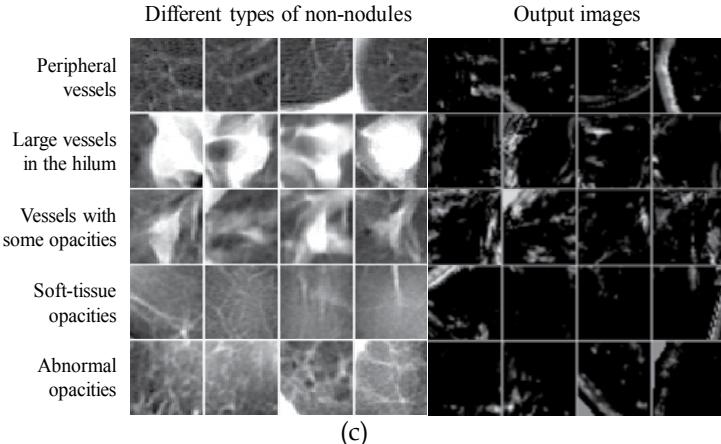


Fig. 9. Illustrations of (a) various types of actual nodules and the corresponding output images of the trained MTANN for non-training cases, (b) various-sized lung vessels and and the corresponding output images, and (c) other types of non-nodules and the corresponding output images.

To test the performance of our CAD scheme utilizing the MTANN lesion enhancement filter and the classification MTANNs, we applied it to the test database containing 69 lung cancers. The MTANN lesion enhancement filter followed by thresholding identified 97% (67/69) of cancers with 6.7 FPs per section. The classification-MTANNs were applied to the nodule candidates (true positives and FPs) for classification of the candidates into nodules or non-nodules. The scores from the 6 classification-MTANNs are shown in Fig. 10. Although the distributions for nodules and non-nodules overlap, many nodules can be separated from non-nodules by decision boundaries. The FROC curve indicating the performance of the mixture of expert MTANNs is shown in Fig. 11. The mixture of expert MTANNs was able to remove 60% (8,172/13,688) or 93% (12,667/13,688) of non-nodules (FPs) with a loss of 1 true positive or 10 true positives, respectively. Thus, our MTANN-based CAD scheme achieved a 96% (66/69) or 84% (57/69) sensitivity with 2.7 (5,516/2,052) or 0.5 (1,021/2,052) FPs per section. The remaining true-positive nodules included a ground-glass opacity, a cancer overlapping vessels, and a cancer touching the pleura. In contrast, the difference-image technique followed by multiple thresholding in the previously reported CAD scheme (Arimura, Katsuragawa et al. 2004) detected 96% (66/69) of cancers with 19.3 FPs per section. Thus, the MTANN lesion-enhancement filter was effective for improving the sensitivity and specificity of a CAD scheme. The feature analysis and the rule-based scheme removed FPs further and achieved 9.3 FPs per section. Finally, with LDA, the previously reported CAD scheme yielded a sensitivity of 84% (57/69) with 1.4 (2,873/2,052) FPs per section. Therefore, MTANNs were effective for improving the sensitivity and specificity of a CAD scheme.

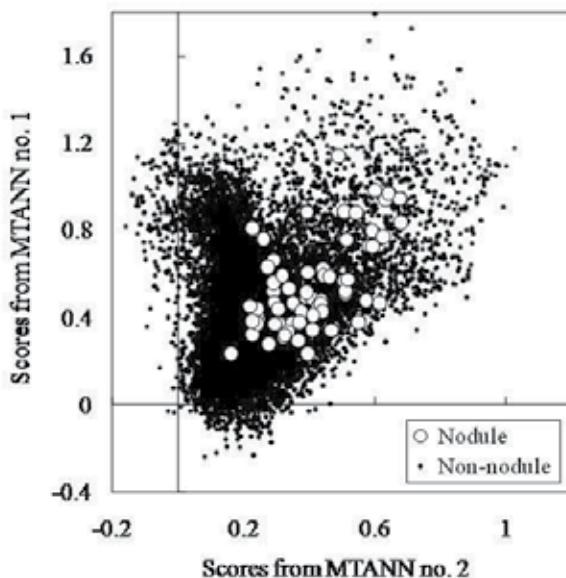


Fig. 10. Distributions of scores from MTANN nos. 1 and 2 of the 6 classification-MTANNs for 67 nodules (white circles) and 13,688 non-nodules (black dots) detected by the lesion-enhancement MTANN filter followed by thresholding.

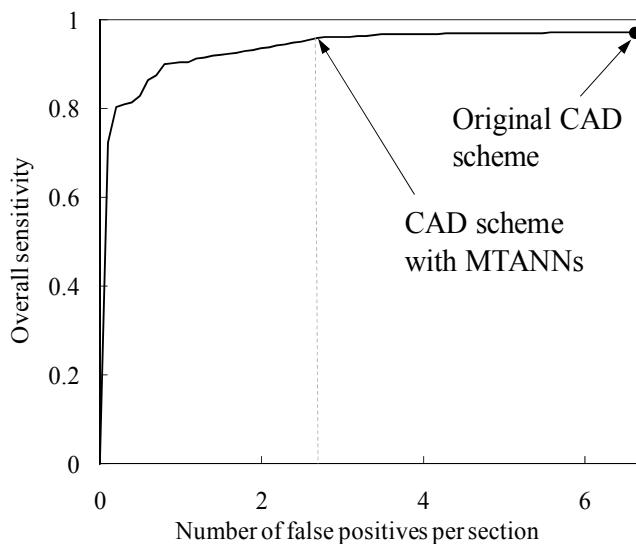


Fig. 11. Performance of the mixture of expert MTANNs for classification between 67 nodules and 13,688 non-nodules. The FROC curve indicates that the mixture of expert MTANNs yielded a reduction of 60% ($8,172/13,688$) or 93% ($12,667/13,688$) of non-nodules (FPs) with a loss of 1 true positive or 10 true positives, respectively, i.e., it achieved a 96% ($66/69$) or 84% ($57/69$) sensitivity with 2.7 ($5,516/2,052$) or 0.5 ($1,021/2,052$) FPs per section, respectively.

5. A CAD Scheme for Detection of Polyps in CTC

5.1. Colorectal Cancer Detection in CTC

Colorectal cancer is the second leading cause of cancer deaths in the United States (Jemal, Murray et al. 2005). Evidence suggests that early detection and removal of polyps (i.e., precursors of colorectal cancer) can reduce the incidence of colorectal cancer (Winawer, Fletcher et al. 1997; Dachman 2003). CT colonography (CTC), also known as virtual colonoscopy, is a technique for detecting colorectal neoplasms by use of a CT scan of the colon (Macari and Bini 2005). The diagnostic performance of CTC in detecting polyps, however, remains uncertain due to a propensity for perceptual errors (Fletcher, Booya et al. 2005). Computer-aided detection (CAD) of polyps has been investigated to overcome the difficulty with CTC. CAD has the potential to improve radiologists' diagnostic performance in the detection of polyps.

Although CAD schemes are useful for improving radiologists' sensitivity in detection of polyps in CTC, a major challenge for CAD schemes is reducing the number of FPs, while maintaining high sensitivity. Major sources of FPs generated by CAD schemes include haustral folds, residual stool, rectal tubes, the ileocecal valve, and extra-colonic structures such as the small bowel and stomach (Yoshida and Dachman 2005). Among those FP sources, rectal tubes are relatively obvious FPs. Radiologists may, however, lose their confidence in CAD as an effective tool if the CAD scheme generates such obvious FPs. Therefore, removal of rectal-tube-induced FPs is desirable. To address this issue, we previously reported a 3D MTANN for distinction between polyps and rectal tubes in 3D CTC volumetric data (Suzuki, Yoshida et al. 2006). The 3D MTANN eliminated all rectal-tube-induced FPs without removal of any true positives. Our purpose in this study was to develop a "mixture of expert" 3D MTANNs for further reduction of FPs in a polyp-detection CAD scheme while maintaining high sensitivity.

5.2. CTC Database

CTC examinations were performed on 73 patients at The University of Chicago Medical Center. The patients' colons were prepared by standard pre-colonoscopy cleansing with administration of cathartics following a water diet or low-fiber diet, and they were insufflated with room air or carbon dioxide. Each patient was scanned in both supine and prone positions. Our database thus contained 146 CTC datasets. The CT scans were performed with either a single- or a multi-detector-row CT scanner (HiSpeed CTi or LightSpeed QX/i, GE Medical Systems, Milwaukee, WI). The CT scanning parameters included collimations between 2.5 and 5.0 mm, reconstruction intervals of 1.0-5.0 mm [1.0 mm (n=2, 1% of the CTC datasets), 1.25 mm (n=3, 2%), 1.5 mm (n=59, 41%), 2.5 mm (n=79, 54%), and 5.0 mm (n=3, 2%)], and tube currents of 60-120 mA with 120 kVp. Each reconstructed CT section had a matrix size of 512 x 512 pixels, with an in-plane pixel size of 0.5-0.7 mm. The CT sections were interpolated to isotropic resolution by use of linear interpolation in the transverse direction. All patients underwent "reference-standard" optical colonoscopy. Radiologists established the locations of polyps in the CTC datasets by use of the colonoscopy and pathology reports, as well as multiplanar reformatted views of the CTC on a viewing workstation (GE Advantage Windows Workstation v.4.2, GE Medical Systems, Milwaukee, WI). In this study, we used 5 mm as the threshold for clinically significant polyps (Johnson and Dachman 2000). Fifteen patients had 28 polyps, 15 of which

were 5-9 mm in diameter, and 13 were 10-25 mm. There was no polyp that was submerged in fluid. Fluid was minimized by use of a saline cathartic preparation as the standard preparation, not a colon gavage. We also created a training database of non-polyps by manual extraction of volumes containing non-polyps from 27 "normal" (non-polyp) CTC cases.

5.3. Performance of Our Initial CAD Scheme

We applied our previously reported CAD scheme (Yoshida and Nappi 2001; Nappi and Yoshida 2003) to the 73 CTC cases. The scheme included centerline-based extraction of the colon (Frimmel, Nappi et al. 2004), shape-based detection of polyps (Yoshida and Näppi 2001; Yoshida, Masutani et al. 2002), and initial reduction of FPs by use of a Bayesian ANN (Kupinski, Edwards et al. 2001) based on geometric and texture features (Nappi and Yoshida 2002; Nappi and Yoshida 2003). We evaluated supine and prone CTC volumes independently. This CAD scheme achieved a 96.4% (27/28 polyps) by-polyp sensitivity with an average of 3.1 (224/73) FPs per patient. Forty-eight true-positive polyp detections in both supine and prone CTC volumes represented 27 polyps. We combined our previously reported CAD scheme with the mixture of expert 3D MTANNs for further reduction of FPs.

5.4. Training of Expert 3D MTANNs

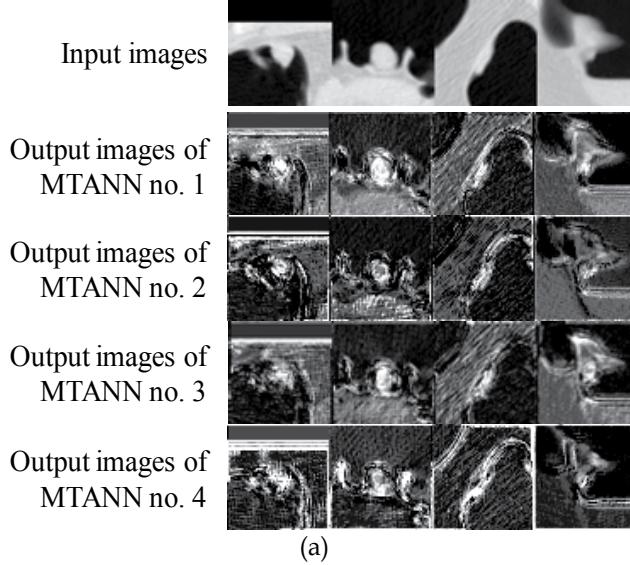
We manually selected ten representative polyp volumes (ten polyps) from the 48 true-positive volumes (containing 27 polyps) in our CTC database as the training polyp cases for expert 3D MTANNs. We classified CAD-generated FP sources into eight categories, i.e., rectal tubes, small bulbous folds, solid stool, stool with bubbles, colonic walls with haustral folds, elongated folds, strip-shaped folds, and the ileocecal valve. We manually selected ten non-polyps in each of the eight categories from the training non-polyp database (which was not used for testing). The ten polyps and the ten rectal tubes were the same as those used in our previous study (Suzuki, Yoshida et al. 2006). The number of sample volumes for each category was ten, because the performance of an expert 3D MTANN was found to be highest when the number of training sample volumes was 20 (i.e., ten polyps and ten non-polyps) in our previous study (Suzuki, Yoshida et al. 2006), and the performance of 2D/3D MTANNs was not sensitive to the number of sample regions/volumes over different types of non-lesions in our previous studies (Suzuki, Armato et al. 2003; Suzuki, Armato et al. 2003; Suzuki and Doi 2005; Suzuki, Li et al. 2005; Suzuki, Yoshida et al. 2006).

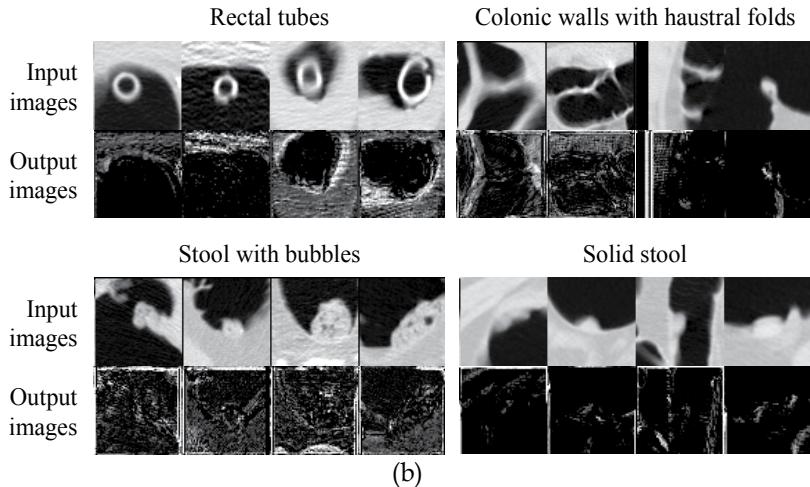
We trained eight expert 3D MTANNs with the ten polyps and ten non-polyps in each category. A three-layer structure was employed for the expert 3D MTANNs(Funahashi 1989). The size of the training volume and the standard deviation of the 3D Gaussian distribution in the teaching volume were 15 x 15 x 15 voxels (i.e., cubic shape) and 4.5 voxels, respectively, which were determined empirically based on our previous studies (Suzuki, Armato et al. 2003; Arimura, Katsuragawa et al. 2004; Suzuki and Doi 2005; Suzuki, Yoshida et al. 2006). The number of hidden units was selected to be 25 by use of a method for designing the structure of an ANN (Suzuki, Horiba et al. 2001; Suzuki 2004). With the parameters above, training of the expert 3D MTANNs was performed by 500,000 iterations. We selected four among the eight expert 3D MTANNs for the mixture of expert 3D MTANNs by experimental analysis, because the mixture of these four expert 3D MTANNs

((1) rectal tubes, (2) stool with bubbles, (3) colonic walls with haustral folds, and (4) solid stool) demonstrated the highest performance (described in the next subsection).

5.5. Evaluation of the Performance for False-Positive Reduction

We applied the trained expert 3D MTANNs to the 27 polyps (48 true-positive volumes) and all 224 non-training FPs identified by our previously reported CAD scheme. The output volumes for these testing cases are shown in Fig. 12. The centers of the input volumes corresponded to the detection results provided by the CAD scheme (including both true positives and FPs); thus, this experiment included the effect of actual off-centering of polyp candidates produced by the initial CAD scheme. Various polyps, including a sessile polyp (the third image from the left in Fig. 12(a)), are represented in the output by distributions of bright voxels, whereas various types of non-polyps appear as darker voxels, indicating the ability of the expert 3D MTANNs to enhance polyps and suppress different types of non-polyps. We applied the 3D scoring method to the output volumes for polyps and non-polyps. The 3D Gaussian weighting function used the same standard deviation as that for the 3D Gaussian distribution in the polyp teaching volume. Distributions of scores from the expert 3D MTANNs for the 27 polyps and all FPs are shown in Fig. 13. Although the two distributions in each graph overlap, a substantial fraction of FPs can be eliminated by use of the expert 3D MTANNs.





(b)

Fig. 12. Illustrations of (a) various testing polyps and the corresponding output volumes of four trained expert 3D MTANNs and (b) four different categories of testing FPs and the output volumes from corresponding expert 3D MTANNs. In the output volumes, polyps appear as distributions of bright voxels (i.e., they are enhanced), whereas different types of FPs appear as dark voxels (i.e., they are suppressed).

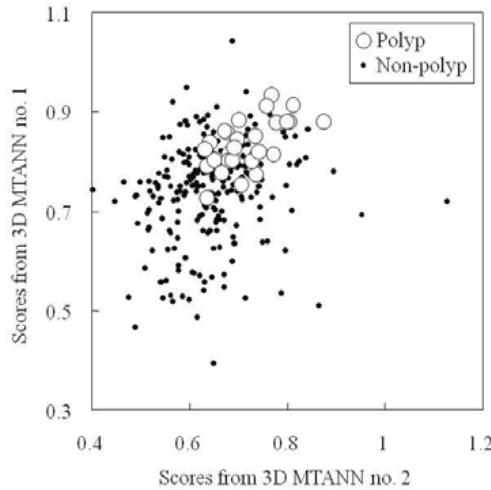


Fig. 13. Distributions of scores from MTANN nos. 1 and 2 in the mixture of expert 3D MTANNs for 27 polyps (white circles) and 224 non-polyps (black dots).

We evaluated the overall performance of the mixture of expert 3D MTANNs for FP reduction by use of free-response receiver-operating-characteristic (FROC) analysis (Egan, Greenberg et al. 1961). The FROC curve of the trained mixture of expert 3D MTANNs is shown in Fig. 14. The FROC curve was obtained by a change in the threshold value for the output of the mixing ANN. This FROC curve indicates that the mixture of expert 3D MTANNs was able to eliminate 63% (142/224) of non-polyps (FPs) without removal of any

of the 27 polyps, i.e., a 96.4% (27/28) overall by-polyp sensitivity was achieved at an FP rate of 1.1 (82/73) per patient.

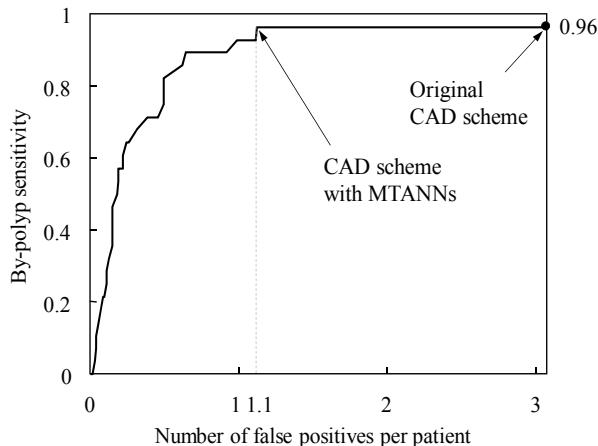


Fig. 14. The FROC curve that shows the overall performance of the mixture of expert 3D MTANNs when it was applied to the entire database of 27 polyps (48 true-positive volumes) and 224 FPs. The FROC curve indicates that the mixture of expert 3D MTANNs yielded a reduction of 63% (142/224) of non-polyps (FPs) without removal of any true positives, i.e., it achieved 100% (27/27 or 17/17) classification performance.

6. Conclusion

The MTANN supervised filter was effective for enhancement of lesions including lung nodules and colorectal polyps and suppression of non-lesions in medical images and was useful for improving the sensitivity and specificity of CAD schemes substantially.

7. References

- American:Cancer:Society (2005). Cancer Facts and Figures 2005. Atlanta, American Cancer Society.
- Arimura, H., S. Katsuragawa, et al. (2004). "Computerized scheme for automated detection of lung nodules in low-dose computed tomography images for lung cancer screening." Academic Radiology 11(6): 617-629.
- Armato, S. G., 3rd, M. L. Giger, et al. (2001). "Automated detection of lung nodules in CT scans: preliminary results." Medical Physics 28(8): 1552-1561.
- Armato, S. G., 3rd, M. L. Giger, et al. (1999). "Computerized detection of pulmonary nodules on CT scans." Radiographics 19(5): 1303-1311.
- Armato, S. G., 3rd, F. Li, et al. (2002). "Lung cancer: performance of automated lung nodule detection applied to cancers missed in a CT screening program." Radiology 225(3): 685-692.
- Chan, H. P., K. Doi, et al. (1987). "Image feature analysis and computer-aided diagnosis in digital radiography. I. Automated detection of microcalcifications in mammography." Medical Physics 14(4): 538-548.

- Dachman, A. H. (2003). *Atlas of Virtual Colonoscopy*. New York, Springer-Verlag.
- Dean, J. C. and C. C. Ilvento (2006). "Improved cancer detection using computer-aided detection with diagnostic and screening mammography: prospective study of 104 cancers." *AJR Am J Roentgenol* 187(1): 20-8.
- Drukier, K., M. L. Giger, et al. (2005). "Robustness of computerized lesion detection and classification scheme across different breast US platforms." *Radiology* 237(3): 834-40.
- Egan, J. P., G. Z. Greenberg, et al. (1961). "Operating characteristics, signal detectability, and the method of free response." *Journal of the Acoustical Society of America* 33: 993-1007.
- Flehinger, B. J., M. Kimmel, et al. (1992). "The effect of surgical treatment on survival from early lung cancer. Implications for screening." *Chest* 101(4): 1013-1018.
- Flehinger, B. J., M. R. Melamed, et al. (1984). "Early lung cancer detection: results of the initial (prevalence) radiologic and cytologic screening in the Memorial Sloan-Kettering study." *American Review of Respiratory Disease* 130(4): 555-560.
- Fletcher, J. G., F. Booya, et al. (2005). "CT colonography: unraveling the twists and turns." *Curr Opin Gastroenterol* 21(1): 90-8.
- Fontana, R. S., D. R. Sanderson, et al. (1984). "Early lung cancer detection: results of the initial (prevalence) radiologic and cytologic screening in the Mayo Clinic study." *American Review of Respiratory Disease* 130(4): 561-565.
- Frangi, A. F., W. J. Niessen, et al. (1999). "Model-based quantitation of 3-D magnetic resonance angiographic images." *IEEE Trans Med Imaging* 18(10): 946-56.
- Frimmel, H., J. Nappi, et al. (2004). "Fast and robust computation of colon centerline in CT colonography." *Medical Physics* 31(11): 3046-3056.
- Frost, J. K., W. C. Ball, Jr., et al. (1984). "Early lung cancer detection: results of the initial (prevalence) radiologic and cytologic screening in the Johns Hopkins study." *American Review of Respiratory Disease* 130(4): 549-554.
- Funahashi, K. (1989). "On the approximate realization of continuous mappings by neural networks." *Neural Networks* 2: 183-192.
- Giger, M. L., K. Doi, et al. (1988). "Image feature analysis and computer-aided diagnosis in digital radiography. 3. Automated detection of nodules in peripheral lung fields." *Medical Physics* 15(2): 158-166.
- Giger, M. L. and K. Suzuki (2007). *Computer-Aided Diagnosis (CAD)*. Biomedical Information Technology. D. D. Feng, Academic Press: 359-374.
- Gilhuijs, K. G., M. L. Giger, et al. (1998). "Computerized analysis of breast lesions in three dimensions using dynamic magnetic-resonance imaging." *Med Phys* 25(9): 1647-54.
- Gurney, J. W. (1996). "Missed lung cancer at CT: imaging findings in nine patients." *Radiology* 199(1): 117-122.
- Heelan, R. T., B. J. Flehinger, et al. (1984). "Non-small-cell lung cancer: results of the New York screening program." *Radiology* 151(2): 289-293.
- Henschke, C. I., D. I. McCauley, et al. (1999). "Early Lung Cancer Action Project: overall design and findings from baseline screening." *Lancet* 354(9173): 99-105.
- Henschke, C. I., D. P. Naidich, et al. (2001). "Early lung cancer action project: initial findings on repeat screenings." *Cancer* 92(1): 153-159.
- Horsch, K., M. L. Giger, et al. (2004). "Performance of computer-aided diagnosis in the interpretation of lesions on breast sonography." *Acad Radiol* 11(3): 272-80.

- Jemal, A., T. Murray, et al. (2005). "Cancer statistics, 2005." *CA Cancer J Clin* 55(1): 10-30.
- Jemal, A., T. Murray, et al. (2005). "Cancer statistics, 2005." *CA: A Cancer Journal for Clinicians* 55(1): 10-30.
- Johnson, C. D. and A. H. Dachman (2000). "CT colonography: the next colon screening examination?" *Radiology* 216(2): 331-341.
- Kaneko, M., K. Eguchi, et al. (1996). "Peripheral lung cancer: screening and detection with low-dose spiral CT versus radiography." *Radiology* 201(3): 798-802.
- Kobayashi, T., X. W. Xu, et al. (1996). "Effect of a computer-aided diagnosis scheme on radiologists' performance in detection of lung nodules on radiographs." *Radiology* 199(3): 843-848.
- Kubik, A. and J. Polak (1986). "Lung cancer detection. Results of a randomized prospective study in Czechoslovakia." *Cancer* 57(12): 2427-2437.
- Kupinski, M. A., D. C. Edwards, et al. (2001). "Ideal observer approximation using Bayesian classification neural networks." *IEEE Trans Med Imaging* 20(9): 886-99.
- Li, F., M. Aoyama, et al. (2004). "Radiologists' performance for differentiating benign from malignant lung nodules on high-resolution CT using computer-estimated likelihood of malignancy." *AJR. American Journal of Roentgenology* 183(5): 1209-1215.
- Li, F., H. Arimura, et al. (2005). "Computer-aided detection of peripheral lung cancers missed at CT: ROC analyses without and with localization." *Radiology* 237(2): 684-90.
- Li, F., S. Sone, et al. (2002). "Lung cancers missed at low-dose helical CT screening in a general population: comparison of clinical, histopathologic, and imaging findings." *Radiology* 225(3): 673-683.
- Macari, M. and E. J. Bini (2005). "CT colonography: where have we been and where are we going?" *Radiology* 237(3): 819-33.
- Miettinen, O. S. (2000). "Screening for lung cancer." *Radiologic Clinics of North America* 38(3): 479-486.
- Miettinen, O. S. and C. I. Henschke (2001). "CT screening for lung cancer: coping with nihilistic recommendations." *Radiology* 221(3): 592-596.
- Mosier, C. I. (1951). "Problems and designs of cross-validation." *Educational and Psychological Measurement* 11: 5-11.
- Nappi, J. and H. Yoshida (2002). "Automated detection of polyps with CT colonography: evaluation of volumetric features for reduction of false-positive findings." *Academic Radiology* 9(4): 386-397.
- Nappi, J. and H. Yoshida (2003). "Feature-guided analysis for reduction of false positives in CAD of polyps for computed tomographic colonography." *Medical Physics* 30(7): 1592-1601.
- Nawa, T., T. Nakagawa, et al. (2002). "Lung cancer screening using low-dose spiral CT: results of baseline and 1-year follow-up studies." *Chest* 122(1): 15-20.
- Oja, E. (1983). *Subspace methods of pattern recognition*. Letchworth, Hertfordshire, England; New York, Research Studies Press; Wiley.
- Otsu, N. (1979). "A Threshold Selection Method from Gray Level Histograms." *IEEE Transactions on Systems, Man and Cybernetics* 9(1): 62-66.
- Rumelhart, D. E., G. E. Hinton, et al. (1986). "Learning representations by back-propagating errors." *Nature* 323: 533-536.

- Sobue, T., T. Suzuki, et al. (1992). "Survival for clinical stage I lung cancer not surgically treated. Comparison between screen-detected and symptom-detected cases. The Japanese Lung Cancer Screening Research Group." *Cancer* 69(3): 685-692.
- Sone, S., F. Li, et al. (2001). "Results of three-year mass screening programme for lung cancer using mobile low-dose spiral computed tomography scanner." *British Journal of Cancer* 84(1): 25-32.
- Sone, S., S. Takashima, et al. (1998). "Mass screening for lung cancer with mobile spiral computed tomography scanner." *Lancet* 351(9111): 1242-1245.
- Suzuki, K. (2004). "Determining the receptive field of a neural filter." *Journal of Neural Engineering* 1(4): 228-237.
- Suzuki, K., H. Abe, et al. (2006). "Image-processing technique for suppressing ribs in chest radiographs by means of massive training artificial neural network (MTANN)." *IEEE Transactions on Medical Imaging* 25(4): 406-416.
- Suzuki, K., S. G. Armato, et al. (2003). Effect of a small number of training cases on the performance of massive training artificial neural network (MTANN) for reduction of false positives in computerized detection of lung nodules in low-dose CT. Proc. SPIE Medical Imaging (SPIE MI), San Diego, CA.
- Suzuki, K., S. G. Armato, et al. (2003). "Massive training artificial neural network (MTANN) for reduction of false positives in computerized detection of lung nodules in low-dose CT." *Medical Physics* 30(7): 1602-1617.
- Suzuki, K. and K. Doi (2005). "How can a massive training artificial neural network (MTANN) be trained with a small number of cases in the distinction between nodules and vessels in thoracic CT?" *Academic Radiology* 12(10): 1333-1341.
- Suzuki, K., I. Horiba, et al. (2001). "A simple neural network pruning algorithm with application to filter synthesis." *Neural Processing Letters* 13(1): 43-53.
- Suzuki, K., I. Horiba, et al. (2002). "Efficient approximation of neural filters for removing quantum noise from images." *IEEE Transactions on Signal Processing* 50(7): 1787-1799.
- Suzuki, K., I. Horiba, et al. (2003). "Neural edge enhancer for supervised edge enhancement from noisy images." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25(12): 1582-1596.
- Suzuki, K., I. Horiba, et al. (2004). "Extraction of left ventricular contours from left ventriculograms by means of a neural edge detector." *IEEE Transactions on Medical Imaging* 23(3): 330-339.
- Suzuki, K., F. Li, et al. (2005). "Computer-aided diagnostic scheme for distinction between benign and malignant nodules in thoracic low-dose CT by use of massive training artificial neural network." *IEEE Transactions on Medical Imaging* 24(9): 1138-1150.
- Suzuki, K., J. Shiraishi, et al. (2005). "False-positive reduction in computer-aided diagnostic scheme for detecting nodules in chest radiographs by means of massive training artificial neural network." *Academic Radiology* 12(2): 191-201.
- Suzuki, K., H. Yoshida, et al. (2008). "Mixture of expert 3D massive-training ANNs for reduction of multiple types of false positives in CAD for detection of polyps in CT colonography." *Med Phys* 35(2): 694-703.
- Suzuki, K., H. Yoshida, et al. (2006). "Massive-training artificial neural network (MTANN) for reduction of false positives in computer-aided detection of polyps: Suppression of rectal tubes." *Medical Physics* 33(10): 3814-3824.

- Swensen, S. J., J. R. Jett, et al. (2003). "Lung cancer screening with CT: Mayo Clinic experience." *Radiology* 226(3): 756-761.
- van Ginneken, B., B. M. ter Haar Romeny, et al. (2001). "Computer-aided diagnosis in chest radiography: a survey." *IEEE Transactions on Medical Imaging* 20(12): 1228-1241.
- Winawer, S. J., R. H. Fletcher, et al. (1997). "Colorectal cancer screening: clinical guidelines and rationale." *Gastroenterology* 112(2): 594-642.
- Yoshida, H. and A. H. Dachman (2005). "CAD techniques, challenges, and controversies in computed tomographic colonography." *Abdom Imaging* 30(1): 26-41.
- Yoshida, H., Y. Masutani, et al. (2002). "Computerized detection of colonic polyps at CT colonography on the basis of volumetric features: pilot study." *Radiology* 222(2): 327-36.
- Yoshida, H. and J. Nappi (2001). "Three-dimensional computer-aided diagnosis scheme for detection of colonic polyps." *IEEE Trans Med Imaging* 20(12): 1261-74.
- Yoshida, H. and J. Näppi (2001). "Three-dimensional computer-aided diagnosis scheme for detection of colonic polyps." *IEEE Trans Med Imaging* 20(12): 1261-74.

Automated detection and analysis of particle beams in laser-plasma accelerator simulations

Daniela M. Ushizima¹, Cameron G. Geddes¹, Estelle Cormier-Michel¹,
E.Wes Bethel¹, Janet Jacobsen¹, Prabhat¹, Oliver R ubel^{1,2,3}, GuntherWeber^{1,2}
and Bernd Hamann^{1,2}

¹*Lawrence Berkeley National Laboratory, Berkeley, CA*

²*University of California, Davis, CA
USA*

Peter Messmer
*Tech-X Corporation, Boulder, CO
USA*

Hans Haggen
*University of Kaiserslautern
Germany*

1. Introduction

Numerical simulations of laser-plasma wakefield (particle) accelerators [Geddes et al. (2009); Tajima & Dawson (1979)] model the acceleration of electrons trapped in plasma oscillations (wakes) left behind when an intense laser pulse propagates through the plasma. The goal of these simulations is to better understand the process involved in plasma wake generation and how electrons are trapped and accelerated by the wake as in Geddes (2005); Geddes et al. (2004); Pukhov & ter Vehn (2002); Tsung et al. (2007; 2004). Understanding of such accelerators, and their development, offers high accelerating gradients, potentially reducing size and cost of new accelerators.

One operating regime of interest is where a trapped subset of electrons loads the wake and forms an isolated group of accelerated particles with low spread in momentum and position [Geddes et al. (2004)], desirable characteristics for many applications. The electrons trapped in the wake may be accelerated to high energies, the plasma gradient in the wake reaching up to a gigaelectronvolt per centimeter [Tajima & Dawson (1979)]. High-energy electron accelerators power intense X-ray radiation to terahertz sources, and are used in many applications including medical radiotherapy and imaging [Geddes et al. (2009)].

To extract information from the simulation about the quality of the beam, a typical approach is to examine plots of the entire dataset, visually determining the parameters necessary to select a subset of particles, which is then further analyzed. This procedure requires

laborious examination of massive data sets over many time steps using several plots, a routine that is unfeasible for large data collections. Demand for automated analysis is growing along with the volume and size of simulations. Current 2D LWFA simulation datasets are typically between 1GB and 100GB in size, but simulations in 3D are of the order of TBs. The increase in the number of datasets and dataset sizes leads to a need for automatic routines to recognize particle patterns as particle bunches (beam of electrons) for subsequent analysis as in Ushizima et al. (2008).

Because of the growth in dataset size, the application of machine learning techniques for scientific data mining is increasingly considered. In plasma simulations, Bagherjeiran & Kamath (2006) presented a comprehensive report on applying graph-based techniques for orbit classification. They used the KAM classifier as in Yip (1991) to label points and components in single and multiple orbits. Love & Kamath (2007) conducted an image space analysis of coherent structures in plasma simulations. They used a number of segmentation and region-growing techniques to isolate regions of interest in orbit plots. Both approaches analyzed particle accelerator data, targeting the system dynamics in terms of particle orbits. However, they did not address particle dynamics as a function of time or inspected the behavior of bunches of particles.

A visual analysis of massive laser wakefield acceleration (LWFA) simulation data was addressed by Rübel et al. (2008), using interactive procedures to query the data. Sophisticated visualization tools were provided to inspect the data manually. Rübel et al. have integrated these tools to the visualization and analysis system VisIt (2009), in addition to utilizing efficient data management based on HDF5 [Adelmann et al. (2007; 2005); Gosink et al. (2006); H5Part (2009)], and the index/query tool FastBit [FastBit (2009)]. Rübel et al. (2009) proposed automatic beam path analysis using a suite of methods to select particles in simulation data and to analyze their temporal evolution. To enable researchers to accurately define particle beams, the method computes a set of measures based on the path of particles relative to the distance of the particles to a beam. To achieve good performance, this framework uses an analysis pipeline designed to quickly reduce the amount of data that needs to be considered in the actual path distance computation. As part of this process, region-growing methods are utilized to detect particle bunches at single time steps. Efficient data reduction is essential to enable automated analysis of large data sets as described in the next section, where data reduction methods are steered to the particular requirements of our clustering analysis.

Ushizima et al. (2008) first described the application of a set of algorithms to automate the data analysis and classification of particle beams in the LWFA simulation data, identifying locations with high density of high energy particles. These algorithms detected high density locations (nodes) in each time step, i.e. maximum points on the particle distribution for only one spatial variable. Each node was correlated to a node in previous or later time steps by linking these nodes according to a pruned minimum spanning tree (PMST). We call the PMST representation “a lifetime diagram”, which is a graphical tool to show temporal information on highly dense groups of particles in the longitudinal direction for the time series. Electron bunch compactness was described by another step of the processing, designed to partition each time step, using fuzzy clustering, into a fixed number of clusters. We combined the lifetime diagram with the clustering results to locate spatially confined beams, demonstrating the ability of the method to detect high quality beams, characterized by high energy and high degree of spatial coherence. A reported drawback of the method in Ushizima et al. (2008) is the inability to detect low energy beams due to the mechanism of privileging high energy

particles, therefore outputting incorrect scattered groups of particles with high energy instead of compact group of particles with low energy.

This paper extends previous work by addressing beam detection, independent of the energy. We divide the investigation in two main steps: (a) detection of maximum density regions of particles using both longitudinal and transversal direction of variation and (b) multidimensional particle clustering over each time step to automatically detect isolated bunches of electrons within resulting partitions. We calculate these partitions using normal mixture models, followed by the selection of the best model according to a cluster compactness criteria. Each clustering algorithm uses a multivariate analysis to identify high density groups of electrons, iteratively searching for the best number of clusters to model the particle distribution. We employ data representation and partitioning to detect electrons undergoing acceleration, using the powerful R statistical tools from Gentleman & Ihaka (2009) and have integrated the data management method of Fastbit [FastBit (2009)] into the R analysis framework. Our main contribution is the automatic detection of compact groups of particles from large, complex and time-dependent scientific data sets of electron simulations. These groups are selected for coherence in both momentum and spatial coordinates, which are characteristic of electron beams that one wants to identify. In addition, we propose several graphical representations of data for fast information assessment that will help guide later feature extractors to derive simulation measurements. Our results show that the proposed framework can detect group of particles that belong to the electron beam even if the particle bunch presents low energy. This is important to allow comparison among many simulation runs with varying beam quality. We are able to automatically detect the beam and characterize it in terms of dispersion measurements, that identifies the time steps where the bunch is most condensed and under acceleration.

The next section (Sec.2) describes the datasets under investigation, the proposed approach and implementation details. Sec.3 presents the results of combining data transformation, geometrical modeling and analysis with classification of electrons from simulation time series. We conclude with discussions and future directions in Sec.4.

2. Material and methods

2.1 Particle acceleration simulations

LWFA simulations are used to model physical parameter variations, such as tuning of laser energy and plasma characteristics, in order to determine the combination that will achieve the desired small energy spread bunch, hence guiding and improving understanding of laboratory experiments. The simulations model the properties of a hydrogen plasma, which is an ionized gas containing free electrons (not bound to a molecule) and positively charged ions. As a laser pulse travels through the plasma, the electric field of the light separates electrons and positively charged ions. While the positive ions are heavy and stay in place, the light electrons are pushed away from the laser pulse creating a “bubble” of positively charged particles behind the pulse as in Geddes et al. (2004); Tsung et al. (2004). In this so-called blowout regime [Pukhov & ter Vehn (2002)], a fraction of electrons can be trapped in the wave and be accelerated, in the same direction as the laser pulse, until they outrun the wave. As the accelerating structure travels at a speed less than the speed of light, the relativistic electrons eventually slip into a decelerating region of the wake, stopping the acceleration process.

The most common algorithm used to simulate LWFA is particle-in-cell (PIC) codes [Birdsall et al. (1991)]. The PIC technique models the dynamics of particles and the electromagnetic field in a simulation window that travels at the speed of light [Geddes et al. (2008; 2004); Tsung

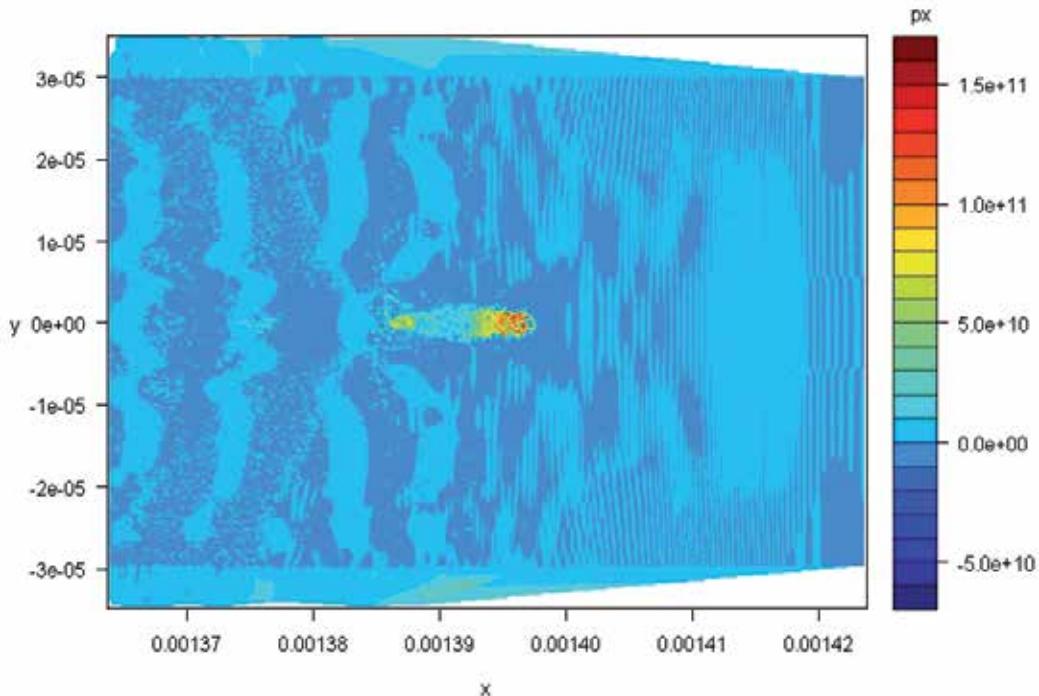


Fig. 1. Level sets as a function of the momentum of simulation particles, for a time step known a priori to contain a high quality beam (red), characterized by high momentum (high px) in a compact envelope, given by x, y position.

et al. (2007)]. The properties of accelerated particle groups vary (e.g., duration in time, position, momentum and momentum spread) and, rather than being prescribed as inputs, are a consequence of a set of parameters defined before the simulation starts, similarly to laboratory experiments.

The simulation data analyzed in this paper were produced by the massively parallel plasma simulation code VORPAL [Nieter & Cary (2004)] developed by Tech-X Corporation and the University of Colorado at Boulder. VORPAL offers a broad range of models and algorithms for treating the interaction of charged particles with the electromagnetic field, including an electromagnetic or electrostatic [Messmer & Bruhwiler (2006)] field solvers, and kinetic, fluid and hybrid models for the plasma. While originally developed for modeling laser-wakefield accelerators, VORPAL's flexibility has enabled it to be applied to a broad range of problems, including electromagnetic cavities, laser-solid interaction, investigations of breakdown in high-power wave guides, or electron cooling concepts, just to mention a few. VORPAL was designed for parallelism and scalability, and it runs routinely on tens of thousands of processors on leadership-class supercomputers at various National Supercomputing Centers. Using HDF5 as the native data format, VORPAL's output can be processed with the powerful analysis tools like the ones reported here.

One way to identify simulation particles that were trapped at some point in time is by looking later on at the particles that have high energy. The particles with the highest energy levels are usually located in the first wake period, forming a compact bunch, as illustrated

by the red region in Figure 1. The yellow level sets (center) show particles that can also be accelerated in wave buckets that follow the first wake. The selection of the particles of interest (the highly accelerated ones) has typically been done by looking at the later time-steps of a simulation and interactively selecting only those particles with a velocity that is larger than a defined threshold [Geddes (2005); Rübel et al. (2008); Tsung et al. (2007; 2004)].

2.2 Simulation datasets

This chapter investigates datasets from 2D simulations that contain the (x, y) position of the particles as well as their momentum in the x and y directions (px, py). The laser pulse and accelerated particles propagate in the x -direction. Each simulation particle represents a group of electrons, with weight (wt). Since no identifiers (id) are stored for the particles, the particle weights are used as identifiers throughout the analysis. If several particles have the same weight in the simulation, these are not traced. Table 1 presents details of the datasets used in our tests, from which we draw only traceable particles (unique identifier).

Dataset	Particles (10^6)	Timesteps	Total Size (GB)
A	0.4	37	1.3
B	1.6	37	4.5
C	0.4	38	1.3
D	3.2	45	11
E	6	39	28

Table 1. Tested 2D simulation datasets.

Data access requires efficient readers, provided by H5Part [H5Part (2009)]. H5Part is a veneer API on top of HDF5 that considerably simplifies reading and writing simulation data to HDF5 files. In order to efficiently query large particle datasets, we utilize the capabilities of FastBit, a state-of-the-art index/query system as in FastBit (2009); Wu et al. (2004; 2006). FastBit resolves queries in a time proportional to the number of hits satisfying the query. This capability is essential when dealing with datasets of hundreds of millions of particles, where the interesting particles might only number in the hundreds or thousands. A naive (non-indexed) scheme would need to load up the entire dataset to resolve the query, which is prohibitively expensive for large datasets. This index/query system supports conditional queries, e.g. “detect all particles such that $(px > 1e10) \& (x > 0) \& (y > 5)$ ”; this can be used to select particles with interesting characteristics in multi-dimensional phase space. FastBit can also track selected particles across time steps by issuing queries of the form $id \in (5, 10, 31)$, which pulls out data for the three specific particles.

FastBit indices are stored within H5Part files and accessed using a custom C++ interface called HDF5-FastQuery [HDF5-FastQuery (2009)]. All our analysis software is written in R. Therefore, in order to utilize FastBit’s functionality within the R runtime, we extended the RcppTemplate package [Samperi (2006)] to make function calls to the HDF5-FastQuery interface. This saves us considerable time to load subsets of particle data, at least 6.6 times faster than R-package *hdf5*. This is a considerable improvement over existing HDF5 packages in R, which often constrain the user to load the entire HDF5 file or complete groups within the file. In addition to efficient data access, our framework implements data reduction by using physical domain knowledge, data analysis algorithms and clustering techniques as described in the following sections.

2.3 Proposed framework

Unlike image data, composed of pixel values at regular spaces, laser wakefield simulations contain particles irregularly spaced in all dimensions. Scattered data is a common problem in scientific data mining when trying to extract patterns in large datasets, particularly because the physical phenomenon is evolving over time [Kamath (2009)]. Data reduction of large datasets is often mandatory before applying clustering algorithms due to their inherent combinatorial complexity. Figure 2 shows our framework for detection of accelerated electron bunches in LWFA simulations; the algorithms for data partitioning and pattern detection are detailed in the next sections.

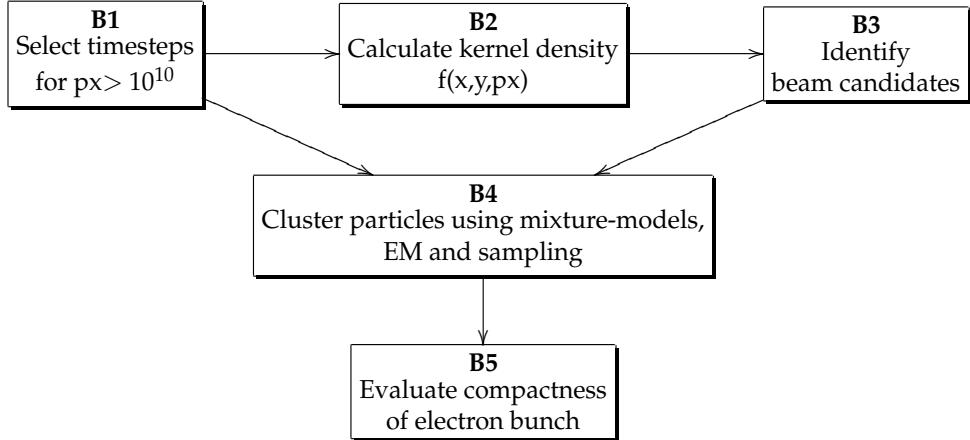


Fig. 2. Framework for data reduction and beam detection applied to each time step of data sets generated by laser wakefield acceleration (LWFA) simulations.

The first step (B1) selects particles and time steps relevant for inspection from a n -time step simulation dataset, discarding those particles unlikely to belong to the physical phenomenon of interest. The pipeline obtains particle distribution (B2), using kernels to calculate an estimate ($f(x,y,px)$) of the probability density function. Next, we find parameters x, y, px for which f is maximum, selecting a subset of particles that may correspond to trapped bunches of electrons (B3). The following step (B4) then groups the simulation particles according to normal mixture models, before applying maximum likelihood estimation and Bayes criteria. The goal is to identify the most likely model and number of clusters that better refine the previous beam candidate particles. The simulation contains several time steps and varying number of particles per time step; we combine the result of beam detection for each time step and calculate statistics of the time series by applying moving averages (B5) to characterize the electron bunch.

2.3.1 High energy particles and densities (B1-B2)

Block B1 performs particle selection given a threshold in momentum in the x -direction, based on the fact that the bunch of electrons of interest should be observed near $px = 10^{11}$. We can then eliminate the low energy particles for which $px < 10^{10}$. The expected wake oscillation is up to $px = 10^9$. Therefore this threshold excludes particles of the background plasma, while

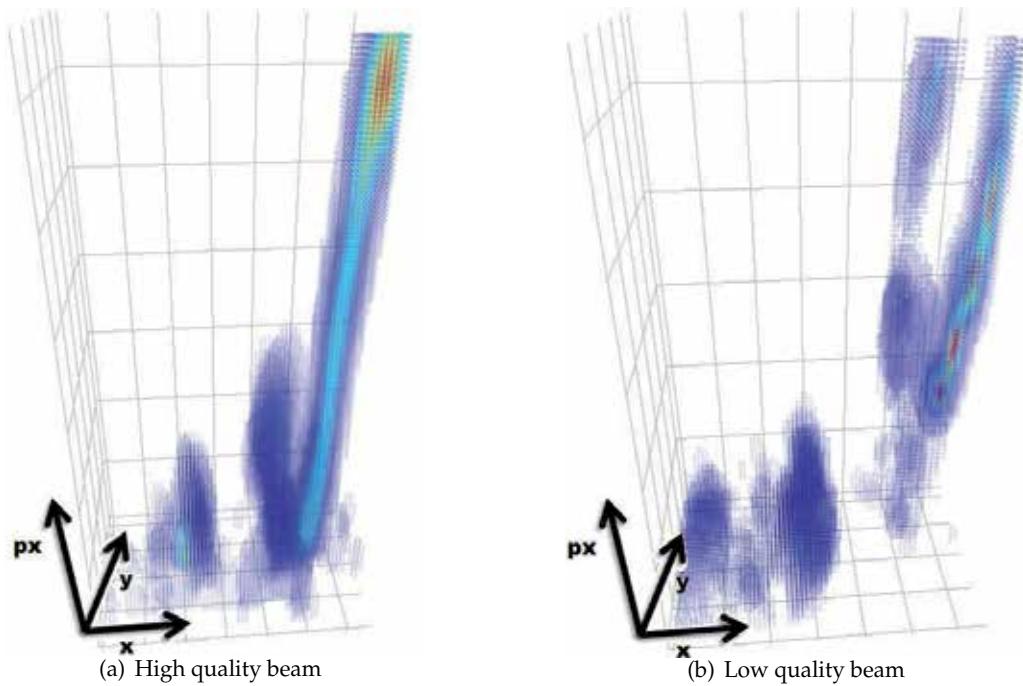


Fig. 3. Kernel density estimation of Dataset A (left) and D (right) at a single time step, showing high-density bunches (red): 3D representation of $f(x, y, px)$ with heatmap colors representing the particle density.

including all particles that could be in an accelerated bunch. The precise choice of the threshold does not affect the result accuracy and a lower threshold could be used at higher computational cost [Ushizima et al. (2008)]. After eliminating low momentum particles, some time steps (in general the first few time steps of the simulation) may not include a relevant amount of particles for inspection. We calculate the simulation average number of particles above threshold on p_x (μ_s) to determine the “representative” time steps, t_i , for which there is a number of particles greater than μ_s , determining a smaller subset of time steps. We observed that this constraint eliminates initial time steps, but maintains consecutive time steps throughout the time series from t_{μ_s} , the first time step for which the number of particles is greater than μ_s . Again, this threshold can be adjusted to lower values.

It is necessary to compute the density of the particles given the (x, y, px) parameters of the particles in each time step. The most widely used nonparametric density estimator is the histogram, whose main disadvantage is its sensitivity to the placement of the bin edges, a problem not shared by kernel density estimators, as described in Wand & Jones (1995). Kernel density estimators are hence a valuable tool to identify subgroups of samples with inhomogeneous behavior and to recover the underlying structure of the dataset, while minimizing binning artifacts. The PDF estimation also depends on the number of particles and a set of smoothing parameters called bandwidth [Weissbach & Gefeller (2009)].

We estimate the probability density function (PDF) $f(x, y, px)$ for time steps $t = [t_{\mu_s}, T]$ in B2, where T is the original number of time steps in the simulation, before extracting beam

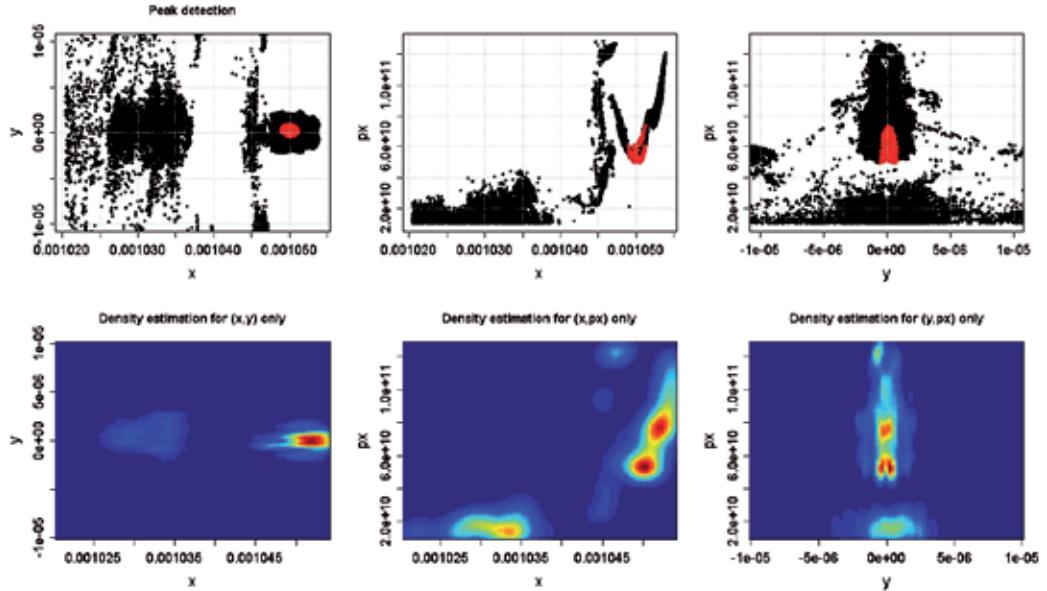


Fig. 4. Projections of beam candidate region detection (block B3) from timestep in dataset D (top) and 2D particle density estimations (bottom) to confirm compactness of selected particles.

candidate regions. An estimation of the PDF is calculated using kernel density estimation [Feng & Tierney (2009)] and defined on a grid with spacing $\Delta x = 0.5\mu\text{m}$, $\Delta y = 0.5\mu\text{m}$ and $\Delta px \approx 10^9$. These parameters are selected based on the physical expectation of electron beam size to be $2\mu\text{m}$ and momentum spread to be approximately 10^{10} [Geddes (2005); Geddes et al. (2004); Pukhov & ter Vehn (2002); Tsung et al. (2004)]. This PDF will be used later for retrieval of the maximum value and the first adjacent bins.

Figure 3 shows 3D densities that are the result of the calculated multivariate kernel density estimators and illustrates the concepts of high and low quality beams. Notice that Fig.3(a) presents a concentrated region in (x, y, px) and higher values of px in comparison with Fig.3(b), which has scattered (red) groups with lower values of px , indicating a low-quality beam. Next, we propose a method to detect these groups of particles, independently of the range of energies they present.

2.3.2 Deriving maximum peaks (B3)

The task in B3 is to find particles at maximum values of f and their immediate vicinity to obtain compact electron bunches in space and with limited dispersion in momentum, as emphasized in red in Figure 4. These criteria determine the ability to characterize the quality of particle beams, which depends on the grouping of electrons in terms of their spatial parameters as well as momentum in the longitudinal (x) and transverse (y) directions. The binning used to calculate f may interfere in the beam quality descriptors if only the absolute maximum of the PDF is taken into account, e.g., the bins may separate a maximum peak into parts if the binning is too small to contain the particles of interest. To prevent this undesirable effect, we adopt a tolerance parameter to select compact bunches and extract more than one

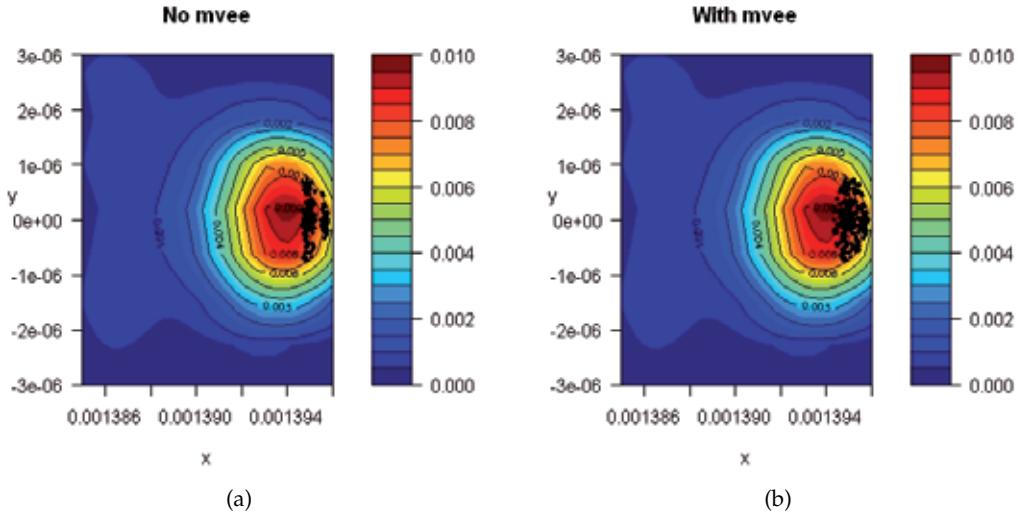


Fig. 5. Comparison of particle selection with/without MVEE: extracting the orientation and the axes of an enclosing ellipse from (a) produces (b), increasing the number of particles from 173 to 263. Colors indicate the density of particles, using only (x, y) -coordinates, and black dots show potential particles to belong to the beam, according to the different methods.

maximum (beam candidate region) per time step. In addition, this is a way of accruing more samples and detecting secondary beams when these are almost as prominent as the primary beam, associated to the maximum of f .

During the searching for values that are approximately equal to $\max(f)$, we keep not only the maximum, but all bins where $f \geq u * \max(f)$, where u is an uncertainty or tolerance parameter, here empirically set to 0.85. While this value enables the detection of the main and the secondary beams (when present), lower values of u could be used to control the amount of particles to be selected at a lower accuracy of beam position. From this point, we refer to the subset of particles conditioned to $u * \max(f)$ and its adjacency, calculated for each time step, as "beam candidates".

Figure 4 (top) presents projections of Figure 3.b with their calculated beam candidates emphasized in red. These are the result of our first attempt to improve particle selection by using an algorithm known as minimum volume enclosing ellipsoid as in Khachiyan & Todd (1993), which is able to enclose previously selected particles and to include others based on a geometrically defined polytope. Figure 5 illustrates the algorithm when applied to LWFA data, showing the selected particles as black dots; these particles are not in the most dense region (red) once the colors refers to (x, y) -density calculation. When including compactness in px , the most dense region happens further ahead. As distinct from calculating center of mass and forcing an *ad hoc* diameter or semi-major/minor axes, the minimum volume enclosing ellipsoid (MVEE) algorithm [Khachiyan & Todd (1993); Kumar & Yildirim (2005); Moshtagh (2009)] takes the subset of points and prescribes a polytope model to extrapolate a preliminary sub-selection to other particles likely to be in the bunch. The MVEE algorithm is a semidefinite programming problem and consists of a better approximation to the convexity of subsets of

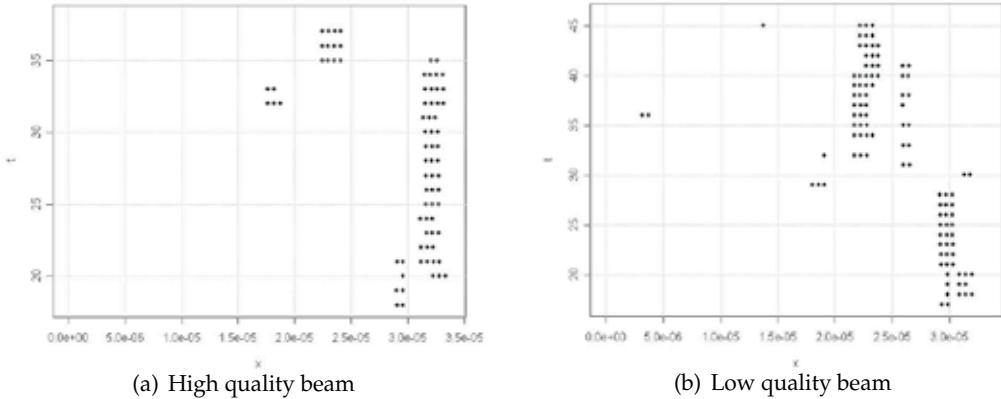


Fig. 6. Lifetime diagram of peaks, center of beam candidates, evolving in time for Dataset A (left) and D (right) for all time steps.

particles that correspond to compact groups of electrons. After querying hypervolumes similar to the one in Figure 3, we applied the geometrical model to adjust the particle selection as illustrated in Figure 5. By running the MVEE algorithm, we determine an ellipse as compact as possible covering the data points of the beam candidate region, increasing the number of samples without increasing the binning parameters. Here, we consider the problem of finding a MVEE, that minimizes the logarithm of the determinant of H such that

$$(x - c)^T H(x - c) \leq n \quad (1)$$

for a set of points x in R^n , an ellipsoid with center c and shape H [Khachiyan & Todd (1993)]. Further discussions on optimization of this algorithm can be found in Ahipasaoglu et al. (2008).

In addition to methods to select beam particles and graphics for each time step, it is often useful to track the bins occupied by the beam candidates by using lifetime diagrams. This diagram show the whole simulation, i.e. a global representation of the temporal evolution of beam candidate in bins. The diagram relates the time steps (t) to the relative position in the simulation window (x), which corresponds to the maximum of f for each time step of the simulation, as calculated in block B3. Figure 6(a) shows earlier time steps containing a bunch of particles that remains at constant speed, with dispersion around $t = 32$ and formation of a second bunch around $t = 35$. Figure 6(b) also shows time steps with formation of particle bunches that remains at constant speed earlier in the simulation, but the group disperses around $t = 28$, followed by the formation of a second bunch around $t = 34$.

The algorithms described in this section focused on the location of a subset of simulated particles, expected to be in the maximum of a multivariate density distribution, dependent on the particle properties. The next section complements the search for the beam by partitioning each time step to find subsets of particles, according to statistical modeling and clustering techniques.

2.3.3 Clustering particles (B4)

Data partitioning is a conceptually intuitive method of organizing simulation particles into similar groups given the absence of class labels. Since clustering is an unsupervised learning

method, evaluation and cluster quality assessment are valuable in interpreting classification results. We include both clustering methods and cluster validity techniques applied to particle acceleration to illustrate the applicability of dispersion measures to accurately evaluate an intrinsic structure, namely, a coherent bunch of electrons.

In order to determine the number of clusters in each time step of the simulation while testing statistical models with different numbers of components, we perform cluster analysis using model-based clustering, where the model and the number of clusters are selected at run time by *mclust* [Fraley & Raftery (2009)]. The model-based clustering algorithm postulates a statistical model for the samples, which are assumed to come from a mixture of normal probability densities. The calculation of normal mixture models considers different covariance structures and different number of clusters [Haughton et al. (2009)] given an objective function (score). The assumption of the number of clusters k entails a loss of generality, so we consider a range of k in addition to parameters that control the shape of the class. These parametric models are flexible in accommodating data as shown in Fraley & Raftery (2002) and consider widely varying characteristics in estimating distributions.

By assuming a normal mixture model, we represent the data d , with n samples and k components, considering a τ_k probability that an observation belongs to the k th component and a multivariate normal distribution φ_k with mean vector μ_k and covariance matrix Σ_k . The likelihood of d with n independent multivariate observations, represented by a Gaussian mixture model with G multivariate mixture components [Fraley & Raftery (2007)] is

$$\prod_{i=1}^n \sum_{k=1}^G \tau_k \varphi_k(d_i | \mu_k, \Sigma_k) \quad (2)$$

with priors conditioned to

$$\tau \geq 0; \sum_{k=1}^G \tau_k = 1. \quad (3)$$

The maximum likelihood estimate uses expectation-maximization (EM) methods, which rely on iterative two-fold processing: an E-step for calculating the conditional probability that an observation belongs to a certain group given the parameters θ_k , and a M-step for computing the parameters that maximize the log-likelihood given the previously calculated conditional probability function [Fraley & Raftery (2002)]. In other words, EM determines the most likely parameters $\theta_1, \dots, \theta_k$ to represent a problem consisting of multivariate observations given by a mixture of k underlying probability distributions [Fraley & Raftery (2006)].

The size of the LWFA datasets can compromise the efficiency of mixture model-based algorithms due to *mclust* initialization [Fraley & Raftery (2002)], then we propose a random sampling technique before calculating the mixture model. To illustrate such algorithm, Figure 7 uses artificial data, generated by two normal distributions $g1(x, y)$ and $g2(x, y)$ with 100 unlabeled samples each (Figure 7.a). In this example, we subsample the data by extracting a quarter of its original samples and calculate mixture-models, varying the structure and the number of the clusters (Figure 7.b). The result of the clustering provides labels for a quarter of the samples (black and red dots in Figure 7.c) and these labels support a supervised learning to classify the remaining samples as in Figure 7.d, a generalization procedure to extrapolate the “learned” models to the full dataset by using expectation-maximization.

Instead of imposing k , which is not known a priori, the objects are associated to each other according to a score that comes from the parameters, unknown quantities to be estimated from the probability distributions [Vermunt & Magidson (2002)]. Figure 7.b shows the calculation

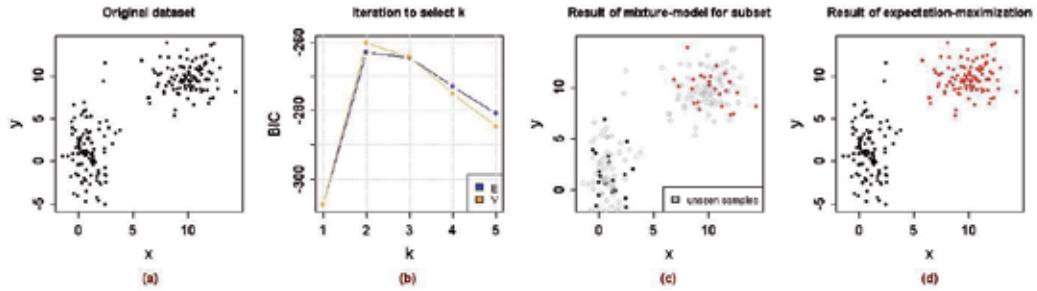


Fig. 7. Example of model-based clustering to clouds of points: (a) two Gaussian distributions with no label assignment; (b) Bayesian information criteria calculation for different number of clusters (k) and different models (E=equal volume and V=varying volume); (c) result of classification using subset (25%) of the data; (d) generalization of model using expectation-maximization.

of a score for different k and the maximum value of the curves imply the number of k the best describe the samples.

This process establishes the inference on the sample rather than on the full population [Fraley & Raftery (2002)]. This decision circumvents the bottleneck of the *mclust* initialization by a sampling strategy to partition large datasets: we propose a biased sampling process that ensures that the beam candidate region is in the sampled subset by guaranteeing that at least 10% (empirically chosen) of the samples belong to the high density particle volumes. We cluster the particles using the normal mixture models for values of $k \in [1, 10]$ to follow an ellipsoidal model with variable volume (VEV) [Banfield & Raftery (1993); Fraley & Raftery (2009)]. We have tested other models as spherical, diagonal and ellipsoidal, which can have equal or varying volume and shapes. However, VEV was the best algorithm for most of the time steps in all the datasets, according to the Bayesian information criteria [Fraley & Raftery (2009); Greene et al. (2008)]. We re-run the experiments to have clustering results using VEV only, but a varying number of k . The resulting clusters from VEV are considered as the training set to classify all the remaining samples by using EM to extrapolate parameters from training samples.

The result of the clustering (B4) is combined with B3 by calculating the intersection between the beam candidates and the a cluster that contains most of the particles from the beam candidates. In other words, we determine which cluster is most likely to contain the beam candidates by majority voting among all possible clusters, finalizing the tasks in block B4. The block B5 only analyzes the most compact group of particles that remains in the each time step.

2.3.4 Cluster quality assessment (B4-B5)

One of our goals in investigating particle simulations is to detect the electron beam and to characterize the dispersion of its particles in terms of spatial and momentum variables using clustering algorithms. Since we do not know a priori the number of clusters that best describe the particle grouping, we need some measure of goodness of fit to evaluate different clustering algorithms. A standard approach is to obtain the number of clusters (k) by maximizing a criterion function and to repeat the clustering procedure for different number of clusters.

We select k by maximizing the Bayesian information criterion (BIC) for a parametrized clustering algorithm using mixture models, following an ellipsoidal, varying volume model. The optimal BIC value considers the log-likelihood, the dimension of the data, and the number of mixture components in the model. The criterion function must describe how well a given clustering algorithm can match the data, defined as a function of the variable k .

Herein we will evaluate the goodness-of-fit of the clustering algorithms for k groups of particles from each time step using BIC to guide model selection for a set of parameterized mixture models with a varying number of classes. BIC adds a penalty to the log-likelihood [Fraley & Raftery (2006)] by considering the number of parameters in a certain model M and the number of observations (n) in the data set, with the form

$$BIC \equiv 2 \loglik_M(d, \theta_k^*) - (\#params)_M \log(n) \quad (4)$$

where $\loglik_M(d, \theta_k^*)$ is the maximized log-likelihood of the model with estimated parameters θ_k^* from the observations d and $(\#params)_M$ number of independent parameters to be estimated in M .

In addition to the evaluation of the clustering method (B4), we also want to verify if our framework can capture the physical phenomena of trapping and acceleration, when the beam is expected to be more compact. We propose the inspection of the particles in adjacent time steps using moving averages [Shumway & Stoffer (2006)] to identify if the electrons are grouped into stable bunches (B5).

The moving averages technique provides a simple way of seeing patterns in time series data, smooths out short-term fluctuations and highlights longer-term trends. This is physically motivated as the bunches of interest move at speed approximately equal to the speed of light, and hence are nearly stationary in the moving simulation window. We intersect particle bunches (b) at adjacent time steps, selecting the particles with the same identifier (id) and calculate statistical parameters (ρ) of a three-point moving average (mv_k), using the following algorithm:

```

 $k \leftarrow 1$ 
for  $t = 2$  to  $n-1$  do
   $id_k \leftarrow id(b_{t-1}) \cap id(b_t) \cap id(b_{t+1})$ 
   $mv_k \leftarrow (b_{t-1}|_{id_k} + b_t|_{id_k} + b_{t+1}|_{id_k})/3$ 
   $\rho \leftarrow statistics(mv_k)$ 
   $k \leftarrow k + 1$ 
end for

```

The particles of the bunch at time step $t - 1$, b_{t-1} , indexed by id_k , are called $b_{t-1}|_{id_k}$ and the function *statistics* calculates parameters such as the mean, variance and maximum values from the moving averages. We use the plots in Figure 10, 13 and 14 to check the persistence of particle bunches by looking at the evolution of statistical parameters as discussed in the next section.

3. Results

Here, we apply the above-described algorithms to analyze laser-plasma wakefield acceleration simulations, using the clustering techniques as part of a completely automated pipeline to detect dense particle groups (“electron bunches”). The main contributions of this work, in

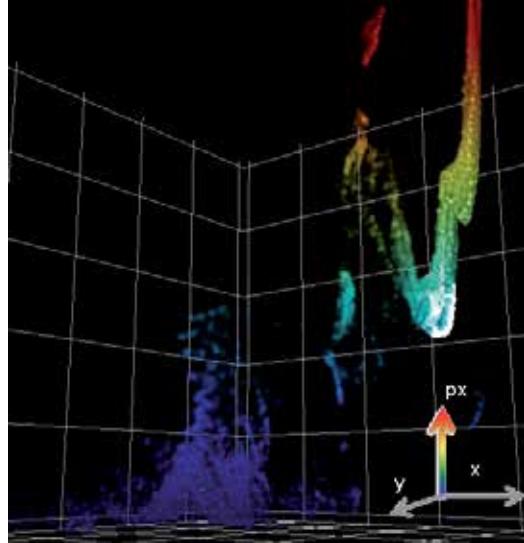


Fig. 8. Result of locating high-density bunches for one time step of dataset D: 3D scatter plot of particles, color is proportional to the particle energy (px) and white blobs correspond to the preliminary detection of beam candidate region as described in Sec.2.3.2.

comparison with the previous approach in Ushizima et al. (2008), are that we can perform beam detection independent of the quality or energy of the beam. Thus compact groups of particles can have either high momentum or low momentum, instead of only being able to correctly detect groups of particles exhibiting high momentum. This improvement stems from determining particle distribution using kernel density estimators, which minimizes the sensitivity of bin size assumptions and placement, enabling accurate detection of maximum values of $f(x, y, px)$. This is in contrast with the previous method that considered f as function of x only. Also, while Rübel et al. (2008) relies on user interaction, here we automatically detect compact groups of particles under acceleration.

We show that using the particle x -coordinate relative to the window size, we can keep track of the maximum values of the kernel density functions and represent these points using lifetime diagrams. Figure 6 shows the evolution of peaks from $f(x, y, px)$, which will support future work to restrict the search for compact bunches using clustering to specific regions around the maximum values. Figure 8 illustrates the result of identifying beam candidate from block B3 at a single time step from the dataset D, showing the (x, y, px) -coordinates of particles and respective detected compact groups. The beam candidate region is represented by a cloud of white dots, containing all the particles for which $0.85 * \max(f)$ holds.

The application of geometrical models such as the MVEE to enclose the detected beam candidates shows how structure assumptions may interfere in the number of particles selected, as illustrated in Figure 5. The advantage of this method is that it expands a previous restrictive selection to other neighboring points that should be included in the beam candidate region. As opposed to an approach that sets a fixed diameter, it also avoids an undesirable impact on the particle spread. We report results considering a geometrical model that encompass the beam candidate region by calculating the MVEE applied to preliminary selection of particles, which was mostly consistent with the shape of the bunch. The geometry assumption may

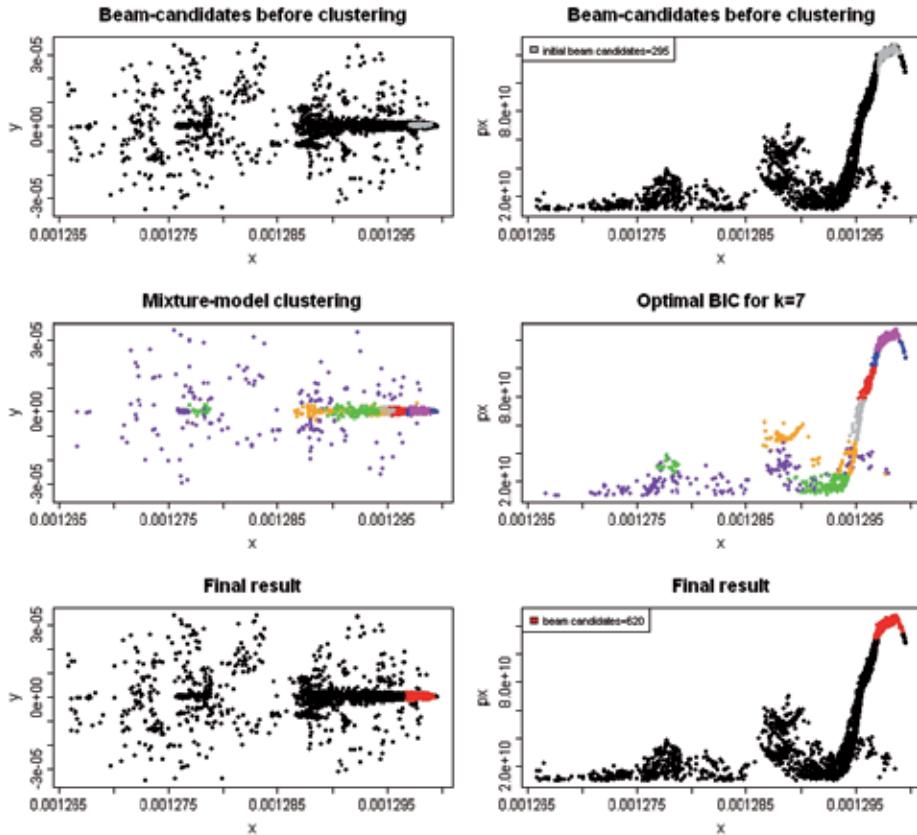


Fig. 9. Result of beam detection for $t_i = 27$ from dataset A: beam candidates in gray from processing in block B3 (top), clustering using mixture models, with colors representing the different partitions over a sampled subset (center) and final result of electron bunch detection, with increased number of particles after generalization with EM-algorithm, from block B4 (bottom).

result in inclusion of outliers if the beam present different shapes; however, we eliminate outliers during the moving averages procedure, keeping particles more likely to be part of the electron bunch.

We calculate model-based clusters for each time step, after retrieving the results from block B1 and B3. We illustrate the partitions of one time step of all datasets in Figure 9, 11 and 12, showing the phase space of a time step where the beam was expected to be compact. In Figure 9, the two top plots show the result of beam candidate selection, in gray, for dataset A as output by block B3. The two center plots present different compact groups of particles given by the mixture model, and the bottom plots give the final result of electron bunch selection (B4), emphasized in red color. The result of B3 indicates potential clusters of particles, important to guide the sampling and identify the cluster position, but the definition of particle partitions that are connected and compact given x, y, px is only accomplished after B4, which finds

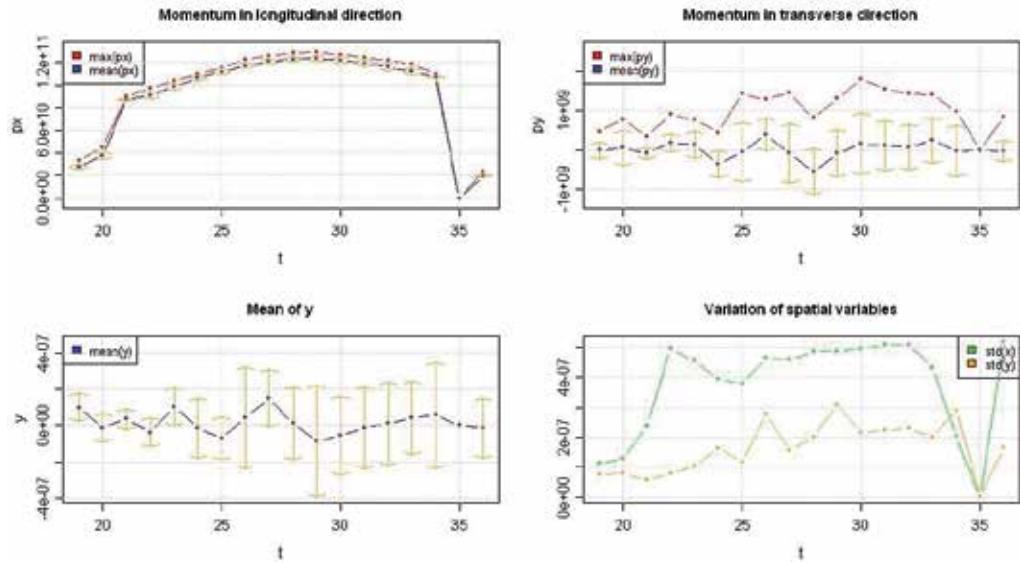
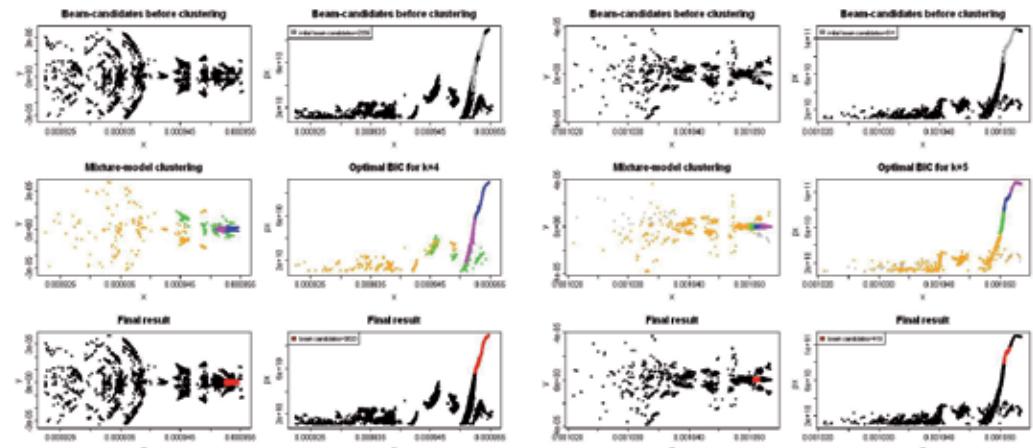


Fig. 10. Beam quality assessment to evaluate the dispersion of particle parameters using the time series in dataset A: the curves show the history of one bunch that forms around $t = 22$, reaching maximum energy around $t = 27$.

the k -component varying-volume ellipsoidal mixture model clustering that best represent the particles, using BIC as criterion function.



(a) Dataset B, $t=21$

(b) Dataset C, $t=22$

Fig. 11. Result of beam detection for $t_i = 21$ from dataset B and $t_i = 22$ from dataset C: beam candidates in gray from processing in block B3 (top), clustering using mixture models, with colors representing the different partitions over a sampled subset (center) and final result of electron bunch detection, after generalization with EM-algorithm, from block B4 (bottom).

Next, we evaluate the compactness of the electron bunch (B5) by calculating moving averages (mv_j) over the time series. Figures 10, 13 and 14 show the result of calculating statistics from mv_j , using the particles selected according to block B4. While the beam detection at each time step may contain outliers, the intersection with adjacent time steps returns the core subset of particles (id_j) that persists at least for three time steps. At the top left of Figure 10, we show the red and blue curves, with the maximum and mean value of px (red and blue, respectively), for each time step. The distance between the red curve and the blue curve, at each time step, is an indicator of the dispersion of the particles in the bunch as well as the length of the yellow arrows (standard deviation of the mv_j with respect to x , y , px or py). Also, notice that the moving averages capture the local behavior of a particle bunch that persists for at least three time steps, but do not guarantee that the bunch is present throughout the simulation. There are time steps where the algorithm does not capture any beam, which correspond to moving average equal to zero as in $t = [28, 34]$ from dataset D in Figure 14.a. The period of non-bunch detection, $mv_j = 0$, corresponds to the presence of peaks on f at different, non-adjacent positions, which is correlated to the dispersion of the particles for that period. It follows similar interpretation of the particle dispersion in terms of spatial parameters (x and y) and energy (px and py) to other datasets. Figures 10, 13 and 14 demonstrate that the algorithm automatically identifies the bunch over a range of simulation conditions and resulting bunch qualities.

Our tests were conducted on an SGI Altix with 32 1.4 GHz Itanium-2 Processors and 180 GBytes of shared memory. The primary motivation for using this computing system is the large memory; the current implementation of the mixture model clustering algorithms in package *mclust* is fairly memory-intensive and does not work on standard workstations for large datasets. The SGI Altix is a multi-user machine, thus computing times in different stages of the framework are approximate. Our process of computing beam candidate regions (block B3) is reasonably fast and could be easily incorporated into routine inspection as a preprocessing step. The clustering computation is more expensive, and new implementations are necessary to improve performance. The approximate computing times of beam candidate (in seconds) and clustering (in minutes) for each dataset are organized as pairs with time in parenthesis: A=(15.6s, 31min), B=(66s, 20min), C=(24.3s, 42min), D=(295.8s, 116min) and E=(417.4s, 975min).

4. Conclusions and Future Work

Previous investigations from Ushizima et al. (2008) and Rübel et al. (2008) to find particle bunches reported results using fixed spatial tolerance around centers of maximum compactness and assumed *ad hoc* thresholding values to determine potential particle candidates involved in the physical phenomena of interest. Ushizima et al. (2008) pointed out limitations inherent to techniques that detects maximum values using only one-dimensional spatial approach (x -axis), which did not capture the most condensed structure when confined to depressions between peaks in px or when dispersed in y .

The current approach circumvented most of these problems, since the algorithm searched for compact high density group of particles using both spatial information, x and y , and momentum in the direction of laser propagation, px . We improved the detection of a high density volume of particles by using the 3D kernel density, followed by the detection of its maximum and enclosing the particle subsets using MVEE, thus generating subsets of particles which are beam candidate regions. These subsets provided the position of the most likely cluster to contain a compact electron bunch in a time step. We proposed the use of moving averages to

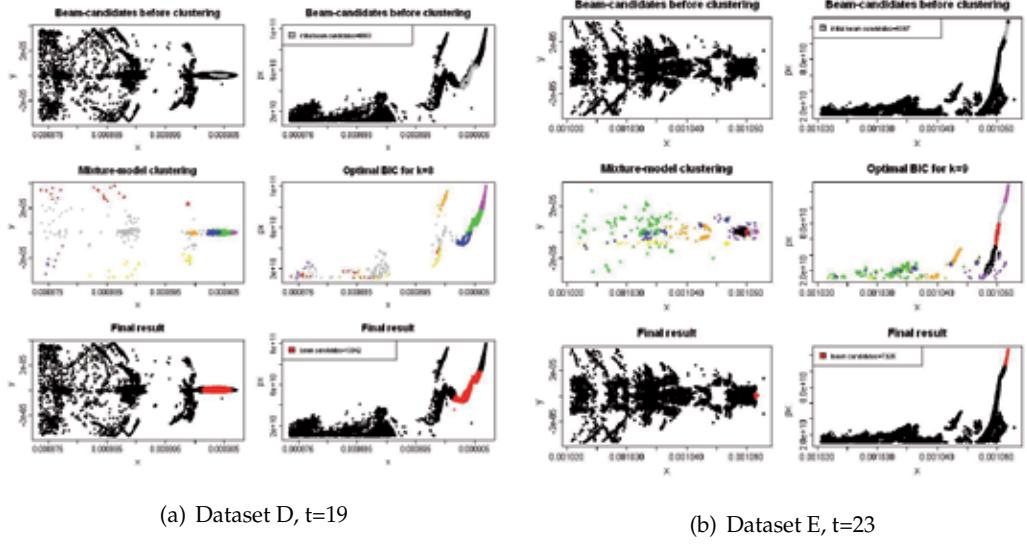


Fig. 12. Result of beam detection for $t_i = 19$ from dataset D and $t_i = 23$ from dataset E: beam candidates in gray from processing in block B3 (top), clustering using mixture models, with colors representing the different partitions over a sampled subset (center) and final result of electron bunch detection, after generalization with EM-algorithm, from block B4 (bottom).

identify periods of bunch stability, in the time series, and we derived dispersion measures to characterize beam compactness and quality.

Our implementation of function calls to the HDF-FastQuery interface allowed us to load data using FastBit in R, saving time while only loading subsets of particles that potentially participate to the phenomenon of interest. Our results showed that we can assess the beam evolution using both mathematical models and machine learning techniques to automate the search for the beam using large LWFA simulation datasets. Application of hierarchical approaches as in the R packages *hclust* and *mclust* are prohibitive if not combined with sampling methods. We present an algorithm to sample the simulation data, but Monte Carlo methods [Banfield & Raftery (1993)] could be used by adding a repetitive randomness process as a way of guaranteeing representation of a beam candidate region and improvement of accuracy. Future evaluations may consider more sophisticated methods such as Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) as in Zhang et al. (1996) and hierarchical clustering based on granularity as in Liang & Li (2007), which are designed for very large data sets. Further investigation should also include subspace clustering as in Kriegel et al. (2009) once the large simulation datasets contain target regions that can be determined using the techniques proposed in our framework.

5. Acknowledgments

This work was supported by the Director, Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 through the Scientific Discovery through Advanced Computing (SciDAC) program's Visualization and Analytics Center for Enabling Technologies (VACET) and by the U.S. DOE Office

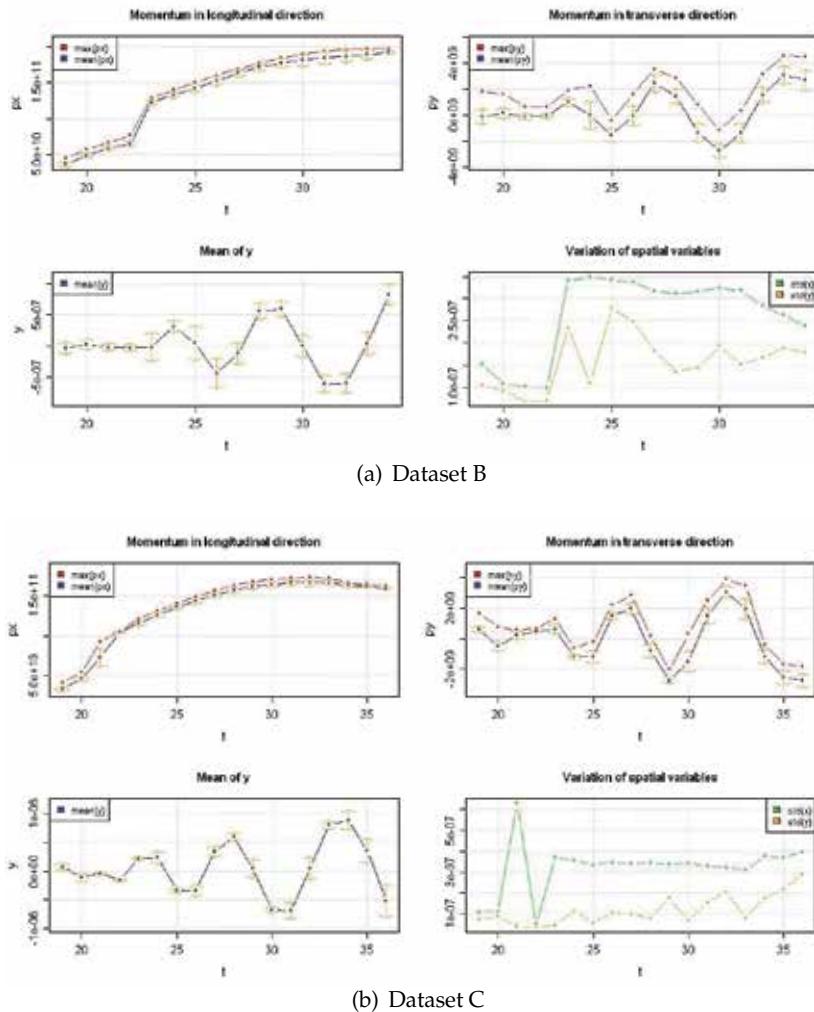


Fig. 13. Beam quality assessment to evaluate the dispersion of particle parameters using the time series in: (a) dataset B: the curves show the history of one bunch that forms around $t = 23$, reaching maximum energy around $t = 33$; (b) dataset C: the curves show the history of one bunch that forms around $t = 21$, reaching maximum energy around $t = 33$.

of Science, Office of High Energy Physics, grant DE-FC02-07ER41499, through the COMPASS SciDAC project and by the U.S. DOE Office of Energy Research by the Applied Mathematical Science subprogram, under Contract Number DE-AC03-76SF00098. This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. We also thank the VORPAL development team for ongoing efforts in development and maintenance on a variety of supercomputing platforms, including those at NERSC NERSC (2009).

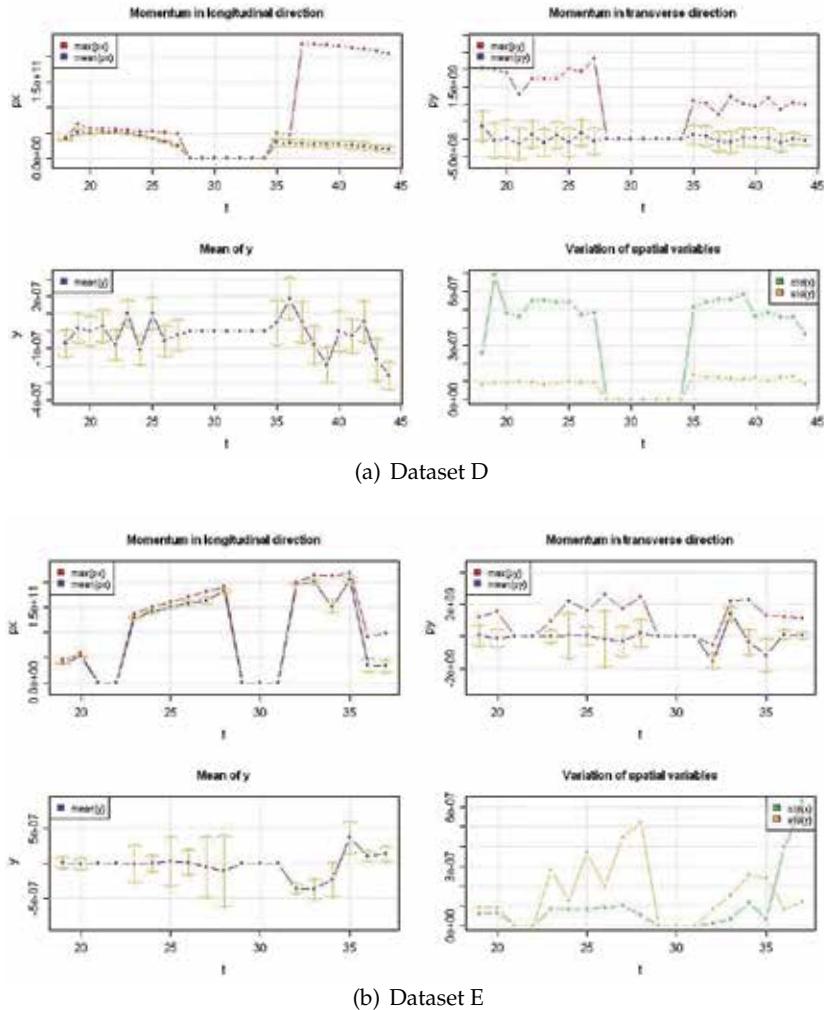


Fig. 14. Beam quality assessment to evaluate the dispersion of particle parameters using the time series in: (a) dataset D: the curves show the history of two bunches: one that forms at the beginning of the simulation, compact and lower energy ($t = [18, 27]$) and a second one with broader dispersion in px and higher energy ($t = [36, 44]$). The beam is not detected by the algorithm from $t = [28, 34]$, represented by zero values in the four graphs; (b) dataset E: the curves show the history of two bunches that form around $t = 23$, reaching maximum energy and compactness around $t = 34$. The beam is not detected by the algorithms from $t = [21, 22]$ and $t = [29, 31]$, represented by zero values.

6. References

Adelmann, A., Gsell, A., Oswald, B., Schietinger, T., Bethel, E. W., Shalf, J., Siegerist, C. & Stockinger, K. (2007). Progress on H5Part: A Portable High Performance Parallel

- Data Interface for Electromagnetic Simulations, *Particle Accelerator Conference PAC07 25–29 June*. <http://vis.lbl.gov/Publications/2007/LBNL-63042.pdf>.
- Adelmann, A., Ryne, R., Shalf, J., & Siegerist, C. (2005). H5part: A portable high performance parallel data interface for particle simulations, *Particle Accelerator Conference PAC05 May 16-20*.
- Ahipasaoglu, S. D., Sun, P. & Todd, M. J. (2008). Linear convergence of a modified frank-wolfe algorithm for computing minimum-volume enclosing ellipsoids, *Optimization Methods Software* **23**(1): 5–19.
- Bagherjeiran, A. & Kamath, C. (2006). Graph-based methods for orbit classification, *SDM*.
- Banfield, J. D. & Raftery, A. E. (1993). Model-based Gaussian and non-Gaussian clustering, *Biometrics* **49**: 803–821.
- Birdsall, C. K., Langdon, A. B., Vehedi, V. & Verboncoeur, J. P. (1991). *Plasma Physics via Computer Simulations*, Adam Hilger, Bristol, Eng.
- FastBit (2009). Fastbit: An efficient compressed bitmap index technology, <https://codeforge.lbl.gov/projects/fastbit/>.
- Feng, D. & Tierney, L. (2009). Miscellaneous 3d plots, <http://cran.r-project.org/web/packages/misc3d/misc3d.pdf>.
- Fraley, C. & Raftery, A. (2006). Mclust version 3 for r: Normal mixture modeling and model-based clustering, Technical Report no. 504.
- Fraley, C. & Raftery, A. (2009). Model-based clustering / normal mixture modeling: the mclust package, <http://www.stat.washington.edu/fraley/mclust>.
- Fraley, C. & Raftery, A. E. (2002). Model-based clustering, discriminant analysis, and density estimation, *Journal of the American Statistical Association* **97**: 611–631.
- Fraley, C. & Raftery, A. E. (2007). Model-based methods of classification: using the mclust software in chemometrics, *Journal of Statistical Software* **18**(6): 1–13.
- Geddes, C. G. R. (2005). *Plasma Channel Guided Laser Wakefield Accelerator*, PhD thesis, University of California, Berkeley.
- Geddes, C. G. R., Bruhwiler, D. L., Cary, J. R., Mori, W. B., J.L. Vay, S. F. M., Katsouleas, T., Cormier-Michel, E., Fawley, W. M., Huang, C., Wang, X., Cowan, B., Decyk, V. K., Esarey, E., Fonseca, R. A., Lu, W., Messmer, P., Mullowney, P., Nakamura, K., Paul, K., Plateau, G. R., Schroeder, C. B., Silva, L. O., Toth., C., Tsung, F. S., Tzoufras, M., Antonson, T., Vieira, J. & Leemans, W. P. (2008). Computational studies and optimization of wakefield accelerators, *J. Phys.: Conf. Ser.* **125**: 1–11.
- Geddes, C. G. R., Cormier-Michel, E., Esarey, E. H., Schroeder, C. B., Vay, J.-L., Leemans, W. P., Bruhwiler, D. L., Cary, J. R., Cowan, B., Durant, M., Hamill, P., Messmer, P., Mullowney, P., Nieter, C., Paul, K., Shasharina, S., Veitzer, S., Weber, G., Rübel, O., Ushizima, D., Prabhat, W. Bethel, E. & Wu, K. (2009). Large Fields for Smaller Facility Sources, *SciDAC Review* **13**.
- Geddes, C. G. R., Toth, C., van Tilborg, J., Esarey, E., Schroeder, C., Bruhwiler, D., Nieter, C., Cary, J. & Leemans, W. (2004). High-Quality Electron Beams from a Laser Wakefield Accelerator Using Plasma-Channel Guiding, *Nature* **438**: 538–541. LBNL-55732.
- Gentleman, R. & Ihaka, R. (2009). The R project for statistical computing, <http://www.r-project.org>.
- Gosink, L., Shalf, J., Stockinger, K., Wu, K. & Bethel, E. W. (2006). HDF5-FastQuery: Accelerating Complex Queries on HDF Datasets using Fast Bitmap Indices, *Proceedings of the 18th International Conference on Scientific and Statistical Database Management*, IEEE Computer Society Press. LBNL-59602.

- Greene, D., Cunningham, P. & Mayer, R. (2008). Unsupervised learning and clustering, *Machine learning techniques for multimedia* pp. 51–90.
- H5Part (2009). H5Part: a portable high performance parallel data interface to hdf5, <https://codeforge.lbl.gov/projects/h5part/>.
- Haughton, D., Legrand, P. & Woolford, S. (2009). Review of three latent class cluster analysis packages: Latent gold, polca and mclust, *The American Statistician* **63**(1): 81–91.
- HDF5-FastQuery (2009). Hdf5-fastquery: Accelerating complex queries on hdf datasets using fast bitmap indices, <http://www-vis.lbl.gov/Events/SC05/HDF5FastQuery/index.html>.
- Kamath, C. (2009). *Scientific Data Mining: A Practical Perspective*, Society for Industrial and Applied Mathematic (SIAM), Philadelphia, USA.
- Khachiyan, L. & Todd, M. (1993). On the complexity of approximating the maximal inscribed ellipsoid for a polytope, *Math. Program.* **61**(2): 137–159.
- Kriegel, H., Kröger, P. & Zimek, A. (2009). Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering, *ACM Trans. Knowl. Discov. Data* **3**(1): 1–58.
- Kumar, P. & Yildirim, E. A. (2005). Minimum-volume enclosing ellipsoids and core, *Journal of Optimization Theory and Applications* **126**: 1–21.
- Liang, J. & Li, G. (2007). Hierarchical clustering algorithm based on granularity, GrC, IEEE, pp. 429–432.
- Love, N. S. & Kamath, C. (2007). Image analysis for the identification of coherent structures in plasma, *Applications of Digital Image Processing*. Edited by Tescher, Andrew G.. *Proceedings of the SPIE*, Vol. 6696.
- Messmer, P. & Bruhwiler, D. L. (2006). Simulating laser pulse propagation and low-frequency wave emission in capillary plasma channel systems with a ponderomotive guiding center model, *Phys. Rev. ST Accel. Beams* **9**(3): 031302.
- Moshtagh, N. (2009). Minimum volume enclosing ellipsoid, <http://www.mathworks.com/matlabcentral/fileexchange/9542>.
- NERSC (2009). National energy research scientific computing center, <http://www.nersc.gov/>.
- Nieter, C. & Cary, J. R. (2004). Vorpal: a versatile plasma simulation code, *J. Comput. Phys.* **196**(2): 448–473.
- Pukhov, A. & ter Vehn, J. M. (2002). Three-dimensional particle-in-cell simulations of laser wakefield experiments, *Applied Physics B-Lasers and Optics* **74**(4-5): 355–361.
- Rübel, O., Geddes, C. G., Cormier-Michel, E., Wu, K., Prabhat, Weber, G. H., Ushizima, D. M., Messmer, P., Hagen, H., Hamann, B. & Bethel, W. (2009). Automatic beam path analysis of laserwakefield particle acceleration data. in submission.
- Rübel, O., Prabhat, Wu, K., Childs, H., Meredith, J., Geddes, C. G. R., Cormier-Michel, E., Ahern, S., weber, G. H., Messmer, P., Hagen, H., Hamann, B. & Bethel, E. W. (2008). High performance multivariate visual data exploration for extremely large data, *SuperComputing 2008 (SC08)*, Austin, Texas, USA.
- Samperi, D. (2006). RcppTemplate, <http://cran2.arsmachinandi.it/doc/packages/RcppTemplate.pdf>.
- Shumway, R. H. & Stoffer, D. S. (2006). *Time Series Analysis and Its Applications (Springer Texts in Statistics)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA.
- Tajima, T. & Dawson, J. M. (1979). Laser electron accelerator, *Physical Review Letters* **43**(4): 267–270.

- Tsung, F., AntonSEN, T., Bruhwiler, D., Cary, J., Decyk, V., Esarey, E., Geddes, C., Huang, C., Hakim, A., Katsouleas, T., Lu, W., Messmer, P., Mori, W., Tzoufras, M. & Vieira, J. (2007). Three-dimensional particle-in-cell simulations of laser wakefield experiments, *J. Phys.: Conf. Ser.* **78**(1): 012077+.
URL: <http://dx.doi.org/10.1088/1742-6596/78/1/012077>
- Tsung, F. S., Narang, R., Mori, W. B., Joshi, C., Fonseca, R. A. & Silva, L. O. (2004). Near-gev-energy laser-wakefield acceleration of self-injected electrons in a centimeter-scale plasma channel, *Phys. Rev. Lett.* **93**(18): 185002.
- Ushizima, D., Rübel, O., Prabhat, Weber, G., Bethel, E. W., Aragon, C., Geddes, C., Cormier-Michel, E., Hamann, B., Messmer, P. & Hagen, H. (2008). Automated Analysis for Detecting Beams in Laser Wakefield Simulations, *2008 Seventh International Conference on Machine Learning and Applications, Proceedings of IEEE ICMLA'08*. LBNL-960E.
- Vermunt, J. & Magidson, J. (2002). Latent class cluster analysis, *Applied latent class analysis* pp. 89–106.
- VisIt (2009). Visit - free interactive parallel visualization and graphical analysis tool, <https://wci.llnl.gov/codes/visit/>.
- Wand, M. P. & Jones, M. C. (1995). *Kernel smoothing*, Chapman and Hall/CRC.
- Weissbach, R. & Gefeller, O. (2009). A rule-of-thumb for the variable bandwidth selection in kernel hazard rate estimation.
- Wu, K., Otoo, E. & Shoshani, A. (2004). On the performance of bitmap indices for high cardinality attributes, *VLDB*, pp. 24–35.
- Wu, K., Otoo, E. & Shoshani, A. (2006). Optimizing bitmap indices with efficient compression, *ACM Transactions on Database Systems* **31**: 1–38.
- Yip, K. M. (1991). *KAM: A System for Intelligently Guided Numerical by Computer*, MIT Press.
- Zhang, T., Ramakrishnan, R. & Livny, M. (1996). Birch: an efficient data clustering method for very large databases, *SIGMOD '96: Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, ACM, New York, NY, USA, pp. 103–114.
URL: <http://dx.doi.org/10.1145/235968.233324>

Specificity Enhancement in microRNA Target Prediction through Knowledge Discovery

Yanju Zhang¹, Jeroen S. de Bruin² and Fons J. Verbeek¹

¹ Imaging and Bioinformatics group

² Algorithms group

Leiden University, Leiden

The Netherlands

1. Introduction

In this chapter we explore and investigate a range of methods in pursue of improving target prediction of microRNA. The currently available prediction methods produce a large output set that also includes a rather high amount of false positives. Additional strategies for target prediction are necessary and we elaborate on one particular group of microRNAs; i.e. those that might bind to the same target. We intend to transfer our approach to other groups of microRNAs as well as the broader application to the important model species.

microRNAs (miRNAs) are a novel class of post-transcriptional gene expression regulators discovered in the genome of plants, animals and viruses. The mature miRNAs are about 22 nucleotides long. They bind to their target messengerRNA (mRNA) and therefore induce translational repression or degradation of target mRNAs (Enright et al., 2003; Bartel, 2004). Recent studies have elucidated that these small molecules are highly conserved between species indicating their fundamental roles conserved in evolutionary selection. They are implicated in developmental timing regulation (Reinhart et al., 2000), apoptosis (Brennecke et al., 2003) and cell proliferation (Lecellier et al., 2005). Some of them have been described to act as potential tumor suppressors (Johnson et al., 2005), potential oncogenes (He et al., 2005) and might be important targets for drugs (Maziere & Enright, 2007).

The identification of large number of miRNAs existing in different species has increased the interest in unraveling the mechanism of this regulator. It has been proven that more than one miRNA regulates one target and vice versa (Enright et al., 2003). Therefore understanding this novel network of regulatory control is highly dependent on identification of miRNA targets. Due to the costly, labor-intensive nature of experimental techniques required, currently, there is no large-scale experimental target validation available leaving the biological function of the majority completely unknown (Enright & Griffiths-Jones, 2007). These limitations of the wet experiments have lead to the development of computational prediction methods.

It has been established that the physical RNA interaction requires sequence complementarity and thermodynamic stability. Unlike plant miRNAs, which bind to their

targets through near-perfect sequence complementarity, the interaction between animal miRNAs and their targets is more flexible. Partial complementarity is frequently found (Enright et al., 2003) and this flexibility complicates computation. Lots of effort has been put into characterizing functional miRNA-target pairing. The most frequently used prediction algorithms are miRanda, TargetScan/TargetScanS, RNAhybrid, DIANA-microT, picTar, and miTarget.

MiRanda (Enright et al., 2003) is one of the earliest developed large-scale target prediction algorithm which was first designed for *Drosophila* then adapted for human and other vertebrates. It consists of three steps: First, a dynamic programming local alignment is carried out between miRNAs and 3'UTR of potential targets using a scoring matrix. After filtering by threshold score, the resulting binding sites are evaluated thermodynamically using the Vienna RNA fold package (Wuchty, 1999). Finally, the miRNA pairs that are conserved across species are kept.

TargetScan/TargetScanS (Lewis et al., 2003; Lewis et al., 2005) have a stronger emphasize on the seed region. In the standard version of TargetScan, the predicted target-sites first require a 7-nucleotide (nt) match to the seed region of miRNA, i.e., nucleotides 2-8; second, conservation in 4 genomes (human, mouse, rat and puffer fish), and third, thermodynamic stability. TargetScanS is the new and simplified version of TargetScan. It extends the cross-species comparison to 5 genomes (human, mouse, rat, dog and chicken) and requires a seed match of only 6-nt long (nucleotides 2-7). Through the requirement of more stringent species conservation it leads to more accurate predictions even without conducting free energy calculations.

RNAhybrid (Rehmsmeier et al., 2004) was the first method which integrated powerful statistical models for large-scale target prediction. Basically, this method finds the energetically most favorable hybridization sites of a small RNA in a large RNA string. It takes candidate target sequences and a set of miRNAs and looks for energetically favorable binding sites. Statistical significance is evaluated with an extreme value statistics of length normalized minimum free energies for individual hits, a Poisson approximation of multiple hits, and the calculation of effective numbers of orthologous targets in comparative studies of multiple organisms. Results are filtered according to *p-value* thresholds.

DIANA-microT identified putative miRNA-target interaction using a modified dynamic programming algorithm with a sliding window of 38 nucleotides that calculated binding energies between two imperfectly paired RNAs. After filtering by an energy threshold, the candidates are examined by the rules derived from mutation experiments of a single let-7 binding site. Finally, those which were conserved between human and mouse were further considered for experimental verification (Grun & Rajewsky, 2007; Sethupathy et al., 2007).

PicTar takes sets of co-expressed miRNAs and searches for combinations of miRNA binding sites in each 3'UTR (Krek et al., 2005). And miTarget is a support vector machine classifier for miRNA target-gene prediction, which utilizes a radial basis function kernel to characterize targets by structural, thermodynamic, and position-based features (Kim et al., 2006).

Among the algorithms discussed previously, miRanda and TargetScan/TargetScanS belong to the sequence-based algorithms which evaluate miRNA-target complementarity first, then calculate the binding site thermodynamics to further prioritize; in contrast, DIANA-microT and RNAhybrid are based on algorithms that are rooted in thermodynamics, thus using thermodynamics as the initial indicator of potential miRNA binding site.

Until now, it remains unclear whether sequence or structure is the better predictor of a miRNA binding site (Maziere & Enright, 2007). All of the above mentioned methods produce a large set of predictions and include a relatively high false positive ratio; all in all this indicates that these methods are promising methods but still far away from perfect. The estimated false-positive rate (FPR) for PicTar, miRanda and TargetScan is about 30%, 24-39% and 22-31% respectively (Bentwich, 2005; Sethupathy et al., 2006b; Lewis et al., 2003). It has been reported that miTarget has a similar performance as TargetScan (Kim et al., 2006). In addition to the relatively high FPR, Enright *et al.* observed that many real targets are not predicted by these methods and this seems to be largely due to requirements for evolutionary conservation of the putative miRNA target-site across different species (Enright et al., 2003; Martin, 2007). In general we also notice that in all of these algorithms, the target prediction is based on features that consider the miRNA-target interaction such as sequence complementarity and stability of miRNA-target duplex.

Through the observations in the population of confirmed miRNAs targets we became aware that some miRNAs are validated as binding the same target. For example, in human miR-17 and miR-20a both regulate the expression of E2F transcription factor 1 (E2F1); while miR-221 and miR-222 both bind to v-kit Hardy-Zuckerman 4 feline sarcoma viral oncogene homolog (KIT). Subsequently, we considered that this observation would allow target identification from the analysis of functionally similar miRNAs.

Based on this idea, we present an approach which analyzes miRNA-miRNA relationships and utilizes them for target prediction. Our aim is to improve target prediction by using different features and discovering significant feature patterns through tuning and combining several machine learning techniques. To this respect, we applied feature selection, principle component analysis, classification, decision trees, and propositionalization-based relational subgroup discovery to reveal the feature patterns between known miRNAs. During this procedure, different data setups were evaluated and the parameters were optimized. Furthermore, the derived rules were applied to functionally unknown miRNAs so as to see if new targets could be predicted. In the analysis of functionally similar miRNAs, we found that genomic distance, seed and overall sequence similarities between miRNAs are dominant features in the description of a group of miRNAs binding the same target. Application of one specific rule resulted in the prediction of targets for five functionally unknown miRNAs which were also detected by some of the existing methods. Our method is complementary to the existing prediction approaches. It contributes to the improvement of target identification by predicting targets with high specificity and without conservation limitation. Moreover, we discovered that knowledge discovery especially the propositionalization-based relational subgroup discovery, is suitable for this application domain since it can interpret patterns of similar function miRNAs with respect to the limited features available.

The remainder of this chapter is organized as follows. In Section 2, miRNA biology and databasing as well as the background of the machine learning techniques which are the components of our method are explained: i.e., miRNA biogenesis and function, related databases, feature selection, principle component analysis, classification, decision trees and propositionalization-based relational subgroup discovery. Section 3 specifies the proposed method including data preparation, algorithm configuration and parameter optimization. The results are summarized in Section 4. Finally, In Section 5, we discuss the strengths and

the weaknesses of the applied machine learning techniques and feasibility of the derived miRNA target prediction rules.

2. Background

The first two subsections are devoted to the exploration of miRNA biology whereas the latter two subsections have a computational nature.

2.1 microRNA biogenesis and function

The mature miRNAs are ~22 nucleotide single-stranded noncoding RNA molecules. They are derived from miRNA genes. First, miRNA gene is transcribed to primary miRNA transcripts (pri-microRNA), which is between a few hundred or a few thousand base pair long. Subsequently, this pri-microRNA is processed into hairpin precursors (pre-microRNA), which has a length of approximately 70 nucleotides, by the protein complex consisting of the nuclease Drosha and the double-stranded RNA binding protein Pasha. The pre-miRNA then is transported to cytoplasm and cut into small RNA duplexes of approximately 22 nucleotides by the endonuclease Dicer. Finally, either the sense strand or antisense strand can function as templates giving rise to mature miRNA. Upon binding to the active RISC complex, mature miRNAs interact with the target mRNA molecules through base pair complementarity, therefore inhibit translation or sometimes induce mRNA degradation (Chen, 2005). Fig. 1 illustrates the process of biogenesis and function of miRNAs. For reasons of simplification the auxiliary protein complexes are not included in the picture.

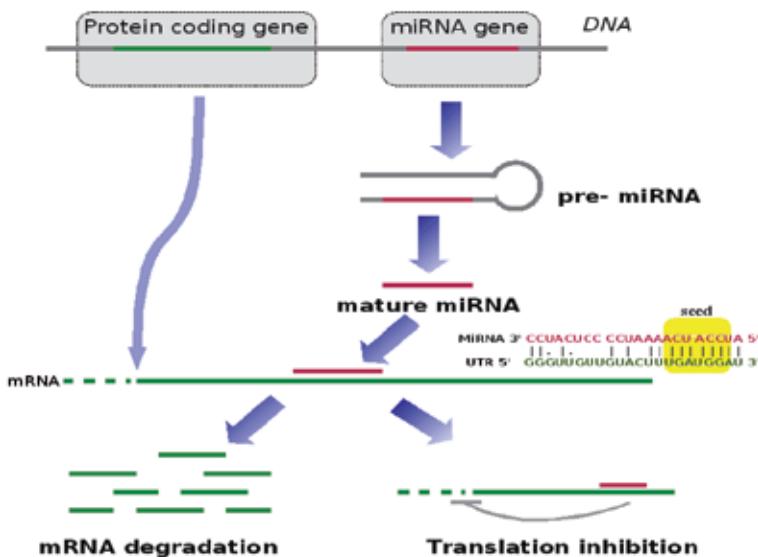


Fig. 1. Simplified illustration of miRNA biogenesis and function. miRNA genes are first transcribed to pre-miRNA, and then proceeded to mature miRNAs. Upon binding to these miRNAs through sequence complementarity, the messengerRNAs (mRNAs), which are called the targets of miRNAs, will be either degraded or translation of the targets will be inhibited.

It is suggested that miRNAs tend to bind 3' UTR (3' Untranslated Region) of their target mRNAs (Lee et al., 1993). Further studies have discovered that position 2-8 of miRNAs, which is called 'seed' region, has been described as a key specificity determinant of binding, requires good or perfect complementarity (Lewis et al., 2003; Lewis et al., 2005). In Fig. 1, a detailed miRNA-target interaction is showed with a highlighted seed region.

2.2 miRNA databases

miRBase: MiRBase is the primary online repository for published miRNA sequence data, annotation and predicted gene targets (Griffiths-Jones et al., 2006; Griffiths-Jones, 2004). It consists of three parts:

1. The miRBase Registry acts as an independent authority of miRNA gene nomenclature, assigning names prior to publication of novel miRNA sequences.
2. The miRBase Sequences is a searchable database for miRNA sequence data and annotation. The latest version (Release 13.0, March 2009) contains 9539 entries representing hairpin precursor miRNAs, expressing 9169 mature miRNA products, in 103 species including primates, rodents, birds, fish, worms, flies, plants and viruses.
3. The miRBase Targets is a comprehensive database of predicted miRNA target genes. The core prediction algorithm currently is miRanda (version 5.0, Nov 2007). It searches over 2500 animal miRNAs against over 400 000 3'UTRs from 17 species for potential target sites. In human, the current version predicts 34788 targets for 851 human miRNAs.

Tarbase: Tarbase is a comprehensive repository of a manually curated collection of experimentally supported animal miRNA targets (Sethupathy et al., 2006a; Papadopoulos et al., 2008). It describes each supported target site by the miRNA which binds it, the target genes, the direct and indirect experiments that were conducted to validate it, binding site complementarity and etc. The latest version (Tarbase 5.0, Jun 2008) records more than 1300 experimentally supported miRNA target interactions for human, mouse, rat, zebrafish, fruitfly, worm, plant, and virus. As machine learning methods become more popular, this database provides a valuable resource to train and test for machine learning based target prediction algorithms.

2.3 Pattern recognition

Pattern recognition is considered a sub-topic of machine learning. It concerns with classification of data either based on *a priori* knowledge or based on statistical information extracted from the patterns. The patterns to be classified are usually groups of measurements, features or observations, which define data points in an appropriate multidimensional space. Our pattern recognition proceeds in three different stages: feature reduction, classification and cross-validation.

Feature reduction: Feature reduction includes feature selection and extraction. Feature selection is the technique of selecting a subset of relevant features for building learning models. In contrast, feature extraction seeks a linear or nonlinear transformation of original variables to a smaller set. The reason why not all features are used is because of performance issues, but also to make results easier to understand and more general. Sequential backward selection is a feature selection algorithm. It starts with entire set, and

then keeps removing one feature at a time so that the entire subset so far performs the best. Principle component analysis (PCA) is an unsupervised linear feature extraction algorithm. It derives new variables in decreasing order of importance that are a linear combinations of the original variables, uncorrelated and retain as much variation as possible (Webb, 2002).

Classification: Classification is the process of assigning labels on data records based on their features. Typically, the process starts with a training dataset that has examples already classified. These records are presented to the classifier, which trains itself to predict the right outcome based on that set. After that, a testing set of unclassified data is presented to the classifier, which classifies all the entries based on its training. Finally, the classification is being inspected. The better the classifier, the more good classifications it has made. Linear discriminant classifier (LDC) and quadratic discriminant classifiers (QDC) are two frequently used classifiers which separate measurements of two or more classes of objects or events by a linear or a quadric surface respectively.

Cross-validation: Cross-validation is the process of repeatedly partitioning a dataset in a training set and a testing set. When the dataset is partitioned in n parts we call that n -fold cross-validation. After partitioning the set in n parts, the classifier is trained with $n-1$ parts, and tested on the remaining part. This process is repeated n times, each time a different part functions as the training part. The n results from the folds then can be averaged to produce a single estimation of error.

2.4 Knowledge discovery

Knowledge discovery is the process which searches large volumes of data for patterns in order to find understandable knowledge about the data. In our knowledge discovery strategy, decision tree and relational subgroup discovery are applied.

Decision tree: The decision tree (Witten & Frank, 1999) is a common machine learning algorithm used for classification and prediction. It represents rules in the form of a tree structure consisting of leaf nodes, decision nodes and edges. This algorithm starts with finding the attribute with the highest information gain which best separates the classes, and then it is split into different groups. Ideally, this process will be repeated until all the leaves are pure.

Relational subgroup discovery: Subgroup discovery belongs to descriptive induction (Zelezny & Lavrac, 2006) which discover patterns described in the form of individual rules. Relational subgroup discovery (RSD) is the algorithm which utilizes relational datasets as input, generates subgroups whose class-distributions differ substantially from the complete dataset with respect to the property of interest (Lavrac et al., 2003). The principle of RSD can be simplified as follows; first, a feature is constructed through first-order feature construction and the features covering empty datasets are retracted. Second, rules are induced using weighted relative accuracy heuristics and weighted covering algorithm. Finally, the induced rules are evaluated by employing the combined probabilistic classifications of all subgroups and the area under the receiver operating characteristics (ROC) curve (Fawcett, 2006). The key improvement of RSD is the application of weighted relative accuracy heuristics and weighted covering algorithm, i.e.

$$WRAcc(H \leftarrow B) = p(B) \cdot (p(H|B) - p(H)) \quad (1)$$

The weighted relative accuracy heuristics is defined as equation 1. In rule $H \leftarrow B$, H stands for Head representing classes, while B denotes the Body which consists of one or a conjunction of first-ordered features. p is the probability function. As shown in the equation, weighted relative accuracy consists of two components: weight $p(B)$, and relative accuracy $p(H | B) - p(H)$. The second term, relative accuracy, is the relative accuracy gain between the conditional probability of class H given that features B is satisfied and the probability of class H . A rule is only interesting if it improves over this default rule $H \leftarrow \text{true}$ accuracy (Zelezny & Lavrac, 2006).

In the weighted covering algorithm, the covered positive examples are not deleted from the current training set which is the case for the classical covering algorithm. Instead, in each run of the covering loop, the examples are given decreasing weights while the number of iterations is increasing. In doing so, it is possible to discover more substantial significant subgroups and thereby achieving to find interesting subgroup properties of the entire population.

3. Experimental setups, methods and materials

3.1 Data collection

In the interest of including maximally useful data, human miRNAs are chosen as the research focus. The latest version of TarBase (TarBase-V5 released at 06/2008) includes 1093 experimentally confirmed human miRNA-target interactions. Among them, 243 are supposed by direct experiment such as in vitro reporter gene (Luciferase) assay, while the rest are validated by an indirect experimental support such as microarrays. Considering the fact that the indirect experiments could induce the candidates which are in the downstream of the miRNA involved pathways, it is uncertain whether these can virtually interact with miRNA or not. Thus they are excluded and only the miRNAs-target interactions with direct experiment support are used in this study.

We observed that some miRNAs are validated as binding the same target. According to this observation, we pair the miRNAs as positive if they bind the same target, and randomly couple the rest as the negative data set. In total, there are 93 positive pairs. After checking the consistency of the name of miRNAs and removing the redundant data (for example, miR-26 and miR-26-1 refer to the same miRNA), 73 pairs are kept and thus another 73 negative pairs are generated. For quality control reasons, the data generation step is repeated 10 times and each set is tested individually in the following analysis.

Here we clarify two notions; known miRNAs are those whose function is known and have been validated for having at least one target, unknown miRNAs refer to those for which the targets are unknown.

3.2 Feature collection

In the study of miRNA-target interaction, it has been established that this physical binding requires sequence complementarity and thermodynamic stability. Here some of miRNA-target interaction features are transformed to the study of functionally similar miRNA pairs. We predefine four features: overall sequence (~22 nt) similarity, seed (position 2-8) similarity, non-seed (position 9-end) similarity and genomic distance. Seed has been proven to be an important region in miRNA-target interaction which display an almost perfect match to the target sequence (Karginov et al., 2007), thus we suggest that seed similarity

between miRNAs is a potentially important feature. Additionally, including non-seed and sequence similarity features enables us to investigate the property behaviors of these two regions. Genomic distance is not a well investigated feature which is defined as base pair distance between two genes. The idea of investigating genomic distance between miRNAs is derived from our former study. Previously, through statistical methods and heterogeneous data support, we demonstrated that the genomic location feature plays a role in miRNA-target interaction for a selection of miRNA families (Zhang et al., 2007). Here we induce this idea to the study of miRNAs relationships based on the genomic distance.

In the data preparation, sequence similarity is calculated using the EBI pairwise global sequence alignment tool: i.e. Needle (Sankoff & Kruskal, 1999). Genomic sequence and location are retrieved from the miRBase Sequence Database. The distance between two miRNAs is calculated by genomic position subtraction when they are located on the same chromosome; otherwise it is set to undefined.

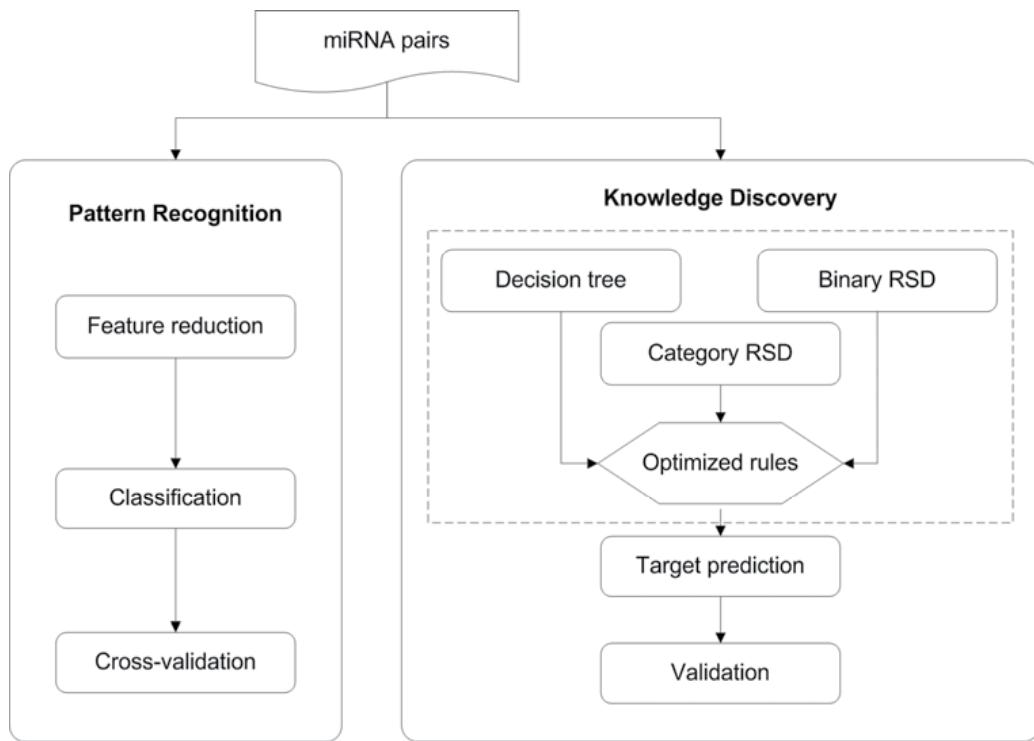


Fig. 2. Workflow. miRNA pairs are analyzed by both pattern recognition and knowledge discovery strategies.

3.3 Workflow

As showed in Fig. 2, we use two strategies to discover miRNA-miRNA relationships. In pattern recognition strategy, different classifiers are applied in order to discriminate positive and negative miRNA pairs. Then the performance of each classifier is evaluated by cross-validation. In knowledge discovery, rules are first discovered from three methods with respect to decision tree and relational subgroup discovery techniques. Through combining

the results, the optimized rules describing functionally alike miRNAs are generated which are used for final targets prediction and validation.

Pattern recognition: In this strategy, the first step is feature reduction. Features are selected by sequential backward elimination algorithm and extracted by principle component analysis. As it is known that sequential forward selection adds new features to a feature set one at a time until the final feature set is reached (Webb, 2002). It is simple and fast. The reason it is not applied in our experiment is due to the limitation that the selected features could not be deleted from the feature set once they have been added. This could lead to local optimum. After dimension reduction, classification is performed by both linear and quadratic classifiers. Finally, the performance is examined by 5-fold cross-validation with 10 repetitions. This part was implemented with PRtools (van der Heijden et al., 2004) a plugin for the MatLab platform.

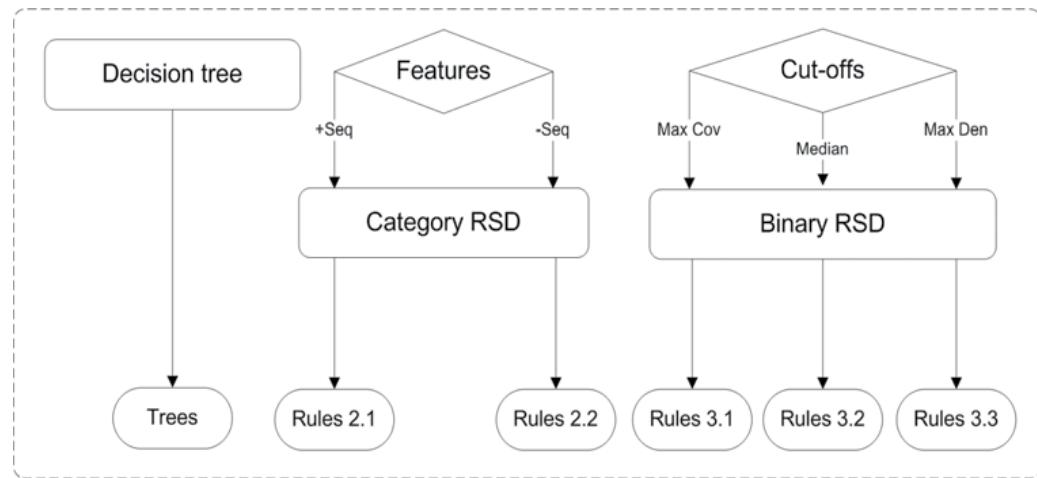


Fig. 3. Detailed experimental design in rule generation stage. Three methods are applied which are Decision tree, Category RSD and Binary RSD. In Category RSD, datasets are first categorized into groups. Subsequently, data with two feature sets, which are with and without overall sequence similarity, are used as the input to RSD algorithm. In Binary RSD, feature values are binarized using decision tree. Due to the fact that data are sampled 10 times, the cut-offs are then established using max coverage (Max Cov), median and max density (Max Den). Finally, RSD is applied to all 3 conditions in order to find out the feature cut-offs, which lead to the most significant rule sets.

Knowledge discovery: In pattern recognition the miRNA is classified through elaborate statistical models; in contrast, in knowledge discovery data patterns are described to allow us to increase our knowledge on the data. This could promote our understanding of functionally similar miRNAs. Furthermore, integration of this knowledge could finally promote target prediction. In this strategy, there are three phases: rule generation illustrated in the framework (dashed) of Fig. 2, target prediction and validation. In the first step, rules are discovered using decision trees and relational subgroup discovery. With the aim to discover the most significant rules, different data structures and feature thresholds are evaluated and compared. Details are explained in the following sections and an overview of this methodology is shown in Fig. 3.

Decision tree learning is utilized as a first step in order to build a classifier discriminating two classes of miRNA pairs. In our experiments, we used the decision tree from the Weka software platform (Witten & Frank, 1999). The features were tested using the J48 classifier and evaluated by 10 fold cross-validation.

Due to the fact that not all the determinant features are known at this stage, we are interested in finding rules for subgroups of functionally similar miRNAs with respect to our predefined features. In our experiments, we used the propositionalization based relational subgroup discovery algorithm (Zelezny & Lavrac, 2006). We prefer rules that contain only the positive pairs and portray high coverage. Consequently, the repetitive rules are selected, if their E-value is greater than 0.01 and at the same time the significance is above 10.

Both the Category RSD and the Binary RSD reveal feature patterns by utilizing the relational subgroup discovery algorithm. The main difference is that the former analyzes the data in a categorized format, whereas in later algorithm the data is transformed to a binary form.

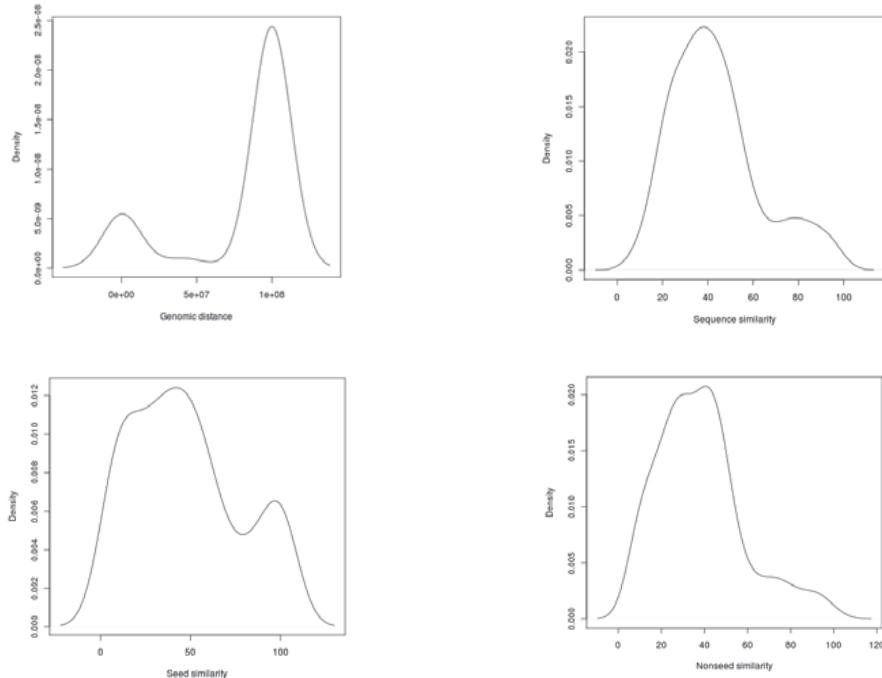


Fig. 4. Density plot for the four features. The plots of distance and seed similarity match bimodal distribution indicating two main groups in each feature. However it is not straightforward to judge sequence and non-seed similarity distributions.

As a pilot experiment for RSD, data is first categorized as follows: the similarity percentage is evenly divided into 5 groups: very low (0-20%], low (20-40%], medium (40-60%], high (60-80%], very high (80-100%]; Distance is categorized into 5 regions: 0-1kb¹, 1-10kb, 10-100kb, 100kb-end, undef (if miRNAs that are paired are located on a different chromosome). Two

¹ The unit of distance on a genome is base pair abbreviated as 'b', kb = kilo base pairs.

relational input tables, which are with and without the overall sequence similarity feature, are constructed and further tested with the purpose of verifying whether the sequence has a global effect or only contributes as the combination of seed and non-seed parts.

Through the observation of density graphs of the features, as depicted in Fig. 4, we concluded that distance and seed similarity feature densities match a bimodal distribution. The same conclusion can, however, not be drawn easily for overall and non-seed sequence similarities. Therefore, in this method, we apply a decision tree algorithm to discriminate 4 feature values into binary values. Each feature is calculated individually and only the root classifier value in the tree is used for establishing the cut-off. After that, binary tables are generated according to three criteria:

- **Maximum coverage** where the value covers the most positive pairs. Max coverage (distance, sequence, seed, non-seed) = 8947013 b, 56.5%, 71.4%, 53.3%
- **Median.** Median (distance, sequence, seed, non-seed) = 3679 b, 65.2%, 71.4%, 60.65%
- **Maximum density** which is the region with the highest positive pair density. Max density (distance, sequence, seed, non-seed) = 3679 b, 69.6%, 75%, 64.7%

4. Results

4.1 Classification

After application of sequential backward feature selection, features including genomic distance, seed similarity and non-seed similarity are selected as the top 3 informative features. Sequence similarity is the least informative feature because it is highly correlated to seed and non-seed similarities. Scatter plots of two classes of miRNA pairs in the selected feature space are depicted in Fig. 5. As can be seen in the four sub-graphs of Fig. 5, the majority of positive and negative miRNA pairs are overlapping which is an indication for the complexity of the classification. The distribution of negative class is more compact. We observed that the majority of this class located in the area of non-seed<60%, seed<70% and distance is infinite. Furthermore, we noticed that for those functionally similar miRNAs, seed similarity vary from 0 to 100%. This implies that miRNAs with the same or different seed sequence can bind the same targets. This is due to the fact that miRNAs can bind to the same targets at the same binding site which leads to high similarity and at different binding site resulting low similarity. The evaluation of the classifier performance shows that the average error and standard deviation for the quadratic classifier are 0.29739 and 0.01082, and for the linear classifier are 0.30987 and 0.0131.

In Fig. 6 the dataset is plotted in 2-dimensional PCA space in combination with the linear and quadratic classifiers. In this 2D projection, the average error and standard deviation for the quadratic classifier are 0.3029 and 0.00721, and for the linear classifier are 0.31657 and 0.00871.

With around 30% of classification errors, this means two classes are difficult to separate using features currently available. Furthermore, although the classifiers provide a statistical explanation and meaning, no biological insight is gained from them in order to be able to interpret the miRNA mechanism(s).

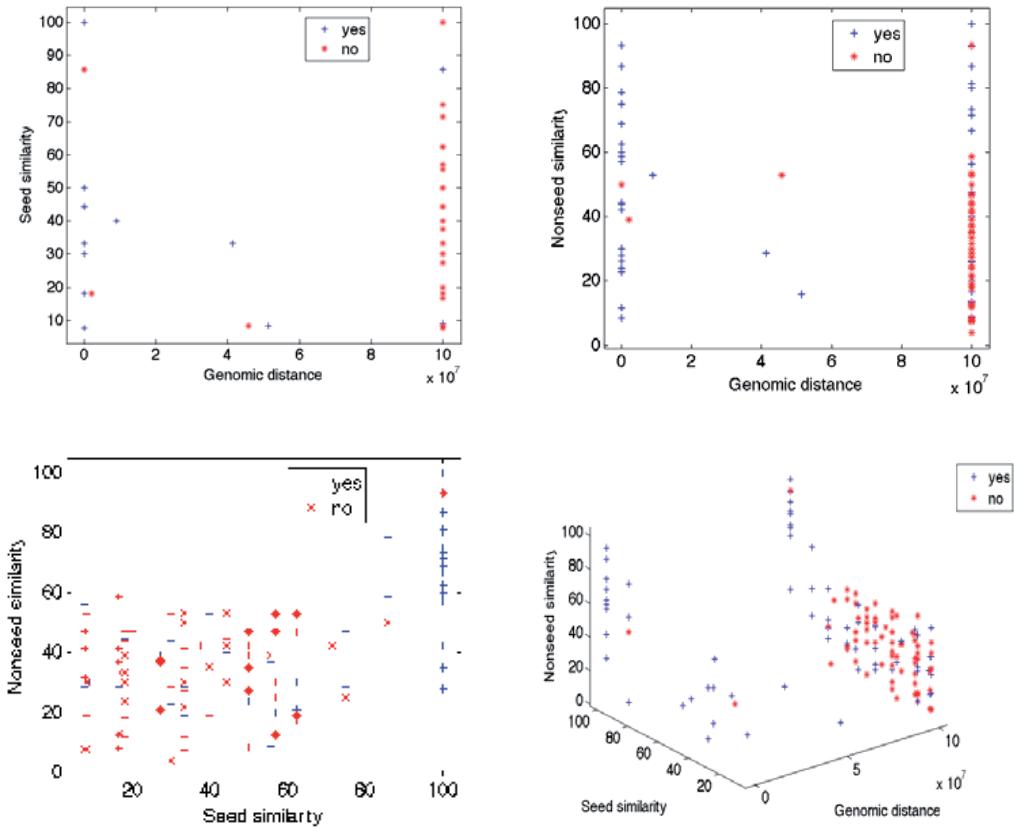


Fig. 5. Scatter plots of two classes of miRNA pairs in the selected feature spaces. Positive pairs are denoted using a token of plus (also in blue), while negatives are demonstrated by asterisk (red).

4.2 Rule discovery

In the decision tree analysis, several different tree structures are generated from 10 replications of the training data. Among them, the root attribute or the first depth of the tree is mainly associated with distance, sequence and seed similarity properties, while non-seed feature appeared only near the leaf nodes. This inconsistency in the tree structures indicated that none of the predefined features, or any combination of them, can significantly classify miRNAs.

The feature patterns discovered from Category RSD are listed in Table 1 where the rules in Table 1b take overall sequence into account but those in Table 1a do not. 'YES'-rules describe functionally similar miRNAs characterized by our predefined features. 'Significance' denotes the average significance over 10 replications. Further inspection of Table 1 shows that both rule sets consist of 3 main groups with features being Seed>80%, Dis<=1 kb and Dis=(1 kb,10 kb] labeled by A, B, C respectively. The remainder is the subset of these groups. Considering overall sequence in the rule generation results only the fourth rule (A.2) in Table 1a and 1b to be different. These results indicate that genomic location and

seed similarity between miRNAs are probably dominant features when deciding which miRNAs bind to the same target. Sequence information may be relevant but it is not as strong as seed and distance features.

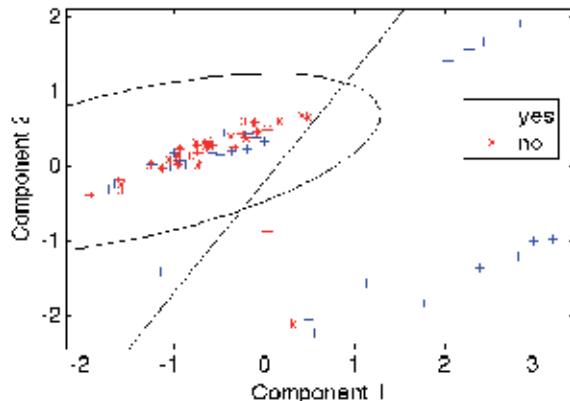


Fig. 6. Scatter plot of two classes of miRNA pairs in 2-dimensional PCA components space together with a linear discriminant classifier showed by a line and a quadratic discriminant classifier illustrated by an arc.

Label	-Overall sequence : YES Rules 2.1	Significance
A	Seed>80%	26.7
A.1	Dis=undef & Seed>80%	14.3
B	Dis<=1 kb	14.1
A.2	Seed>80% & Nonseed=(60%,80%)	12.6
C	Dis=(1 kb,10 kb]	11

(a)

Label	+Overall sequence: YES Rules 2.2	Significance
A	Seed>80%	26.7
A.1	Dis=undef & Seed>80%	14.3
B	Dis<=1 kb	14.1
A.2	Seed>80% & Seq=(60%,80%)	11.2
C	Dis=(1 kb,10 kb]	11

(b)

Table 1. Category RSD results. Rules generated from two data structures: considering overall sequence, seed, non-seed similarities as well as distance (a) and only seed, non-seed similarities and distance (b).

Table 2 shows the rules generated by Binary RSD, thereby using three cut-off criteria: Max coverage (a), Median (b) and Max density (c). As can be seen, three rule sets have similar structures but different feature cut-offs which lead to different significance. The main feature groups derived using max coverage, median and max density criteria respectively are Seed>71.4% (A) and Dis<=8947013 b (B) in rule set 3.1; Seed>71.4% (A), Dis<=3679 b (B)

and $\text{Seq} > 65.2\%$ (C) in rule set 3.2; and $\text{Seed} > 75\%$ (A), $\text{Dis} \leq 3679$ b (B) and $\text{Seq} > 69.6\%$ (C) in rule set 3.3. Others are the subsets of these groups.

Label	Max coverage: YES Rules 3.1	Significance
A.1	$\text{Seed} > 71.4\% \text{ & } \text{Seq} > 56.5\%$	30
A	$\text{Seed} > 71.4\%$	27.2
A.2	$\text{Nonseed} > 53.3\% \text{ & } \text{Seed} > 71.4\% \text{ & } \text{Seq} > 56.5\%$	21.6
B	$\text{Dis} \leq 8947013$ b	19.8
A.3	$\text{Nonseed} > 53.3\% \text{ & } \text{Seed} > 71.4\%$	18.2
A.4	$\text{Dis} > 8947013$ b & $\text{Seed} > 71.4\% \text{ & } \text{Seq} > 56.5\%$	13.5
A.5	$\text{Dis} > 8947013$ b & $\text{Seed} > 71.4\%$	12.3

(a)

Label	Median: YES Rules 3.2	Significance
A	$\text{Seed} > 71.4\%$	27.2
A.1	$\text{Seed} > 71.4\% \text{ & } \text{Seq} > 65.2\%$	23.3
B	$\text{Dis} \leq 3679$ b	23.3
B.1	$\text{Dis} \leq 3679$ b & $\text{Nonseed} \leq 60.65\%$	15.9
A.2	$\text{Dis} > 3679$ b & $\text{Seed} > 71.4\%$	14.9
A.3	$\text{Nonseed} > 60.65\% \text{ & } \text{Seed} > 71.4\% \text{ & } \text{Seq} > 65.2\%$	13.7
A.4	$\text{Nonseed} > 60.65\% \text{ & } \text{Seed} > 71.4\%$	13.7
C.1	$\text{Nonseed} > 60.65\% \text{ & } \text{Seq} > 65.2\%$	13.7
C	$\text{Seq} > 65.2\%$	12.2

(b)

Label	Max density: YES Rules 3.3	Significance
A	$\text{Seed} > 75\%$	26.7
B	$\text{Dis} \leq 3679$ b	23.3
A.1	$\text{Seed} > 75\% \text{ & } \text{Seq} > 69.6\%$	20.8
C	$\text{Seq} > 69.6\%$	20.8
B.1	$\text{Dis} \leq 3679$ b & $\text{Nonseed} \leq 64.7\%$	18
B.2/C.1	$\text{Dis} \leq 3679$ b & $\text{Seq} \leq 69.6\%$	14.1
A.2	$\text{Dis} > 3679$ b & $\text{Seed} > 75\%$	11.5
A.3/C.2	$\text{Nonseed} > 64.7\% \text{ & } \text{Seed} > 75\% \text{ & } \text{Seq} > 69.6\%$	11
C.3	$\text{Nonseed} > 64.7\% \text{ & } \text{Seq} > 69.6\%$	11

(c)

Table 2. Binary RSD results. Rules generated from 3 sets of parameters are shown in a sequence of Max coverage (a), Median (b) and Max density (c).

Furthermore, the rules with similar features but different feature values are compared. The decision on final cut-off is based on the value which results in the highest significance. Therefore the final optimized rules are:

Rule 1: IF distance between two miRNAs ≤ 3679 b,
Rule 2: IF seed similarity between two miRNAs $> 71.4\%$,

*Rule 3: IF sequence similarity between two miRNAs > 69.6%
THEN they bind the same target.*

To evaluate our methods, as a reference, a permutation test is performed. We repeat the learning procedure for each training set with the labels randomly shuffled. Using Max coverage as a cutoff criterion, we obtained that all the rules have the max significance lower than 8. This test therefore demonstrates that the rules derived from the original data are more significant compared to the random situation.

4.3 Target prediction

We apply the above rules searching for miRNAs which serve similar functions as the known miRNAs. Rule 1, 2 and 3 discovered 75, 655 and 150 miRNA pairs respectively in each subgroup which highly extends our previous findings (Zhang et al., 2008) based on similar methodology. Among them, 23 miRNA predicted targets which are covered by all of the 3 rules are selected for further validation, since this group has relative small pairs which are easy to validate. Furthermore, as they involve more constraints, it is considered to be more reliable.

By further inspection of these 23 miRNA pairs, we found that it consists of 3 confirmed pairs in which both individual miRNAs from each pair are well studied, 15 pairs with both members from the same family which are supposed to have the same targets, and 5 new pairs which have one well-studied miRNA and one functional unknown partner. Therefore, we induce the targets for these 5 unknown miRNAs hsa-miR-18a/ 18b/ 20b /212 /200c from their known partner. Their predicted targets are listed in Table 3.

Informatic validation is performed to check the prediction consistency with the existing methods. Table 3 shows validation for the 3 confirmed and 5 predicted miRNA pairs. The miRNAs with confirmed targets are indicated in italic, while the miRNAs in boldface are the unknown ones for which the targets are predicted from their known partners. All of their targets are validated by examining whether they are predicted by TargetScan, miRanda, Pictar, miTarget and RNAhybrid. For example the table can be read as following: whether the target (BCL2) is predicted by the existing methods (TargetScan) for m1 (hsa-miR 15a) or m2 (hsa-miR-16). Consequently, we discover that among our prediction, Retinoblastoma 1 (RB1) for hsa-miR-20b are predicted by TargetScan and Pictar; Circadian Locomoter Output Cycles Kaput (Clock) for hsa-miR-200c is captured by miRanda; Rho GTPase activating protein (RICS) for hsa-miR-212 is detected by Pictar; E2F transcription factor 1 (E2F1) and AIB1 for hsa-miR-18a are identified by miTarget.

Our prediction			Also predicted by									
miRNA1 (m1)	miRNA1(m2)	Targets	TargetScan		miRanda		Pictar		miTarget		RNAhybrid-mfe(kcal/mol)	
			m1	m2	m1	m2	m1	m2	m1	m2	m1	m2
<i>hsa-miR-15a</i>	<i>hsa-miR-16</i>	BCL2	√	√	✗	✗	√	√	✗	√	-24.3	-24.1
<i>hsa-miR-17</i>	<i>hsa-miR-20a</i>	E2F1	√	√	√	✗	√	√	√	√	-26.8	-24.6
<i>hsa-miR-221</i>	<i>hsa-miR-222</i>	KIT	√	√	✗	✗	✗	✗	√	√	-24.9	-26.4
<i>hsa-miR-17</i>	<i>hsa-miR-18a</i>	E2F1	√	✗	√	✗	√	✗	√	√	-26.8	-26.8
		AIB1	-	-	-	-	-	-	√	√	-26.3	-26.6
<i>hsa-miR-106a</i>	<i>hsa-miR-18b</i>	RB1	√	✗	✗	✗	√	✗	✗	✗	-23.2	-28.3
<i>hsa-miR-106a</i>	<i>hsa-miR-20b</i>	RB1	√	√	✗	✗	√	√	✗	✗	-23.2	-27.2
<i>hsa-miR-132</i>	<i>hsa-miR-212</i>	RICS	✗	✗	-	-	✓	✓	-	-	-	-
<i>hsa-miR-141</i>	<i>hsa-miR-200c</i>	Clock	✗	✗	√	√	✗	✗	✓	✗	-22.1	-20.1

Table 3. Informatic validation of confirmed and predicted miRNA pairs. miRNA1 and miRNA2 are the partners in one pair. Target column shows the validated targets for the known miRNAs (in italic) and the predicted targets for the unknown miRNAs (in boldface). m1 and m2 columns denote whether the targets are predicted by the existing methods for miRNA1 (m1) and miRNA2 (m2) respectively.

5. Conclusions and discussion

Machine learning is widely used in commercial businesses where vast amounts of data are produced. The life-sciences, molecular oriented research in particular, is a rapidly growing field which has gained a lot of attention lately especially now that the genomes of the major research model species have been sequenced and are publicly available. With the development of more and more large-scale and advanced techniques in biology, the need to discover hidden information triggered the application of machine learning in the field of the life-sciences. But these applications bear a risk, since, first of all, most biological mechanisms are not yet fully understood, and second, some techniques produce too little experimental data due to the limitations of these techniques, thereby making machine learning unreliable. In this chapter, we explained how we integrated different machine learning algorithms and tuned and optimized experimental setups to a growing but not yet mature research field, miRNA target prediction. The innovation of this approach is not only integration and optimization of machine learning algorithms, but also the prediction through new features in miRNA relationship instead of widely studied features of miRNA-target interaction. Existing methods for analysis have shown to be insufficient in identifying targets from this perspective.

As illustrated in the methods and results sections, pattern recognition generates models enabling class descriptions. In this case, a rather high misclassification error around 30% is surfacing. In contrast, subgroup discovery aims at discovering statistically unusual patterns of interesting classes (Zelezny & Lavrac, 2006). It discovers three main groups describing only the positive miRNA pairs.

One of the disadvantages of pattern recognition method is that the model is not biologically interpretable. Consisting of linear or quadratic transformations of features, the classifiers tell nothing about the mechanisms of miRNA-target binding. However decision tree and

relational subgroup discovery are descriptive induction algorithms which discover patterns in the form of rules. With these discovered rules, we gain knowledge about miRNA-target interaction which can, subsequently, be used to predict more targets.

We compared two main algorithmic approaches used in knowledge discovery. Given the circumstances that not all the targets and useful features are known in advance, the classification of miRNA data using decision trees is not recommended. However, the relational subgroup discovery, an advanced subgroup discovery algorithm, has shown to be suitable for this application domain since it can discover the rules for subgroups of similar function miRNAs with respect to our predefined features. During the rule mining, we also noticed that feature threshold optimization is a crucial procedure which helps revealing the significant rules.

We have established that distance, seed and sequence similarities are determinants. The question is whether it makes sense from the biological point of view. It has been reported that many miRNAs appear in clusters on a single polycistronic transcript (Tanzer & Stadler, 2004). They are transcribed together in a long primary transcript, yielding one or more hairpin precursors and finally are cut to multi-mature miRNAs. Tanzer *et al.* reported that the human mir-17 cluster contains six precursor miRNA (mir-17/ 18/ 19a/ 20/ 19b-1/ 92-1) within a region of about 1kb on chromosome 13 (Tanzer & Stadler, 2004). These observations are similar with the feature embedded in Rule 1 (*cf.* Section 4.2). Besides the fact that clustered miRNAs can be transcribed together, we further showed that miRNAs that are in close proximity to each other can bind to the same target so as to serve as the regulators for the same goal. In this study, we showed that the genomic location also contributes to miRNA target identification.

As for seed similarity, Rule 2 (*cf.* Section 4.2) describes that the miRNAs with seed similarity above 71.4% share the same targets. This means only a perfect match or one mismatch in the seed is allowed in the process of binding the same targets. This is consistent with the idea that seed is a specific region, in particular it requires a nearly perfect match with the target (Karginov *et al.*, 2007). Moreover, TargetScanS also only requires a 6-nt seed match comprising nucleotides 2-7 of the miRNA. Thus, the rule requiring at least 6 out of 7 nucleotides to be similar in seed region can be considered reasonable.

Overall sequence similarity is also a predictor but not as decisive as seed and genomic distance. This means that not only the seed region is important; sometimes two miRNAs with generally similar sequences can also bind to the same target. This is consistent with the finding that some miRNA-target interaction bindings have a mismatch or wobble in the 5' seed region but compensate through excellent complementarity at the 3' end, which leads to high average sequence complementarity (Maziere & Enright, 2007).

In order to support our findings, we validated the results using five existing algorithms presented in Table 3. Not all of the predicted targets are identified by TargetScan, miRanda, Pictar, miTarget and RNAhybrid, whereas this is the same case for the known targets. Most of the candidates are predicted by at least one of these methods. Both miTarget and our method are based on machine learning techniques; miTarget uses a support vector machine and considers sequence and structure features of miRNA-target duplexes whereas we focus the integration of several machine learning algorithms on the genomic location and sequence features between miRNAs. Moreover, we noticed that miRanda has a relatively low performance for target prediction in human. This may be due to the fact that miRanda was initially developed to predict miRNA targets in *Drosophila melanogaster*, and later

adapted to vertebrate genomes (Enright et al., 2003). In the application of RNAhybrid tool, a predefined threshold of the normalized minimum free energy (mfe) is lacking, we therefore decided to list the original values. We found that most of our predicted miRNA-target duplexes are more stable illustrated by the lower minimum free energy relative to the known ones.

In addition to these encouraging results, we also noticed that only groups of miRNA relationships are discovered by our method. Some miRNAs which are located far apart and whose seed similarity is low still have the same target. This indicated that besides genomic distance, seed and sequence similarities, more features need to be included in order to find more and better patterns shared by functionally alike miRNAs. Grimson et al. uncovered five general features of target site context beyond seed pairing that boost site efficacy (Grimson et al., 2007). In future research we will explore the site context in the miRNA relationship analysis. Additionally, we also consider taking into account miRNA co-expression patterns. In summary, we conclude that genomic distance, seed and sequence similarities are the determinants for describing the relationships of functionally similar miRNAs. Our method is complementary to the approaches that are currently used. It contributes to the improvement of target identification by predicting targets with high specificity. Moreover, it does not require conservation information for classification, so it is free from the limitations of some of the existing methods. In future research, with more biologically validated targets and features available, more rules can be generated from a large dataset, and consequently more targets can be identified to the functionally unknown miRNAs. The methodology can be transferred to a broad range of other species as well.

6. Acknowledgements

We would like to thank Dr. Erno Vreugdenhil for discussing some biological implications of the results and Peter van de Putten for suggestions on the use of WEKA. This research has been partially supported by the BioRange program of the Netherlands BioInformatics Centre (BSIK grant).

7. References

- Bartel, D. P. (2004). MicroRNAs: Genomics, Biogenesis, Mechanism, and Function. *Cell*, Vol. 116, No. 2, 281-297
- Bentwich, I. (2005). Prediction and validation of microRNAs and their targets. *FEBS Lett*, Vol. 579, No. 26, 5904-5910
- Brennecke, J., Hipfner, D. R., Stark, A., Russell, R. B., & Cohen, S. M. (2003). bantam encodes a developmentally regulated microRNA that controls cell proliferation and regulates the proapoptotic gene hid in *Drosophila*. *Cell*, Vol. 113, No. 1, 25-36
- Chen, C. Z. (2005). MicroRNAs as oncogenes and tumor suppressors. *N Engl J Med*, Vol. 353, No. 17, 1768-1771
- Enright, A. J. & Griffiths-Jones, S. (2007). miRBase: a database of microRNA sequences, targets and nomenclature. In: *microRNAs: From Basic Science to Disease Biology*, K. Appasani, S. Altman, & V. R. Ambros (Eds.), pp. 157-171, Cambridge University Press

- Enright, A. J., John, B., Gaul, U., Tuschl, T., Sander, C., & Marks, D. S. (2003). MicroRNA targets in *Drosophila*. *Genome Biol*, Vol. 5, No. 1
- Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recogn.Lett.*, Vol. 27, 861-874
- Griffiths-Jones, S. (2004). The microRNA Registry. *Nucleic Acids Res*, Vol. 32
- Griffiths-Jones, S., Grocock, R. J., van Dongen, S., Bateman, A., & Enright, A. J. (2006). miRBase: microRNA sequences, targets and gene nomenclature. *Nucleic Acids Res*, Vol. 34
- Grimson, A., Farh, K. K.-H., Johnston, W. K. K., Garrett-Engele, P., Lim, L. P. P., & Bartel, D. P. P. (2007). MicroRNA Targeting Specificity in Mammals: Determinants beyond Seed Pairing. *Mol Cell*, Vol. 27, No. 1, 91-105
- Grun, D. & Rajewsky, N. (2007). Computational prediction of microRNA targets in vertebrates, fruitflies and nematodes. In: *microRNAs: From Basic Science to Disease Biology*, K.Appasani, S. Altman, & V. R. Ambros (Eds.), pp. 172-186, Cambridge University Press
- He, L., Thomson, M. M., Hemann, M. T., Hernando-Monge, E., Mu, D., Goodson, S. et al. (2005). A microRNA polycistron as a potential human oncogene. *Nature*, Vol. 435, No. 7043, 828-833
- Johnson, S. M., Grosshans, H., Shingara, J., Byrom, M., Jarvis, R., Cheng, A. et al. (2005). RAS is regulated by the let-7 microRNA family. *Cell*, Vol. 120, No. 5, 635-647
- Karginov, F. V., Conaco, C., Xuan, Z., Schmidt, B. H., Parker, J. S., Mandel, G. et al. (2007). A biochemical approach to identifying microRNA targets. *Proceedings of the National Academy of Sciences*, 19291-19296
- Kim, S. K., Nam, J. W., Rhee, J. K., Lee, W. J., & Zhang, B. T. (2006). miTarget: microRNA target-gene prediction using a Support Vector Machine. *BMC Bioinformatics*, Vol. 7,
- Krek, A., Grun, D., Poy, M. N., Wolf, R., Rosenberg, L., Epstein, E. J. et al. (2005). Combinatorial microRNA target predictions. *Nature Genetics*, Vol. 37, No. 5, 495-500
- Lavrac, N., Zelezny, F., & Flach, P. A. (2003). RSD: Relational Subgroup Discovery through First-Order Feature Construction. In *Proceedings of the 12th International Conference on Inductive Logic Programming* (pp. 149-165). Springer-Verlag
- Lecellier, C. H., Dunoyer, P., Arar, K., Lehmann-Che, J., Eyquem, S., Himber, C. et al. (2005). A cellular microRNA mediates antiviral defense in human cells. *Science*, Vol. 308, No. 5721, 795-825
- Lee, R. C., Feinbaum, R. L., & Ambros, V. (1993). The *C. elegans* heterochronic gene lin-4 encodes small RNAs with antisense complementarity to lin-14. *Cell*, Vol. 75, No. 5, 843-854
- Lewis, B. P., Burge, C. B., & Bartel, D. P. (2005). Conserved seed pairing, often flanked by adenosines, indicates that thousands of human genes are microRNA targets. *Cell*, Vol. 120, No. 1, 15-20
- Lewis, B. P., Shih, I. H., Jones-Rhoades, M. W., Bartel, D. P., & Burge, C. B. (2003). Prediction of mammalian microRNA targets. *Cell*, Vol. 115, No. 7, 787-798
- Martin, G. (2007). Prediction and validation of microRNA targets in animal genomes. *J Biosci*, Vol. 32, No. 6, 1049-1052
- Maziere, P. & Enright, A. J. (2007). Prediction of microRNA targets. *Drug Discov Today*, Vol. 12, No. 11-12, 452-458

- Papadopoulos, G. L., Reczko, M., Simossis, V. A., Sethupathy, P., & Hatzigeorgiou, A. G. (2008). The database of experimentally supported targets: a functional update of TarBase. *Nucleic Acids Research*, Vol. 37, No. Database issue, D155-D158
- Rehmsmeier, M., Steffen, P., Hochsmann, M., & Giegerich, R. (2004). Fast and effective prediction of microRNA/target duplexes. *RNA*, Vol. 10, No. 10, 1507-1517
- Reinhart, B. J., Slack, F. J., Basson, M., Pasquinelli, A. E., Bettinger, J. C., Rougvie, A. E. et al. (2000). The 21-nucleotide let-7 RNA regulates developmental timing in *Caenorhabditis elegans*. *Nature*, Vol. 403, No. 6772, 901-906
- Sankoff, D. & Kruskal, J. (1999). *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Center for the Study of Language and Inf
- Sethupathy, P., Corda, B., & Hatzigeorgiou, A. G. (2006a). TarBase: A comprehensive database of experimentally supported animal microRNA targets. *RNA*, Vol. 12, No. 2, 192-197
- Sethupathy, P., Megraw, M., & Hatzigeorgiou, A. G. (2006b). A guide through present computational approaches for the identification of mammalian microRNA targets. *Nature Methods*, Vol. 3, No. 11, 881-886
- Sethupathy, P., Megraw, M., & Hatzigeorgiou, A. G. (2007). Computational approaches to elucidate miRNA biology. In: *microRNAs: From Basic Science to Disease Biology*, K.Appasani, S. Altman, & V. R. Ambros (Eds.), pp. 188-198, Cambridge University Press
- Tanzer, A. & Stadler, P. F. (2004). Molecular evolution of a microRNA cluster. *J Mol Biol*, Vol. 339, No. 2, 327-335
- van der Heijden, F., Duin, R., de Ridder, D., & Tax, D. M. J. (2004). *Classification, Parameter Estimation and State Estimation: An Engineering Approach Using MATLAB*. John Wiley & Sons
- Webb, A. R. (2002). *Statistical Pattern Recognition*. John Wiley and Sons Ltd.
- Witten, I. H. & Frank, E. (1999). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann
- Wuchty, S. a. F. W. a. H. I. L. a. S. P. (1999). Complete suboptimal folding of RNA and the stability of secondary structures. *Biopolymers*, Vol. 49, 145-165
- Zelezny, F. & Lavrac, N. (2006). Propositionalization-based relational subgroup discovery with RSD. *Machine Learning*, Vol. 62, No. 1-2, 33-63
- Zhang, Y., de Bruin, J. S., & Verbeek, F. J. (2008). miRNA target prediction through mining of miRNA relationships. *BioInformatics and BioEngineering*, 1-6
- Zhang, Y., Woltering, J. M., & Verbeek, F. J. (2007). Screen of MicroRNA Targets in Zebrafish Using Heterogeneous Data Sources: A Case Study for Dre-miR-10 and Dre-miR-196. *International Journal of Mathematical, Physical and Engineering Sciences*, Vol. 2, No. 1, 10-18

Extraction Of Meaningful Rules In A Medical Database

Sang C. Suh, Nagendra B. Pabbisetty and Sri G. Anaparthi
*Texas A&M University - Commerce
U. S. A.*

1. Introduction

Data Mining has become a prominent approach in recent years for generating rules within databases which concentrates on generating valuable information for decision making. Clustering, in Data Mining is the problem of identifying the distribution of patterns and intrinsic correlations in large data sets by portioning the data points into similarity classes. The traditional clustering algorithms which use distance between points define boundaries in clustering. But, these clustering mechanisms don't apply on boolean and categorical attributes. The clustering is applied on unstructured data to extract knowledge that may take a form of predictive rules.

Clustering enhances the value of existing databases by revealing rules in the data. These rules are useful for understanding trends, making predictions of future events from historical data, or synthesizing data records into meaningful clusters. Through clustering are similar data items grouped together to form clusters. Clustering algorithms usually employ a distance metric based (e.g., Euclidean) similarity measure in order to partition the database such that data points in the same partition are more similar than points in different partitions. In this chapter, we study clustering algorithms for data with categorical attributes. Instead of using traditional clustering algorithms that use distances between points for clustering which is not an appropriate concept for boolean and categorical attributes, we propose a novel concept of HAC (Hierarchy of Attributes and Concepts) to measure the similarity/proximity between a pair of data points.

Hierarchical clustering is one of the most frequently used methods in unsupervised learning. Given a set of data points, the output is a binary tree whose leaves are the data points and whose internal nodes represent nested clusters of various sizes. The tree organizes these clusters hierarchically, where the hope is that this hierarchy agrees with the intuitive organization of real-world data.

Hierarchical structures are ubiquitous in the natural world. The idea of hierarchical (also known as agglomerative) clustering is to begin with each point from the input as a separate cluster. We then build clusters by, repeatedly merging the two clusters that are closest in features from the initial points. This gives a hierarchy of containment, as each point in the input belongs to a succession of larger clusters. If we keep merging, we end up with a single cluster that contains all points, and the structure of the hierarchy can be represented as a

(binary) tree. From this tree we can extract a set of k clusters. For example, terminating the merging when only k clusters remain, or when the closest pair of clusters are at a distance exceeding some threshold. The crucial part of this algorithm is to define a metric to measure the distance between two clusters of multiple points. One kind of definition that uses mathematical model is: the smallest distance between a point in one cluster and a point in another; the greatest distance between such points; or the average distance. Each definition has its own advantages and disadvantages.

This research focuses on hierarchical conceptual clustering in structured, discrete-valued databases. By structured data, we refer to information consisting of data points and relationships between the data points. This differs from a definition of unstructured data as containing free text and structured data containing feature vectors.

Conceptual clustering is an important way of summarizing and explaining data [1, 6]. However, the recent formulation of this paradigm has allowed little exploration of conceptual clustering as a means of improving performance. Furthermore, previous work in conceptual clustering has not explicitly dealt with constraints imposed by real world environments. This chapter presents a clustering using HAC (Hierarchy of attributes and concepts), which is a hierarchical conceptual clustering system that organizes data so as to maximize inference ability. This algorithm uses both the hierarchical and conceptual clustering methods to implement clustering by discovering substructures in database which compress the original data and represent structural concepts in the data. Once a substructure is discovered, the substructure is used to simplify the data by replacing instances of the substructure with a pointer to the substructure definition. The discovered substructures allow abstraction over detailed structures in the original data. Iteration of the substructure discovery process constructs a hierarchical description of the structural data in terms of the discovered substructures. This hierarchy provides varying levels of interpretation that can be accessed based on the specific data analysis goals.

An important property of the conceptual clustering is that, it can enhances the value of existing databases by revealing patterns in the data. These patterns may be useful for understanding trends, for making predictions of future occurrences from historical evidence, or for synthesizing data records into meaningful clusters. A conceptual clustering system accepts a set of object descriptions (events, observations, facts) and produces a classification scheme over the observations. These systems use an evaluation function to determine classes with "good" conceptual descriptions. A learning of this kind is referred to as learning from observation (as opposed to learning from examples). Typically, conceptual clustering systems assume that the observations are available indefinitely so that batch processing is possible using all observations.

In this study, HAC will be used as an aid to represent medical domain knowledge substructures to simplify the generation process of the databases through clustering. As a result, the research will identify interesting relationships and patterns among the data, and represent them in the form of association rules.

2. Related Work

Clustering is the unsupervised classification of patterns (observations, data items or feature vectors) into groups (clusters). The clustering problem has been addressed in many contexts and by researchers in many disciplines; this reflects its broad appeal and usefulness as one

of the steps in exploratory data analysis. However, clustering is a difficult problem combinatorial and differences in assumptions and contexts in different communities have made the transfer of useful generic concepts and methodologies slow to occur [2].

Hierarchical clustering is one of the most frequently used methods in unsupervised learning. Given a set of data points, the output is a binary tree (dendrogram) whose leaves are the data points and whose internal nodes represent nested clusters of various sizes. The tree organizes these clusters hierarchically, where the hope is that this hierarchy agrees with the intuitive organization of real-world data. Hierarchical structures are ubiquitous in the natural world [5].

There are two general approaches to hierarchical clustering: top-down and bottom-up. The top-down approach starts with a cluster containing all points that is recursively split until enough sub clusters have been created. Heckel et al. [10, 11] use this approach on vector fields where each point has a position and a vector. The cluster with the highest error value will split into two clusters recursively so that the error values of the remaining clusters decrease with each split.

The bottom-up approach starts with all points as individual clusters and merges the two clusters with least difference, until one big cluster has been formed from all clusters. Telea and van Wijk [10] use this approach to simplify complex vector fields using an elliptic similarity function. They merge the pair of vectors with the least position, magnitude and direction differences until all vectors have been merged into one single vector.

Conceptual clustering is used to summarize the result. It enhances the value of existing databases by revealing patterns in the data. These patterns may be useful for understanding trends, for making predictions of future occurrences from historical evidence, or for synthesizing data records into meaningful clusters. The hybrid conceptual clustering [3] is used to handle both the incremental and non incremental problems for clustering successfully but it is computationally expensive moreover can be applied on small data sets. In past decades, many conceptual clustering algorithms have been proposed which can automatically acquire knowledge or concepts from large amounts of information acquired from experience or observation [1, 7, 8, 9]. Concepts in COBWEB are represented by probabilistic expressions and are acquired by using four learning operators and an evaluation function called category utility. But the category utility used in original COBWEB has a bias to prefer larger size classes in concept hierarchy. This bias produces some spurious intermediate nodes in concept hierarchy (classification tree). These nodes make tree deeper and complex, so we can't understand concepts within the nodes of tree easily [9].

This chapter presents an efficient non-metric measure called HAC (Hierarchy of Attributes and Concepts) for clustering of categorical as well as non-categorical(quantitative) data through which the proximity and relationships between data items can be identified.

3. Information

3.1 Hierarchy of Attributes and Concepts

The present paper introduces a hierarchical description of concepts by attributes, mathematical formalization presents the concepts as matrices whose columns represents terms constructed by attributes. A spherical model is developed to present a vocabulary

(spanned space) of concepts. In the beginning of this section the following definitions are introduced.

Attribute: Is a basic characteristic or a feature of a term.

Term: Is considered as a set of connected attributes.

Concept: Is a language independent meaning associated with at least one term, or set of terms.

Vocabulary: Is a set of terms and concepts.

HAC is both a hierarchical and conceptual clustering system that organizes data to maximize inference ability. This algorithm implements clustering by discovering substructures in database which compress the original data and represent structural concepts in the data. Once a substructure is discovered, it is used to simplify the data by replacing instances of the substructure with a pointer to the substructure definition. The discovered substructures allow abstraction over detailed structures in the original data. Iteration of the substructure discovery process constructs a hierarchical description of the structural data in terms of the discovered substructures. This hierarchy provides varying levels of interpretation that can be accessed based on the specific data analysis goals. HAC accepts database of structured data (concepts) as input. This type of data is naturally represented using a graph or diagrammatical form. The graph representation includes labeled vertices with vertex ID (identification) numbers, attributes on X-axis and downward directed edges (see Fig 1). Each vertex represents a concept and the value of that concept is given by the directed edges (usually map to the attribute's value on X-axis or to some other concept).

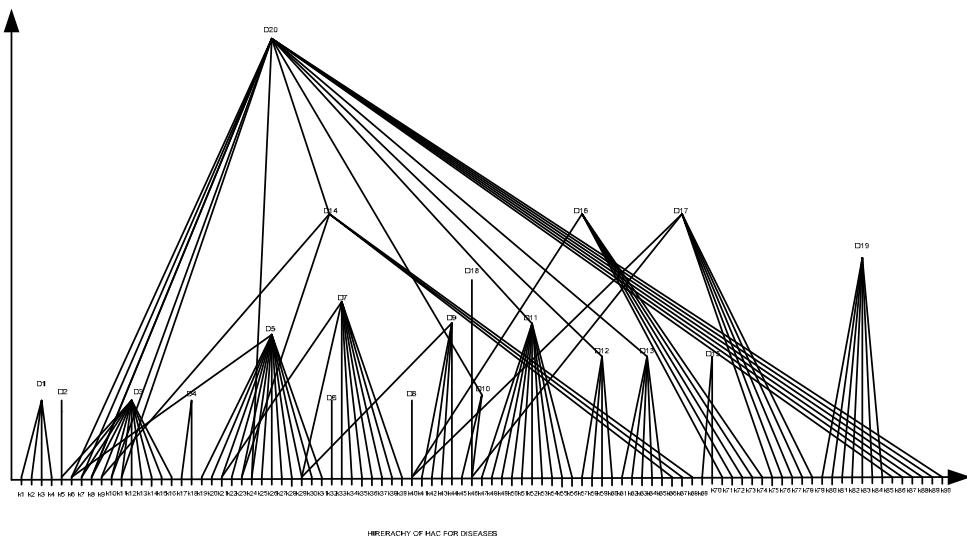


Fig. 1. The Graphical representation of HAC

ID	Disease Name	Causes Id
1	Acute sinusitis	K1,K2,K3,K4
2	Anaphylaxis	K5,K6,K7,K8,K9,K10,K11,K12,K13,K14,K15,K16
3	Penicillin allergy	K6
4	Latex allergy	K17,K18
5	Peanut allergy	K6,K19,K20,K21,K22,K23,K24,K25,K26,K27,K28, K29,K30,K31
6	Mold allergy	K32
7	Nickel allergy	K21,K29,K33,K34,K35,K36,K37,K38,K39
8	Dust mine allergy	K40
9	Aspergillosis	K29,K41,K42,K4,K44
10	Soy allergy	K45,K46
11	Shellfish allergy	K47,K48,K49,K50,K51,K52,K53,K54,K55,K56
12	Wheat allergy	K57,K58,K59,K60
.....
20	Food allergy	K6,K7,K8,K9,K10,K11,K23,K24,K45,K46,K47,K4, K49,K50,K51,K52,K53,K54,K56,K57,K58,K59,K60 ,K62,K63,K64,K65K66,K67,K68,K88,K89,K90,K91 ,K92,K93

Table 1. The HAC attribute Table based on attribute allergies.

Using the graph we can create a concept table for every attribute in the database and this table will have all the concepts and their value which are relevant according to the attribute. Ultimately with the help of these concept tables we can get a table where each record will have an attribute name, its value and relationship with other concepts. Each attribute in such a table represents a cluster.

For example Table1, which is formed from set of documents within the domain of education, consist of three columns. The second column specifies the attribute name and the third column specifies the attribute values. Now from table 1 we form another table called Concept Table (Table 2). Every row in this table (Table 2) consists of three fields. The first field shows the concept name whereas the second and third attributes represents concept value and concept attributes.

For example, the Concept Name C1 whose value is sense is a concept formed from the attributes F1,F2,F12,F17,F21,F36,F54 which represents various Food ID's. Recall that HAC can be represented as a closed diagrammatical entity shown in Fig 2.

Food Category ID	Food Category	Food ID
C1	Dairy	F1,F2,F12,F17,F21,F36,F54
C2	Seafood	F3,F4,F73,F74,F75,F76,F77,F78
C3	Poultry	F18,F40,F48,F26,F10,F30
C4	Grains	F7,F16,F29,F33,F50,F68,F69,F70,F8
C5	Nuts	F5,F90,F84,F91,F93
C6	Fruits	F32,F37,F71,F72,F81,F82,F85,F86,F87,F89,F92,F95
C7	Vegetables	F6,F53,F55,F58,F67,F80,F83
C8	Bakery	F11,F13,F14,F20,F43,F45,F51
C9	Drinks	F9,F15,F24,F25,F35,F39
C10	Fat items	F22,F28,F41,F5
C11	Seeds	F42,F60,F61
C12	Leafy and vegetables	F52,F57,F64,F65,F79,F88
C13	Junk foods	F19,F23,F27,F31,F38,F49,F44,F46,F47,F94
C14	Spices	F34,F56,F62,F63,F66

Table 2. The HAC concept Table based on the attribute Food Category.

3.2 Approach

HAC is both a hierarchical and conceptual clustering system that organizes data so as to maximize inference ability. This algorithm implement clustering by discovering substructures in database which compress the original data and represent structural concepts in the data. Once a substructure is discovered, the substructure is used to simplify the data by replacing instances of the substructure with a pointer to the substructure definition. The discovered substructures allow abstraction over detailed structures in the original data. Iteration of the substructure discovery process constructs a hierarchical description of the structural data in terms of the discovered substructures. This hierarchy provides varying levels of interpretation that can be accessed based on the specific data analysis goals.

HAC accepts database of structured data (concepts) as input. This type of data is naturally represented using a graph. The graph representation includes labeled vertices with vertex id numbers, attributes on X-axis and downward directed edges. Each vertex represents a concept and the value of that concept is given by the directed edges (usually map to the attribute's value on X-axis or to some other concept). With this graph we can make a concept table for every attribute in the database and this table will have all the concepts and their value which are relevant according to the attribute. Ultimately with the help of these

concept tables we can get a table in which each record will have an attribute name, its value and relationship with different concepts. Each attribute in this table represents a cluster.

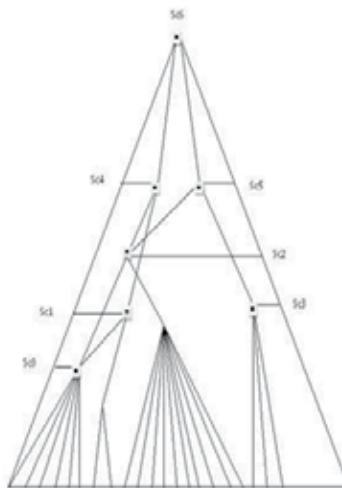


Fig. 2. The Triangular representation of the HAC.

4. Spherical Model

The graph model developed in the previous section is useful for connection purposes. To provide an opportunity to generate and manipulate concepts a matrix form is considered here after.

$$C_{k \times n} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \dots & \dots & \dots & \dots \\ a_{k1} & a_{k2} & \dots & a_{kn} \end{pmatrix}$$

Every column in $C_{k \times n}$ represents a term and every a_{ij} represents an attribute. Thus every concept could be decomposed to set of independent terms and every term be generated by pivot element. Therefore every concept (matrix) could be transformed to set of linearly independent terms (columns), created by linearly independent attributes (enteries). As a consequence every concept will have an invertible minor and the dimensions of this minor will be called a rank of the concept. The set of linearly independent terms T_i where $i = 1, 2, \dots, n$ over the field of real numbers may span the entire set of concepts.

$$T = \text{span } \{T_i\} \quad i=1, \dots, m \quad (1)$$

If the dimension of the set T is too large the matrix form of concepts presentation would happen to be memory inefficient. To overcome this obstacle a spherical model is developed to represent terms and concepts generation. The central section of the sphere represents the entire set of linearly independent attributes used to build up the basis T_i where $i = 1, 2, \dots,$

m. The next part forms the terms and is followed by concepts. Finally the model is ending with only one highest point as shown in figure 3.

If the total number of terms, coming from a certain domain, is n then the maximum number of concepts to be generated by $(n-1)$ terms over a numerical field of 1 element (number) is

$$\binom{n}{n-1}.$$

But if $n-2$ elements are used over the same numeric field the number of composed concepts would be $\binom{n}{n-2}$.

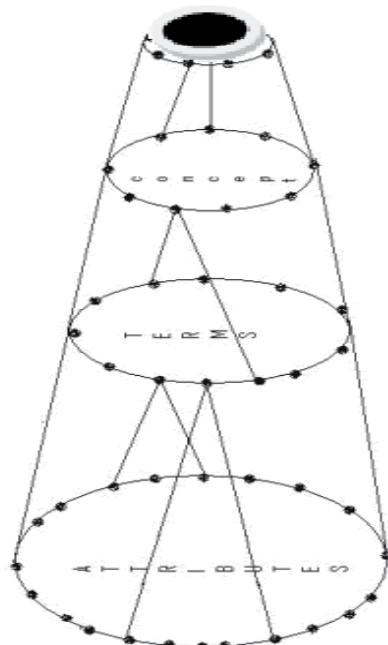


Fig. 3. The Spherical Model

Therefore the total number of concepts to be generated over a field with 1 element and n terms is 2^n . If the numeric field consists of j elements (numbers), and the maximum size of a term (column) is k then the total number of concepts to be generated is $j^k 2^n$. From theoretical view point it could be a very big number, but in practice the number of concepts would be smaller because some terms could happen to be mutually contradictory and cannot be linked in a single concept.

5. Algorithm

On the basis of the theoretical concepts an algorithm is developed to form an HAC.
Algorithm Forming_of_HAC_Clusters (A_Name [], A_Values [])

1. Make concept tables for different attributes using HAC. In this basically we have to make such that each level or concept is meaningful.
2. Now, make a "Cluster Point Table" which will have attribute name, attribute value and relationship with different concepts. Attribute name is basically the name of cluster and each and every attribute name is associated with an attribute value
3. Concept will give the value of cluster which is a combination of different attributes and concepts.
4. Concept values can be generated by using following steps:
5. Let total_num= Total number of attributes values in given table;
6. While I <= total_num
Begin:
Num=1;
 AC_1=total number of Attribute or concepts in concept table;
 AC_2=total number of Attribute or concepts in concept table;
 If AC_1 and AC_2 forms a cluster
 C_num= AC_1 * AC_2;
 Num++;
 End if
 Increment I;
End.
7. For i=1 to num
 V_i = total number of records under different combination of C1_(j, j+1---m) and C2_(k, k+1---n)
 Here, V_(i, i+1----total_num) are different clusters or attributes in Cluster Point Table.
 C1_(j, j+1---m) are different concepts of concept table1
 C2_(k, k+1---m) are different concepts of concept table2

From this Cluster Point table and Concept tables (made through HAC) we can generate rules.

The above algorithm is used to form the concepts and form an HAC from the given attribute values. The same algorithm is used if an update of an HAC is needed. The algorithm is capable of adding new concepts, terms and attributes.

6. Case Study: Clustering Using the HAC

This is an example between various allergy diseases and food categories in an allergy database. First a chart of various concepts is formed in which each concept will have allergy disease name as its value. Note that it could include multiple entries.

6.1 HAC1

In Table 1 which is an attribute table for allergy disease, the attribute names are mapped with its values for reference purposes. The concepts D1 through D20 in the TACR for the attribute "Allergy diseases" shown in Table 3 summarize the concepts in the HAC hierarchy of Figure 1 that hold among different attributes and concepts of diseases.

Note that the root concept D20 embraces all concepts and attributes of "Food related allergy diseases."

Cause ID	Cause
K1	cold virus
K2	bacterial or fungal
K3	deviated nasal septum
K4	nasal polyps
K5	Drugs
K6	Peanuts
K7	tree nuts
K8	Milk
K9	Eggs
K10	Fish
K11	Shellfish
K12	Insect stings from bees
.....
K93	Mushrooms

Table 3. TACR for Allergy Diseases

6.2 HAC2

Similarly, we can build the HAC structure for another attribute “Food category” as shown in Figure 4. The hierarchy shows a conceptual relationship among the values and concepts related to the attribute “Foods”. For this purpose the attribute table in Table 2 is used.

Food ID	Food values
F1	Eggs
F2	Milk
F3	Fish
F4	Shellfish
F5	Tree nut
F6	Beans
F7	Wheat
F8	Gluten
F9	Alcoholic drinks
F10	Beef
F11	Bread
F12	Butter
...
F95	Cherry

Table 4. TACR for Food Category

The concepts C1 through C14 in the TACR for the attribute “Food Category” shown in Table 4 summarize the concepts in the HAC hierarchy of Figure 4 showing the relationships among different values and concepts associated with Food Category.

Using the TACRs for the selected attributes, Diseases and Food Category in this case, a graph can be constructed to represent clusters at concept levels. The graph is shown in Figure 5. Each point in the graph represents a concept that is formed by a combination of attribute values. Let's call this point in the graph a Concept Cluster Point (CCP). Since the cluster point represents a high level concept, it naturally converts to a rule that matches with a concept. Furthermore, the support of each rule can be given by calculating the number of contributing entries in the original relational table to form the concept.

Conceptual Points	Support	Concepts (diseases)	Concepts (food category)
V1	5	D5	C5
V2	2	D10	C5
V3	6	D11	C2
V4	6	D12	C4
V5	2	D13	C1
-----	---	-----	----
V15	1	D20	C6
V16	1	D20	C7

Table 5. Cluster Point Table

In Table 5 above, the CCPs V1 through V9 each represent a concept cluster representing a high level concept associated with departments and degrees with a support value. Hence the next step is to convert these CCPs into characteristic rules. To help this process, the CCP graph in Figure 5 is used. Note that each CCP is associated with an appropriate support value which represents the weight to support the converted rule representing the concept. Each CCP in Figure 5 represents rules regarding how many people affected with specific allergy affected by food category and concepts. A rule can directly be generated from the cluster point table with the support from the attribute tables and TACRs. The cluster point table is shown as cluster point graph, each point in the graph represents rule associated with support. For example, if we want to query how the milk allergy is affected by dairy products.

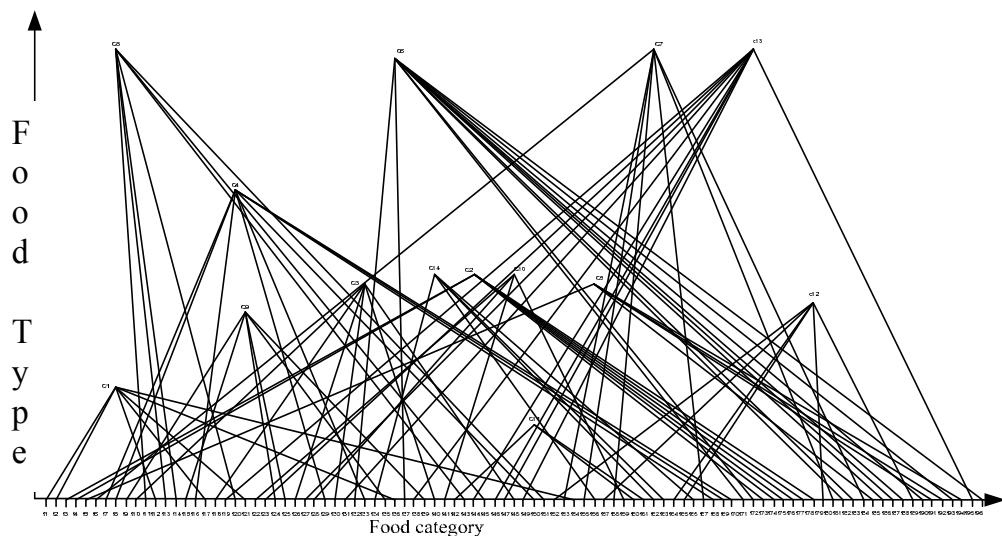


Fig. 4. HAC for Food and Foodtypes

In this example, the milk allergy has D13 and the dairy product is C1, both the combination of D13 and C1 represents V3 that has support 5.

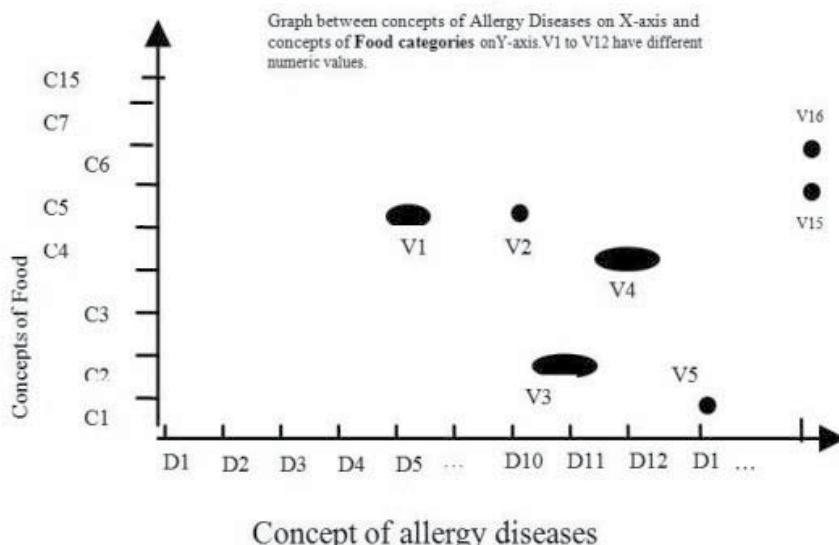


Fig. 5. Cluster Point Graph

7. Implementation

We have used a medical database which contains information about patient causes and food types. This is generated syntactically based on the case studies found in the websites [13, 14].

We used 2 concept tables for implementation of our algorithm; first concept table is for attribute 'Allergy diseases' and second concept table is for attribute 'Food categories'.

Each concept table will have a different hierarchy level for respective attributes, example in Concept table for attribute 'Allergy disease' concept D1 (Acute sinusitis) represents hierarchy level of cold virus, bacterial or fungal, deviated nasal septum, nasal polyps. Similarly concept table for the attribute 'Food categories' represents different hierarchy level of degrees. Using these two concept tables we made 'Cluster Point Table' which has all the possible clusters, each cluster is a combination of 'Allergy disease' attribute and 'Food categories' attribute.

We used non-numeric or categorical data for our clustering algorithm. This algorithm is implemented in 'JAVA' with a simple user interface which makes it very easy to use. User just needs to select different concepts of different attributes, concepts are dynamically generated from concept tables in database. After selecting the concepts the user needs to press the 'Submit' button that will display the result or you can say rules for our 'Cluster Point Graph' on an additional results page. 'Cluster Point Graph' represents all the clusters possible from different combinations of different concepts.

8. Result

Depending on the concept tables the results are generated in the form of rules. Each row in the 'cluster point table' represents a cluster that takes a form of rule. The total number of rules depends on the number of concept tables. Here we are using two concept tables that have generated rules has follows:

1. The two concept tables used are Allergy Diseases and Food Categories. The sample rules generated such as:

- a) Most likely, the persons are affected by specific allergy due to specific food category.
- b) The total number of persons with specific values of different concepts.

From the user perspective the user selects the disease and food category then it shows the resulted cluster value from cluster point table. If the user select Concept 'Milk Allergy' and Concept 'Dairy Products' if you submit the query then it will generate the rule as 'There are 5 persons affected by 'Milk Allergy' due to 'Dairy Products' or either way we can generate the rule or there are 45 persons with different symptoms are affected. The sample result is shown in Figure 6.

2. Furthermore, the algorithm can be implemented on three concept tables. These generates rules, based on the concept tables and input tables. The HAC generates different rules based on the concept tables with different combinations of concepts. Each concept table represents conceptual hierarchies of categorical attributes. This algorithm specifies the clustering on categorical attributes on the concept tables to derive the association rules in the form of cluster point table.

Implementation of HAC

Implementation of HAC

Disease Name	Food Category	Vote
MILK ALLERGY	DAIRY	5
SOY ALLERGY	NUTS	2
SHELFISH ALLERGY	SEAFOOD	6
WHEAT ALLERGY	GRAINS	6
FOOD ALLERGY	DAIRY	2
PEANUT ALLERGY	NUTS	5
EGG ALLERGY	DAIRY	2
MILK ALLERGY	BACKERY	1
FOOD ALLERGY	NUTS	1
FOOD ALLERGY	FRUITS	3
FOOD ALLERGY	VEGETABLES	2
EGG ALLERGY	BACKERY	3
FOOD ALLERGY	JUNK FOODS	1
FOOD ALLERGY	POULTRY	2
HAY FEVER	POULTRY	1
MILK ALLERGY	JUNK FOODS	1
MILK ALLERGY	GRAINS	1

Miscellaneous

Disease Name	Food Category	Vote
NONALLERGIC RHINITIS		2
CHRONIC SINUSITIS		2
HAY FEVER		1
MOLD ALLERGY		1
ASPERGILLOYSIS		1

Fig. 6. Sample output screen of Cluster Point table

9. Conclusion

In this paper, we studied the hierarchical conceptual clustering applied on structured databases. We have given a new method of HAC with conceptual clustering to explore categorical data. The main contributions of the paper are to develop HAC algorithm which is applied on categorical attributes instead of traditional algorithms which apply the distance metric measures. The results show that the data is categorized using hierarchical conceptual clustering with HAC. There are numerous types of clustering techniques most of these techniques are applicable only on the unstructured data. Sometimes, there is a need to apply clustering on categorical attributes which is not suitable to apply on it. So, the non-metric measures are used to perform clustering on the categorical attributes which represents the closest proximity between the data attributes. HAC is used to represents databases in the form concept tables for categorical data which contains the concepts formed on the domain. This technique can be applied on any of the fields which have structure data. The information is extracted using the HAC algorithm from structured data.

The structured data is applied on input and the results formed are rules extracted from the data. Clustering is important for both types of data. The modern data mining mechanisms are used to apply on the data. From a machine learning standpoint, this research has been greatly influenced by work in conceptual clustering. HAC seeks classifications that maximize a heuristic measure (as in conceptual clustering systems) and uses a search strategy abstracted from incremental systems such as UNIMEM [8].

10. Future Work

Our algorithm can be effective in the medically related areas to allergic disease. Implementing this algorithm with multiple concept tables can be a potential extension of the approach in our case we have implemented only on two concept tables. Another important enhancement of this algorithm can be its implementation on other domains in medicine and also domains apart from medical areas.

11. Acknowledgement

This research has been partially supported by L-3 Communication Corporation, ComCept Division under Project Corvus and TAMU-C research grant #140854-20300.

12. References

- [1] Douglas H. Fisher, "Knowledge Acquisition Via Incremental Conceptual Clustering", University of California, Machine Learning, Volume 2, Issue 2, pp. 139-172, CA, USA, 1987.
- [2] A.K. Jain, M.N. Murty, P.J. Flynn , "Data Clustering: A Review", ACM Computing Surveys (CSUR), Volume 31, Issue 3, pp. 264 - 323 , September 1999
- [3] Jungsoon Park Yoo, Chrisila,C.P., & Sung Yoo. "A Hybrid Conceptual Clustering System", Proceedings of the 1996 ACM 24th Annual Conference on Computer science, pp. 105-114, 1996

- [4] Fei Wu., Georges,G., "Gradual clustering Algorithms", Seventh International Conference on Database Systems for Advanced Applications,2001.
- [5] Sudipto Guha1, Rajeev Rastogi and Kyuseok Shim3, "ROCK: A Robust Clustering Algorithm for Categorical Attributes", IEEE International Conference on Data Engineering, "Information Systems", Volume 25, Number 5, pp. 345 - 366, 2000.
- [6] Sang C.Suh, Sam I. Saffer, Nikil Goel, "Generating Meaningful Rules Using Attribute Concept Hierarchy ", Artificial Neural Networks in Engineering, pp. 1-6, 2006.
- [7]. E. A. Freigenbaum, H. Simon, "EPAM-like models of recognition and learning", Cognitive Science, Volume 8, pp. 305 - 336, 1984.
- [8] M. Lebowitz, "Experiment with incremental concept formation: UNIMEM", Machine Learning, Volume: 2, 1987, pp. 103 -138
- [9] Pyo Jae Kim, Jin Young Choi, "Incremental Conceptual Clustering Using a Modified Category Utility", http://iccl.snu.ac.kr/ilab/paper/dist_file/124ITC_pjkim.pdf
- [10] Carl J. Granberg, Lingi, "Hierarchical clustering of large volumetric datasets", Computer graphics and interactive techniques in Australasia and South East Asia, Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia, pp. 425 - 428,2005
- [11] Heckel B.,Weber G., Hamann B., Joy K, "Construction of vector field hierarchies", Proceedings of Visualization'99, pp. 19-25,1999
- [12] Telea A., van Wijk, J, "Simplified representation of vector fields", Proceedings of Visualization'99, pp.35-42,1999
- [13] www.mayoclinic.com
- [14] <http://www.wrongdiagnosis.com>

Establishing and retrieving domain knowledge from semi-structural corpora

*Hsien-chang WANG, [†]Pei-chin YANG and *Chen-chieh LI

**Chang Jung Christian University, [†]National Cheng Kung University
Taiwan R.O.C.*

1. Introduction

1.1 Knowledge representation

The most essential part of building an expert system is the acquirement and representation of domain knowledge. In the seventies, Feigenbaum indicated the important concept of knowledge engineering. He emphasized that to utilize knowledge in problem-solving process is equally important with knowing how to solve a problem. Knowledge, according to how it is stored, can be classified to tacit knowledge and explicit knowledge. The tacit knowledge, existing in the brain of experts, can only be acquired through interviewing the domain experts. On the other hand, the explicit knowledge can be expressed clearly. Since explicit knowledge is easier to be handled, it was used in most expert systems.

Knowledge representation affects how problems are solved. Human knowledge can be expressed in the form of mathematic formulas, speech, text and figures. In artificial intelligence domain, especially in expert system research, several knowledge representation forms had been proposed (Negnevitsky, 2002). They are:

1. Semantic networks (Quillian, 1965, 1968):
Using directed graph to represent knowledge objects and their relationship.
Each object in the network is linked to other objects by their semantic relationships.
2. Case-based format (Watson, 1997; Kolodner 1993):
Knowledge is stored in the form of cases-solutions.
3. Rule-based format (Triantaphyllou & Felici, 2006):
If-Then rules are stored as the knowledge source.
4. Frame-based format (Minsky, 1975):
Objects are divided into several frames, and each frame contains its corresponding attribute to describe the characteristics of the objects.
5. Ontology (Munn, 2009 ; Uschold & Gruninger, 1996):
It is a representation of some pre-existing domain of reality which reflects the properties of the objects within its domain in such a way that there obtains a systematic correlation between reality and the representation itself. It is formalized in a way that allows it to support automatic information processing.

In this study, we adopted the concept of both ontology and frame-based approach for the knowledge representation.

1.2 Domain Knowledge (Eco-knowledge)

Many researches in 70's revealed that attempting to make a general purpose intelligent system is an unrealistic idea (Newell and Simon, 1972). The inference engine for a general purpose system is hard to build, the knowledge base is also difficult to accumulate and integrate. To make intelligent systems feasible for real applications, it was suggested that one should focus his application on a specific domain. Thus, domain knowledge plays a very important role in the realization of an intelligent system.

Accompany with the raising of eco-consciousness, going outdoor for an eco-tourism becomes a popular activity in these days. During an eco-tourism, people observe many animal and plant species, and will like to know about their names, characteristics, behaviors and further knowledge. To acquire eco-knowledge, people can listen to the explanation of a narrator or consult illustrated handbooks. However, it would be wonderful if we can build an intelligent system which is able to answer queries about specific eco-knowledge.

This goal of building an intelligent eco-knowledge system engenders three problems: (1) it requires large amount of labor to sort out all the domain knowledge; (2) it has to deal with the problem that sentences with different wording maybe describe the same fact; (3) non-expert person may not familiar with those proper nouns used by the narrator and the handbooks.

Thanks to the massive progress of linguistic processing techniques, it is possible to deal with large amount of corpora to extract the most meaningful part for flexible applications. Also, the pattern matching techniques enable the matching and discrimination between different terms more efficiently and correctly. Thus it is very possible to make our goal realize, i.e., to build an intelligent system for ordinary people to inquire eco-knowledge.

In Taiwan, the fact that there are over 500 wild bird species and eco-tourism being more popular makes bird watching a prevalent activity. Thus, this study is aimed to build an intelligent system for wild bird knowledge inquiring. The specific aims are: (1) to define the major key-features of the wild birds; (2) to collect the descriptions of wild birds; (3) to extract the linguistic features of the corpora; (4) to build up structural domain expertise automatically; (5) to define a coding schema for transforming key-features into lexical vectors; (6) to define the membership values of key-features; (7) to evaluate the similarity measurement of different key-features; (8) to illustrate the top-N answers for the input inquires.

2. Materials and Methods

Our research framework consists of four major parts as shown in Figure 1. They are described in detail in the following paragraphs.

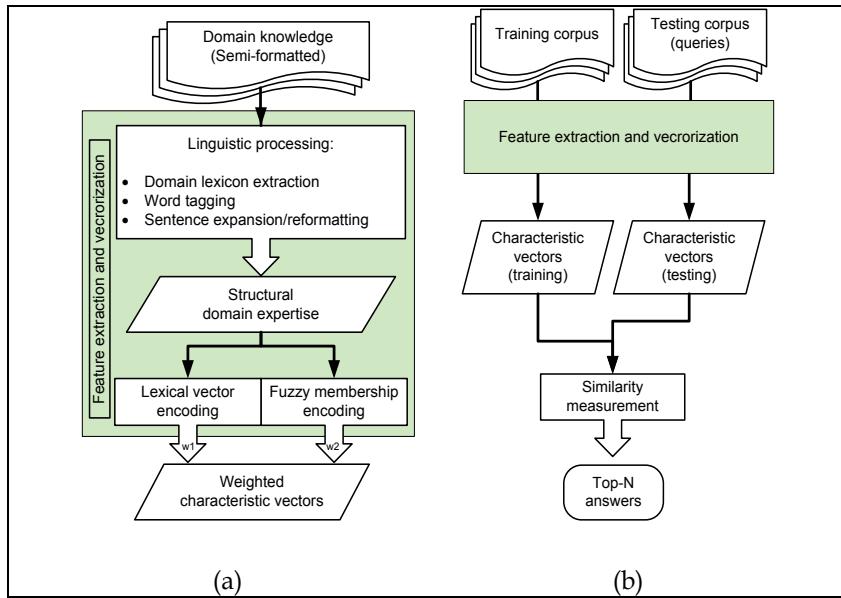


Fig. 1. The research framework for the intelligent wild bird knowledge system. (a). the training phase; (b). the testing phase.

2.1 Semi-structural Corpora

The input corpora are the so-called semi-structural domain knowledge, which contained the descriptions of each of the 442 wild bird species in Taiwan. Here, semi-structural means that the descriptions in the corpora seem follow a certain structural, however, variations often can be observed in such corpora. For example, the following sentences are all describing a fact that the bird has a white tail:

- It has a white tail.
- Its tail is white.
- It has a tail in white.

The above sentences, although in slightly different format, all consist of the part (of bird) and the attributes. Actually, it is the common format for most eco-knowledge descriptions found in the illustrated books. When describing an object (specie), the sentences are represented in the following form:

```

object  -> {[part]}
[part] -> [part name] + {[attributes]}
[part] -> {[attributes]} + [part_name]
[attribute] -> [color] + [texture] + [shape] + [modifier]
[part_name] -> {head, neck, tail, wing, back, beak ...}
[color] -> {black, brick-red, red, brown, dark grey ...}
[modifier] -> {long, shiny, tiny, conspicuous ...}

```

Words in braces may appear repeatedly; words in square brackets are variable terms which can be further decomposed into other components; words without brackets are final

symbols, i.e., those which found in the original descriptions. The next step is to define the final symbols, i.e., the domain lexicon to reduce the complexity of processing.

2.2 Linguistic Processing

As shown in the previous section, the domain lexicon contains five types: [part_name], [color], [texture], [shape] and [modifier]. Since [part_name] contains the domain specific words, it has to be defined first by domain experts. The other four types of lexicon can be derived automatically by applying linguistic processing tools: CKIP AutoTag (CKIP, 2009) and HowNet (Dong & Dong, 2009).

In order to derive the domain lexicon, we apply the auto-tagging program, which is developed by CKIP group, to perform word segmentation and obtain the POS (part-of-speech) tags of each word. The semi-structural corpora are fed to the auto-tagging program, and the resulting POS-tagged words are then processed by HowNet (Dong & Dong, 2009).

HowNet is an on-line common-sense knowledge base unveiling inter-conceptual relations and inter-attribute relations of concepts. For each meaningful word, HowNet provide its semantic attributes. For instance, we can extract words with the attribute "color" easily from HonNet. Thus, by examining all the processed words, we can group those words with attribute "color" together and thus form the [color] domain lexicon. Same procedure can be applied to the [part], [texture] and [modifier] domain lexicons.

The derived domain lexicon may contain words which are too rare or too detailed. It will cause further processing inefficient. Thus, those lexicon need to be refined. For the four types of lexicon, i.e., [part], [color], [texture] and [shape], the refinement processing is described below.

The [part] lexicon originally contain more than 130 words, however, they can be reduced to the most fundamental parts. For example, the words {forehead, upper head, backhead, hair, tophead} are reduced to the fundamental form „head“.

The [color] lexicon contains 152 color words. Based on the theory of basic color (Berlin & Kay, 1969), they are reduced to 11 fundamental colors: {black, grey, white, pink, red, orange, yellow, green, blue, purple, brown}.

The [texture] lexicon is reduced to 16 words: {M-shape, Z-shape, V-shape, fork-shape, T-shape, triangular, mackerel scale, worm hole, round, wave, point, line, thick spot, thin spot, horizontal, vertical}.

The [modifier] lexicon contains those with HowNet attributes „modifier“. Those words are used as emphasized words, such as {striking, shiny, straight, interlaced, ...}

2.3 Vector Encoding

Each sentence in the corpora is transformed into two types of vectors, i.e., the *lexical vector* and the *fuzzy vector*. The lexical vector concerns the lexical part of the described sentences. Lexical vector encoding is simply binary encoding. The elements of the lexical vector are either 0's or 1's. The dimension of lexical vector equals to the number of all reduced lexicon terms. (That's why we reduced the lexicon terms as mentioned above, i.e., to reduce the dimension for faster processing). For each word in the sentence to be encoded, it causes an 1 in the corresponding dimension of the vector.

The fuzzy vector of a sentence consists of the membership value between sentence-words and lexicon-words. The membership values are divided into three types: part, color and texture. We can use three tables to illustrate how fuzzy vector encoding is done.

For fuzzy membership of [part], each detailed-part word is valued by the relationship of how it closes to the fundamental parts. This process is done by averaging several expert's opinions of the membership values. An example membership table is shown below:

		Fundamental parts							
		Head	Back	Tail	Body	Wing	Ear	Beak	...
Detailed parts	Forehead	0.8	0	0	0	0	0	0	...
	Upper beak	0.1	0	0	0	0	0	0.9	...

Table 1. Membership table for detailed parts.

The [color] membership is obtained by the subjective opinions of 10 taggers. For each detailed-color word, the membership value for the eleven fundamental colors are taged and averaged to produce a membership table. An example membership table is shown below:

		Fundamental colors							
		Black	Red	White	Orange	Pink	Grey	Brown	...
Detailed colors	Dark brown	0.1	0.4	0	0	0	0	0.9	...
	Rust	0.2	0.5	0	0	0	0.1	0.6	...

Table 2. Membership table for detailed colors.

The [texture] membership table can be derived by a similar process. Combining all three membership values, a sentence can then be encoded into a fuzzy vector.

2.4 Vector Similarity Measure

In vector space, the similarity of two vectors X and Y can be calculated using five methods (Manning & Schutze, 1999) as shown in the Table 3.

Similarity measure	Definition
Mathcing coefficient	$X \cap Y$
Dice coefficient	$\frac{2 X \cap Y }{ X + Y }$
Jaccard (or Tanimoto) coefficient	$\frac{ X \cap Y }{ X \cup Y }$
Overlap coefficient	$\frac{ X \cap Y }{\min(X , Y)}$
Cosine measure	$\frac{2 X \cap Y }{\sqrt{ X \times Y }}$

Table 3. Definitions of Vector similarity.

While calculating the similarity of two vectors, most approached used the cosine measure of vector intersection angle. However, since it's hard to predict the fuzzy degree of object description made by user, fuzzy encoding vectors should not use the same similarity measure as literal vectors did. In this study, we use Cosine similarity measure for lexical vectors and Overlap coefficient for fuzzy vectors. The final similarity of two descriptions is evaluated according to the measure combining the weight of lexical similarity (S_{Lex}) and fuzzy similarity (S_{Fuz}). The similarity can be evaluated by the following formula:

$$S = \alpha \cdot S_{Lex} + (1 - \alpha) \cdot S_{Fuz} \quad (1)$$

The literal similarity of two vectors can be defined by equation (2).

$$S_{Lex}(X, Y) = \frac{\bar{X} \cdot \bar{Y}}{|\bar{X}| |\bar{Y}|} = \frac{\sum_{i=1}^m X_i Y_i}{\sqrt{\sum_{i=1}^m X_i^2} \times \sqrt{\sum_{i=1}^m Y_i^2}} \quad (2)$$

Where, m is the dimension of the literal vector.

Let S and T be two-dimensional matrix (table) of two sentences. The fuzzy vector similarity can be expressed as equation (3) below.

$$S_{Fuz}(\bar{S}, \bar{T}) = \sum_{i=1}^s \max(\text{olp}(\bar{S}_i, \bar{T})) \quad (3)$$

Where, $\text{olp}(A, B)$ represent the overlap coefficient of vector A, B. The equation for $\text{olp}(A, B)$ is shown below:

$$\text{olp}(\bar{A}, \bar{B}) = \frac{\bar{A} \cap \bar{B}}{\min(|\bar{A}|, |\bar{B}|)} = \frac{|\bar{C}|}{\min(|\bar{A}|, |\bar{B}|)} \quad (4)$$

where, $C = (c_1, c_2, \dots, c_n)$, $c_i = \min(A_i, B_i)$

3. Results and Discussion

The training corpus is a popular illustrated handbook (Wang et al, 1991) with detail descriptions of the features of 442 wild birds in Taiwan. The content is highly recommended by the bird watchers in Taiwan. The structures of the descriptions are very similar, however, the sentences may not grammatically valid due to the need of reducing the page amount. There are totally 6257 sentences in the training corpus.

The testing material varies from three scopes: 1) content of 40 birds from another illustrated handbook (Wu and Hsu, 1995); 2) descriptions of 20 random chosen birds made by a domain expert; 3) naive people's descriptions of 20 randomly chosen birds. The testing handbook contains similar description format as the training one, but was published by different group of people. The expert is a senior birdwatcher with experience of bird watching more than eight years. The naive people had no experience or expertise of wild birds. Figure 2 shows the four types of description for a bird named Black-browed Barbet.

五色鳥 (Black-browed Barbet)		
Corpus source	Description in Chinese	Description in English
A	嘴粗厚，黑色，腳鉛灰色。頭部大致為藍色，額、喉黃色，眉斑雜有黑色羽毛，眼先有紅色斑點，前頸亦有紅斑。後頸、背部鮮綠色，胸以下鮮黃綠色。	Beak is thick, black, foot is lead-grey. Head is almost blue, forehead and throat is yellow, eyebrow contain black feather, red dot in front of eye, fore_neck has red dot too. Back_neck and back is bright green, yellow-green below chest.
B	頭部由鮮豔的紅、黃、青、綠、黑組成，所以才稱為五色鳥。頭部大致為藍色，額、喉黃色，眉斑雜有黑色羽毛，眼先有紅色斑點，前頸亦有紅斑。後頸、背部均為鮮綠色，胸以下鮮黃綠色。	Head consist of five bright colors: red, yellow, blue, green and black, that's why it is named "five-color bird". Head is almost blue, forehead and throat is yellow, eyebrow contain black feather, red dot in front of eye, fore_neck has red dot too. Back_neck and back is bright green, yellow-green below chest.
C	全身綠色。嘴基粗厚，鐵灰色。腳灰綠色。頭藍色，額頭跟喉黃色。眼先有紅點，眉斑黑色。	The whole body is green. Beak-base is thick, iron-grey. Foot is grey-green. Head is blue, forehead and throat is yellow. A red dot before its eye, eyebrow is black.
D	頭有紅色、黃色、藍色、綠色、黑色。頭部藍色，額頭、喉嚨黃色，頸部有紅斑。後頸部、背面綠色。	Its head is red, yellow, blue, green and black. Head is blue, forehead and throat is yellow, neck contain red dot. Back_neck and back is green.

Table 4. Example of descriptions for the Black-browed Barbet. The description comes from the (A) training handbook, (B) testing handbook, (C) expert and (D) naive people, respectively.

The experiments were performed with the weight factor α set to 0, 0.1, 0.3, 0.5, 0.7, 0.9 and 1 respectively. The value of α is equal to zero if we want to ignore the lexical score; and is equal to one if we want to ignore the fuzzy score. For each testing bird, the top-N scores are recorded with N=1, 3, 5, and 10. The experimental results are shown in Figure 3 and Figure 4.

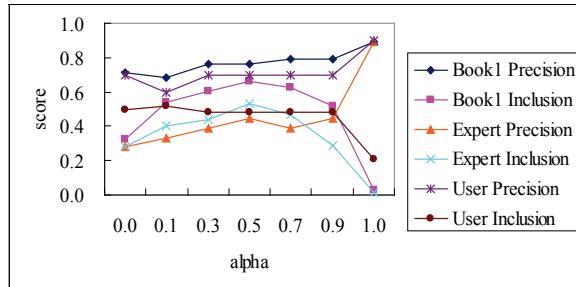


Fig. 3. The precision and inclusion rates for handbook, expert and naive user with alpha ranged from 0 to 1.

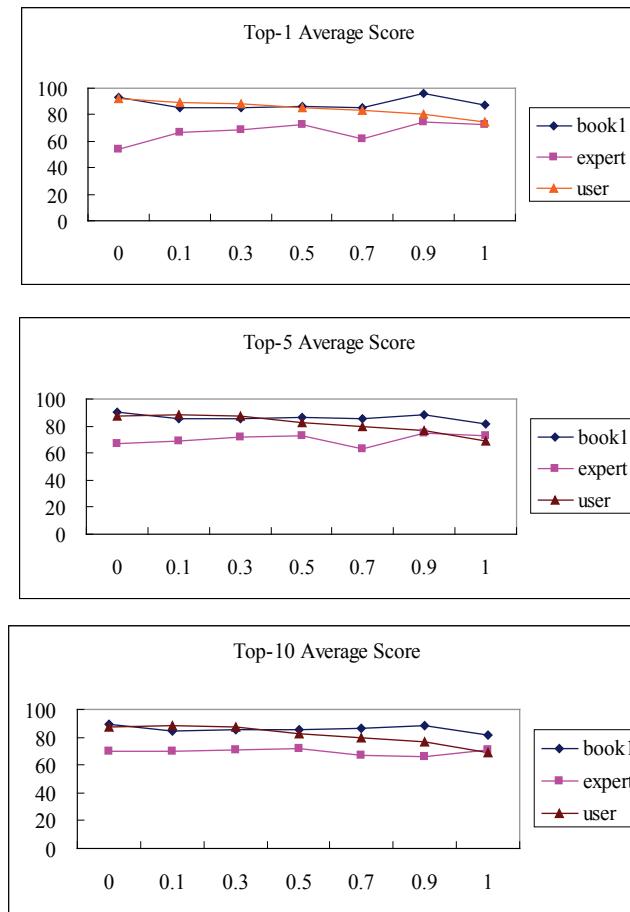


Fig. 4. The averaged score of top-N results for handbook, expert and naive user with alpha ranged from 0 to 1.

The first experiment is to compare the precision and inclusion rate of different testing data. Suppose the number of total testing data set is K, the number of correct answers appeared in the top-N candidates is C. The precision rate is defined as:

$$\text{Precision rate} = \frac{C}{K} \quad (5)$$

Suppose the total number of top-N candidates is T, the inclusion rate is defined as:

$$\text{Inclusion rate} = \frac{C}{T} \quad (6)$$

The precision rate, as defined in usual cases, tells if the correct answer is retrieved. The inclusion rate shows whether redundant answers are also reported while retrieving the answers. Figure 3 shows that, if the weighting (α) of lexical vector closed to 1.0, the precision

rate will be high and, the inclusion rate will be low. This is because redundant answers with the same similarity scores are also retrieved if consider only the lexical scores.

In Figure 4, the average matching score of expert increases as the value of α moving from 0 to 1. This is because that the wording of expert is similar to those in the handbook and thus has higher score while using large α value. (Note that higher α means higher lexical weighting.)

On the other hand, the score of naive is higher when choosing smaller α value. That is, the weighting of fuzzy vector affects the similarity score. This result corresponds with the fact that naive people are not familiar with the domain specific wordings, and introducing the fuzzy vector score has the advantage of compensating the mismatch between their wordings to those in the training corpus.

Fig. 5 and Fig. 6 show the precision rate and inclusion rate of top-10 results for all three types of testing corpora. The notation in these two figures are:

- B2: corporos from another illustrated handbook;
- E: corporos from a domain expert;
- U: corporos from a noive user;
- _P: precesion ratio;
- _I: inclusion ratio;
- _1: use cosine measure for literal and fuzzy similarity;
- _2: use overlap measure for literal and fuzzy similarity;
- _3: use cosine for literal similarity and overlap as fuzzy similarity;

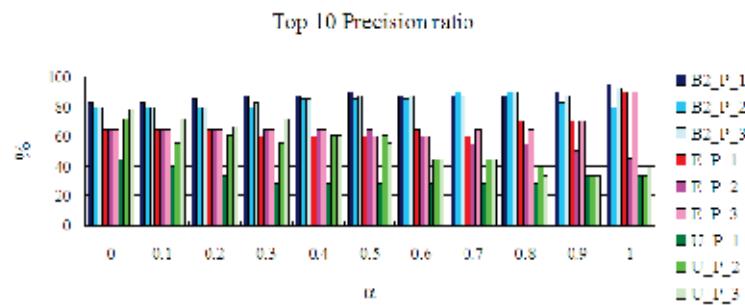


Fig. 5. Top-10 precision ratio for all testing corpora.

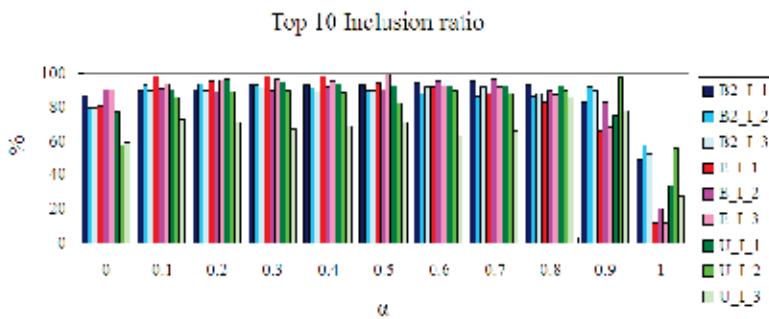


Fig. 6. Top-10 inclusion ratio for all testing corpora.

Since the training corpus is the descriptions of wild bird in an illustrated handbook, corpus B2 (another illustrated book) had the best average precision ratio; corpus E (domain expert) also achieve good result; however, corpus U (naive user) can only got good result when α is closed to 0. The results showed that to allow user query in spontaneous descriptiton, the system should have high weighting in the fuzzy vector instead of literal vector.

4. Conclusion and Future Works

In this study, we proposed an approach to establish and retrieve domain knowledge automatically. The domain knowledge is established by combining the method of linguistic processing and frame-based representation. The features of descriptions consist of two major types: literal vectors and fuzzy vectors. The cosine and overlap measure is chosen to compute the similarity between literal vectors and fuzzy vectors respectively.

According to our study, several results were observed:

1. The proposed approach for domain knowledge processing is useful for establishing and retrieving eco-knowledge.
2. For some birds, its features maybe marked directly on the figures in the book, a few descriptions may be missed in the text data. This will cause some mismatch in the experiment.
3. If an experienced bird watcher wants to use the inquiry system, the literal weighting should be increased. Experiment results showed that the weighting factor could be set as 0.9.
4. For a naive use to user the inquiry system, the literal weighting should be decreased. The weighting factor could be set as 0.2.

For queries made by expert, it seems that only lexical matching is enough. However, for naive people who have no expertise on how to use specialized wording for the description of birds, combining lexical vector score with the fuzzy ones is a good choice.

Since color attributes are essential for discrimination of birds, it plays an important role in the visual cognition of birds. Currently, our study adopted only the eleven basic colors, more sophisticate color membership determination should be considered to obtain better results.

The further interesting research topic will be discovering the commonality and difference between book-style knowledge and knowledge collected from large amount of spontaneous description about objects.

5. References

- Berlin B. & Kay P. (1969). *Basic Color Terms : Their Universality and Evolution*, University of California Press.
- CKIP, (2009), CKIP AutoTag, available at <http://ckipsvr.iis.sinica.edu.tw/>
- Dong Z. & Dong Q. (2009). *HowNet Knowledge Database*, <http://www.keenage.com>.
- Kolodner J. (1993). *Case-Based Reasoning*, Morgan Kaufmann, 978-1558602373.
- Manning C. D. & Schutze H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, 978-0262133609, Cambridge.

- Minsky, M. (1975). A framework for representation knowledge. *The Psychology of Computer Vision*, McGraw-Hill, 978-0070710481, New York.
- Munn K. & Smith B. (2009). *Applied Ontology, An Introduction*, Ontos Verlag Transaction Pub, 978-3938793985.
- Negnevitsky M., (2002). *Artificial Intelligence, A Guide to Intelligent Systems*, Addison-wesley, 978-0321204660, England.
- Newell A. & Simon H.A. (1972). *Human Problem Solving*, Prentice Hall, Englewood Cliffs, 978-0134454030, NJ.
- Quillian M.R. (1965). Word concepts: a theory and simulation of some basic semantic capabilities, *Behavioral Science*, Vol. 12, No. 5., pp. 410-430.
- Quillian M.R. (1968). Semantic Memory, *Semantic Information Processing*, The MIT Press, Ch. 4, pp.227-270, 978-0262130448.
- Triantaphyllou E. & Felici G. (2006). *Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques (Massive Computing)*, Springer, 978-0387342948.
- Wang G.H. et al., (1991). *Taiwan Wild Birds*, Arthur Books, Taipei.
- Watson I. (1997). *Applying Case-Based Reasoning: Techniques for Enterprise Systems*, Morgan Kaufmann, 978-1558604629.
- Wu T.H. & Hsw W.B. (1995), *Guiding Map of Bird Watching in Taiwan*, BigTree Culture, Taipei.
- Uschold, M., Gruninger, M. (1996). *Ontologies: Principles, Methods and Applications*, The Knowledge Engineering Review, 11, 93-136.



Edited by Yagang Zhang

Machine learning techniques have the potential of alleviating the complexity of knowledge acquisition. This book presents today's state and development tendencies of machine learning. It is a multi-author book. Taking into account the large amount of knowledge about machine learning and practice presented in the book, it is divided into three major parts: Introduction, Machine Learning Theory and Applications. Part I focuses on the introduction to machine learning. The author also attempts to promote a new design of thinking machines and development philosophy. Considering the growing complexity and serious difficulties of information processing in machine learning, in Part II of the book, the theoretical foundations of machine learning are considered, and they mainly include self-organizing maps (SOMs), clustering, artificial neural networks, nonlinear control, fuzzy system and knowledge-based system (KBS). Part III contains selected applications of various machine learning approaches, from flight delays, network intrusion, immune system, ship design to CT and RNA target prediction. The book will be of interest to industrial engineers and scientists as well as academics who wish to pursue machine learning. The book is intended for both graduate and postgraduate students in fields such as computer science, cybernetics, system sciences, engineering, statistics, and social sciences, and as a reference for software professionals and practitioners.

Photo by agsandrew / iStock

InTechOpen

ISBN 978-953-51-5899-8



9 789535 158998