Soft Computing for Knowledge Discovery and Data Mining

Soft Computing for Knowledge Discovery and Data Mining

edited by

Oded Maimon
Tel-Aviv University
Israel

and

Lior Rokach
Ben-Gurion University of the Negev
Israel

Springer

Oded Maimon
Tel Aviv University
Dept.of Industrial Engineering
69978 TEL-AVIV
ISRAEL
maimon@eng.tau.ac.il

Lior Rokach
Ben-Gurion University
Dept. of Information System Engineering
84105 BEER-SHEVA
ISRAEL
liorrk@bgu.ac.il

Library of Congress Control Number: 2007934794

Soft Computing for Knowledge Discovery and Data Mining
Edited by Oded Maimon and Lior Rokach

ISBN 978-0-387-69934-9 e-ISBN 978-0-387-69935-6

Printed on acid-free paper.

9 8 7 6 5 4 3 2 1

springer.com

To my family
– O.M.

To my wife Ronit, and my two boys, Yarden and Roy
– L.R.

Preface

The information age has made it easy to store large amounts of data. Data mining is a new and exciting field that tries to solve the crisis of information overload by exploring large and complex bodies of data in order to discover useful patterns. It is extreme importance because it enables modeling and knowledge extraction from abundance data availability. Therefore theoreticians and practitioners are continually seeking techniques to make the process more efficient, cost-effective and accurate. Among the more promising technique that have emerged in recent years are soft computing methods such as fuzzy sets, artificial neural networks, genetic algorithms. These techniques exploit a tolerance for imprecision, uncertainty and partial truth to achieve tractability, robustness and low cost solutions. This book shows that the soft computing methods extend the envelope of problems that data mining can solve efficiently.

This book presents a comprehensive discussion of the state of the art in data mining along with the main soft computing techniques behind it. In addition to presenting a general theory of data mining, the book provides an in-depth examination of core soft computing algorithms.

To help interested researchers and practitioners who are not familiar with the field, the book starts with a gentle introduction to data mining and knowledge discovery in databases (KDD) and prepares the reader for the next chapters. The rest of the book is organized into four parts. The first three parts devoted to the principal constituents of soft computing: neural networks, evolutionary algorithms and fuzzy logic. The last part compiles the recent advances in soft computing and data mining.

This book was written to provide investigators in the fields of information systems, engineering, computer science, statistics and management, with a profound source for the role of soft computing in data mining. In addition, social sciences, psychology, medicine, genetics, and other fields that are interested in solving complicated problems can much benefit from this book. The book can also serve as a reference book for graduate / advanced undergraduate level courses in data mining and machine learning. Practitioners among the readers may be particularly interested in the descriptions of real-world data mining projects performed with soft-computing.

We would like to thank all authors for their valuable contributions. We would like to express our special thanks to Susan Lagerstrom-Fife and Sharon Palleschi of Springer for working closely with us during the production of this book.

Tel-Aviv, Israel *Oded Maimon*
Beer-Sheva, Israel *Lior Rokach*

July 2007

Contents

List of Contributors

Ajith Abraham
Center of Excellence for Quantifiable Quality of Service (Q2S),
Norwegian University of Science and Technology,
Trondheim, Norway
`ajith.abraham@ieee.org`

Arnulfo Azcarraga
College of Computer Studies,
De La Salle University, Manila,
The Philippines
`azcarragaa`
`@canlubang.dlsu.edu.ph`

Ricardo José Gabrielli Barreto Campello
Instituto de Ciências Matemáticas e de Computação,
Universidade de São Paulo
`campello@icmc.usp.br`

André Carlos Ponce de Leon Ferreira de Carvalho
Instituto de Ciê
ncias Matemá
ticas e de Computação
Universidade de São Paulo
`andre@icmc.usp.br`

Jorge Casillas
Dept. of Computer Science and Artificial Intelligence,
University of Granada,
Spain
`casillas@decsai.ugr.es`

Yixin Chen
Dept. of Computer and Information Science
The University of Mississippi
MS 38655
`ychen@cs.olemiss.edu`

Hong Cheng
University of Illinois at Urbana-Champaign
`hcheng3@cs.uiuc.edu`

Swagatam Das
Dept. of Electronics and Telecommunication Engineering,
Jadavpur University,
Kolkata 700032,
India.

Christos Dimou
Electrical and Computer Engineering Dept.
Aristotle University of Thessaloniki,
54 124, Thessaloniki,
Greece
`cdimou@issel.ee.auth.gr`

Alex A. Freitas
Computing Laboratory,
University of Kent,
Canterbury, Kent, CT2 7NF, UK
`A.A.Freitas@kent.ac.uk`

Jiawei Han
University of Illinois at Urbana-Champaign
`hanj@cs.uiuc.edu`

Eduardo Raul Hruschka
`eduardo.hruschka`
`@pesquisador.cnpq.br`

Ming-Huei Hsieh
Dept. of International Business,
National Taiwan University,
Taiwan
`mhhsieh@management.ntu.edu.tw`

Jonathan Lawry
Artificial Intelligence Group,
Department of Engineering Mathematics,
University of Bristol,
BS8 1TR, UK.
`j.lawry@bris.ac.uk`

Ana Carolina Lorena
Centro de Matemática,
Computação e Cognição
Universidade Federal do ABC
Rua Catequese, 242,
Santo André, SP, Brazil
`ana.lorena@ufabc.edu.br`

Oded Maimon
Dept. of Industrial Engineering
Tel-Aviv University
Israel
`maimon@eng.tau.ac.il`

Francisco J. Martínez-López
Dept. of Marketing, University of Granada, Spain
`fjmlopez@ugr.es`

Murilo Coelho Naldi
Instituto de Ciê
ncias Matemá
ticas e de Computação
Universidade de São Paulo
`murilocn@icmc.usp.br`

Shan-Ling Pan
School of Computing,
National University of Singapore,
Singapore
`pansl@comp.nus.edu.sg`

Gisele L. Pappa
Computing Laboratory
University of Kent
Canterbury, Kent, CT2 7NF, UK
`glp6@kent.ac.uk`

Huy Nguyen Anh Pham
Dept. of Computer Science,
298 Coates Hall,
Louisiana State University,
Baton Rouge, LA 70803
`hpham15@lsu.edu`

Zengchang Qin
Berkeley Initiative in Soft Computing (BISC),
Computer Science Division,
EECS Department,
University of California,
Berkeley, CA 94720, US.

zqin@eecs.berkeley.edu

Lior Rokach
Dept. of Information System Engineering,
Ben-Gurion University,
Israel
liorrk@bgu.ac.il

Sandip Roy
Dept. of Computer Science and Engineering,
Asansol Engineering College,
Asansol-713304, India.

Alon Schclar
School of Computer Science,
Tel Aviv University,
Tel Aviv 69978,
Israel
shekler@post.tau.ac.il

Rudy Setiono
School of Computing,
National University of Singapore,
Singapore
rudys@comp.nus.edu.sg

Andreas L. Symeonidis
Electrical and Computer Engineering Dept.
Aristotle University of Thessaloniki,
54 124, Thessaloniki,
Greece
asymeon@iti.gr

Pericles A. Mitkas
Electrical and Computer Engineering Dept.
Aristotle University of Thessaloniki,
54 124, Thessaloniki,
Greece
mitkas@eng.auth.gr

Evangelos Triantaphyllou
Dept. of Computer Science,
298 Coates Hall,
Louisiana State University,
Baton Rouge, LA 70803
trianta@lsu.edu

Philip S. Yu
IBM T. J. Watson Research Center
psyu@us.ibm.com

G. Peter Zhang
Georgia State University,
Dept. of Managerial Sciences
gpzhang@gsu.edu

Introduction to Soft Computing for Knowledge Discovery and Data Mining

Oded Maimon[1] and Lior Rokach[2]

[1] Department of Industrial Engineering, Tel-Aviv University, Ramat-Aviv 69978, Israel,
maimon@eng.tau.ac.il

[2] Department of Information System Engineering, Ben-Gurion University, Beer-Sheba, Israel,
liorrk@bgu.ac.il

Summary. In this chapter we introduce the Soft Computing areas for Data Mining and the Knowledge Discovery Process, discuss the need for plurality of methods, and present the book organization and abstracts.

1 Introduction

Data Mining is the science, art and technology of exploring data in order to discover insightful unknown patterns. It is a part of the overall process of Knowledge Discovery in Databases (KDD). The accessibility and abundance of information today makes data mining a matter of considerable importance and necessity.

Soft computing is a collection of new techniques in artificial intelligence, which exploit the tolerance for imprecision, uncertainty and partial truth to achieve tractability, robustness and low solution cost. Given the history and recent growth of the field, it is not surprising that several mature soft computing methods are now available to the practitioner, including: fuzzy logic, artificial neural networks, genetic algorithms, and swarm intelligence. The aims of this book are to present and explain the important role of soft computing methods in data mining and knowledge discovery.

The unique contributions of this book is in the introduction of soft computing as a viable approach for data mining theory and practice, the detailed descriptions of novel soft-computing approaches in data mining, and the illustrations of various applications solved in soft computing techniques, including: Manufacturing, Medical, Banking, Insurance, Business Intelligence and others. The book does not include some of the most standard techniques in Data Mining, such as Decision Trees (the reader is welcome to our new book, from 2007, dedicated entirely to Decision Trees). The book include the leading soft computing methods, though for volume reasons it could not cover all methods, and there are further emerging techniques, such as fractal based data mining (a topic of our current research).

Since the information age, the accumulation of data has become easier and storing it inexpensive. It has been estimated that the amount of stored information doubles less than twenty months. Unfortunately, as the amount of electronically stored information increases, the ability to understand and make use of it does not keep pace with its growth. Data Mining is a term coined to describe the process of sifting through large databases for interesting patterns and relationships. The studies today aim at evidence-based modeling and analysis, as is the leading practice in medicine, finance, intelligence and many other fields. Evidently, in the presence of the vast techniques' repertoire and the complexity and diversity of the explored domains, one real challenge today in the data mining field is to know how to utilize this repertoire in order to achieve the best results. The book shows that the soft computing methods extend the envelope of problems that data mining can solve efficiently. The techniques of soft computing are important for researchers in the fields of data mining, machine learning, databases and information systems, engineering, computer science and statistics.

This book was written to provide investigators in the fields of information systems, engineering, computer science, statistics and management, with a profound source for the role of soft computing in data mining. In addition, social sciences, psychology, medicine, genetics, and other fields that are interested in solving complicated problems can much benefit from this book. Practitioners among the readers may be particularly interested in the descriptions of real-world data mining projects performed with soft computing.

The material of this book has been taught by the authors in graduate and undergraduate courses at Tel-Aviv University and Ben-Gurion University. The book can also serve as a reference book for graduate and advanced undergraduate level courses in data mining and machine learning.

In this introductory chapter we briefly present the framework and overall knowledge discovery process in the next two sections, and then the logic and organization of this book, with brief description of each chapter.

2 The Knowledge Discovery process

This book is about methods, which are the core of the Knowledge Discovery process. For completion we briefly present here the process steps. The knowledge discovery process is iterative and interactive, consisting of nine steps.

Note that the process is iterative at each step, meaning that moving back to previous steps may be required. The process has many "artistic" aspects in the sense that one cannot present one formula or make a complete taxonomy for the right choices for each step and application type. Thus it is required to understand the process and the different needs and possibilities in each step.

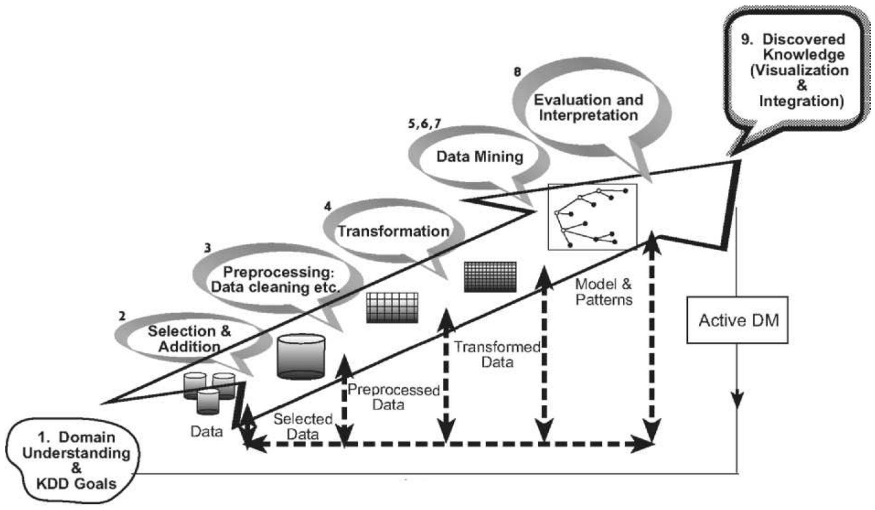

9. Discovered Knowledge (Visualization & Integration)
8
Evaluation and Interpretation
5,6,7
Data Mining
4
Transformation
3
Preprocessing: Data cleaning etc.
2
Selection & Addition
Model & Patterns
Active DM
Transformed Data
Preprocessed Data
Selected Data
Data
1. Domain Understanding & KDD Goals

Fig. 1. The Process of Knowledge Discovery in Databases.

The process starts with determining the KDD goals, and "ends" with the implementation of the discovered knowledge. Then the loop is closed - the Active Data Mining part starts. As a result, changes can be made in the application domain (such as offering different features to mobile phone users in order to reduce churning). This closes the loop, and the effects are then measured on the new data repositories, and the KDD process is launched again.

Following is a brief description of the nine-step KDD process, starting with a managerial step:

1. Developing an understanding of the application domain: This is the initial preparatory step. It prepares the scene for understanding what should be done with the many decisions (about transformations, algorithms, representation, etc.). The people who are in charge of a KDD project need to understand and define the goals of the end-user and the environment in which the knowledge discovery process will take place (including relevant prior knowledge). As the KDD process proceeds, there may be even a revision of this step.
   Having understood the KDD goals, the preprocessing of the data starts, defined in the next three steps.
2. Selecting and creating a data set on which discovery will be performed: Having defined the goals, the data that will be used for the knowledge discovery should be determined. This includes finding out what data is available, obtaining additional necessary data, and then integrating all the data for the knowledge discovery into one data set, including the attributes

that will be considered for the process. This process is very important because the Data Mining learns and discovers from the available data. This is the evidence base for constructing the models.

3. Preprocessing and cleansing: In this stage, data reliability is enhanced. It includes data clearing, such as handling missing values and removal of noise or outliers. There are many methods explained in the handbook, from doing almost nothing to becoming the major part (in terms of time consumed) of a KDD project in certain projects. It may involve complex statistical methods or using a Data Mining algorithm in this context. For example, if one suspects that a certain attribute is of insufficient reliability or has many missing data, then this attribute could become the goal of a data mining supervised algorithm, or finding the centroids of clustering. A prediction model for this attribute will be developed, and then missing data can be predicted. The extension to which one pays attention to this level depends on many factors. In any case, studying the aspects is important and often revealing by itself, regarding complex information systems.
4. Data transformation: In this stage, the generation of better data for the data mining is prepared and developed. Methods here include dimension reduction (such as feature selection and record sampling), and attribute transformation (such as discretization of numerical attributes and functional transformation). This step can be crucial for the success of the entire KDD project, and it is usually very project-specific. For example, in medical examinations, the quotient of attributes may often be the most important factor, and not each one by itself. In marketing, we may need to consider effects beyond our control as well as efforts and temporal issues (such as studying the effect of advertising accumulation). However, even if we do not use the right transformation at the beginning, we may obtain a surprising effect that hints to us about the transformation needed (in the next iteration). Thus the KDD process reflects upon itself and leads to an understanding of the transformation needed.
   Having completed the above four steps, the following four steps are related to the Data Mining part, where the focus is on the algorithmic aspects employed for each project:
5. Choosing the appropriate Data Mining task: We are now ready to decide on which type and approach of Data Mining to use, for example, classification, regression, or clustering. This mostly depends on the KDD goals, and also on the previous steps. There are two major goals in Data Mining: prediction and description. Prediction is often referred to as supervised Data Mining, while descriptive Data Mining includes the unsupervised and visualization aspects of Data Mining. Most Data Mining techniques are based on inductive learning, where a model is constructed explicitly or implicitly by generalizing from a sufficient number of training examples. The underlying assumption of the inductive approach is that the trained

model is applicable to future cases. The strategy also takes into account the level of meta-learning for the particular set of available data.

6. Choosing the Data Mining algorithm: Having the strategy, we now decide on the tactics. This stage includes selecting the specific method to be used for searching patterns (including multiple inducers). For example, in considering precision versus understandability, the former is better with neural networks, while the latter is better with decision trees. For each strategy of meta-learning there are several possibilities of how it can be accomplished. Meta-learning focuses on explaining what causes a Data Mining algorithm to be successful or not in a particular problem. Thus, this approach attempts to understand the conditions under which a Data Mining algorithm is most appropriate. Each algorithm has parameters and tactics of learning (such as ten-fold cross-validation or another division for training and testing).
7. Employing the Data Mining algorithm: Finally the implementation of the Data Mining algorithm is reached. In this step we might need to employ the algorithm several times until a satisfied result is obtained, for instance by tuning the algorithm's control parameters, such as the minimum number of instances in a single leaf of a decision tree.
8. Evaluation: In this stage we evaluate and interpret the mined patterns (rules, reliability, etc.), with respect to the goals defined in the first step. Here we consider the preprocessing steps with respect to their effect on the Data Mining algorithm results (for example, adding features in Step 4 and repeating from there). This step focuses on the comprehensibility and usefulness of the induced model. In this step the discovered knowledge is also documented for further usage.

   The last step is the usage and overall feedback on the patterns and discovery results obtained by the Data Mining:
9. Using the discovered knowledge: We are now ready to incorporate the knowledge into another system for further action. The knowledge becomes active in the sense that we may make changes to the system and measure the effects. Actually the success of this step determines the effectiveness of the entire KDD process. There are many challenges in this step, such as loosing the "laboratory conditions" under which we have operated. For instance, the knowledge was discovered from a certain static snapshot (usually sample) of the data, but now the data becomes dynamic. Data structures may change (certain attributes become unavailable), and the data domain may be modified (such as, an attribute may have a value that was not assumed before).

3 The need for plurality of methods

Data Mining methods are becoming part of general purpose Integrated Information Technology (IIT) software packages. Starting from the data sources (such as operational databases, semi- and non-structured data and reports, Internet sites etc.), then the tier of the data warehouse, followed by OLAP (On Line Analytical Processing) servers and concluding with analysis tools, where Data Mining tools are the most advanced.

We can naively distinguish among three levels of analysis. The simplest one is achieved by report generators (for example, presenting all claims that occurred because of a certain cause last year, such as car theft). We then proceed to OLAP multi-level analysis (for example presenting the ten towns where there was the highest increase of vehicle theft in the last month as compared to with the month before). Finally a complex analysis is carried out for discovering the patterns that predict car thefts in these cities, and what might occur if anti theft devices were installed. The latter is based on modeling of the phenomena, where the first two levels are ways of data aggregation and fast manipulation.

Empirical comparison of the performance of different approaches and their variants in a wide range of application domains has shown that each performs best in some, but not all, domains. This phenomenon is known as the selective superiority problem, which means, in our case, that no induction approach or algorithm can be the best in all possible domains. The reason is that each algorithm contains an explicit or implicit bias that leads it to prefer certain generalizations over others, and it will be successful only as long as this bias matches the characteristics of the application domain.

Results have demonstrated the existence and correctness of this "no free lunch theorem". If one inducer is better than another in some domains, then there are necessarily other domains in which this relationship is reversed. This implies in KDD that for a given problem a certain approach can yield more knowledge from the same data than other approaches.

In many application domains, the generalization error (on the overall domain, not just the one spanned in the given data set) of even the best methods is far above the training set, and the question of whether it can be improved, and if so how, is an open and important one. Part of the answer to this question is to determine the minimum error achievable by any classifier in the application domain (known as the optimal Bayes error). If existing classifiers do not reach this level, new approaches are needed. Although this problem has received considerable attention, no generally reliable method has so far been demonstrated. This is one of the challenges of the DM research – not only to solve it, but even to quantify and understand it better. Heuristic methods can then be compared absolutely and not just against each other.

A subset of this generalized study is the question of which approach and inducer to use for a given problem. To be even more specific, the performance measure need to be defined appropriately for each problem. Though there are some commonly accepted measures it is not enough. For example, if the analyst is looking for accuracy only, one solution is to try each one in turn, and by estimating the generalization error, to choose the one that appears to perform best. Another approach, known as multi-strategy learning, attempts to combine two or more different paradigms in a single algorithm.

The dilemma of what method to choose becomes even greater if other factors such as comprehensibility are taken into consideration. For instance, for a specific domain, neural networks may outperform decision trees in accuracy. However from the comprehensibility aspect, decision trees are considered superior. In other words, in this case even if the researcher knows that neural network is more accurate, the dilemma of what methods to use still exists (or maybe to combine methods for their separate strength).

Induction is one of the central problems in many disciplines such as machine learning, pattern recognition, and statistics. However the feature that distinguishes Data Mining from traditional methods is its scalability to very large sets of varied types of input data. Scalability means working in an environment of high number of records, high dimensionality, and a high number of classes or heterogeneousness. Nevertheless, trying to discover knowledge in real life and large databases introduces time and memory problems.

As large databases have become the norms in many fields (including astronomy, molecular biology, finance, marketing, health care, and many others), the use of Data Mining to discover patterns in them has become potentially very beneficial for the enterprise. Many companies are staking a large part of their future on these "Data Mining" applications, and turn to the research community for solutions to the fundamental problems they encounter.

While a very large amount of available data used to be the dream of any data analyst, nowadays the synonym for "very large" has become "terabyte" or "pentabyte", a barely imaginable volume of information. Information-intensive organizations (like telecom companies and financial institutions) are expected to accumulate pentabyte of raw data every one to two years.

High dimensionality of the input (that is, the number of attributes) increases the size of the search space in an exponential manner (known as the "Curse of Dimensionality"), and thus increases the chance that the inducer will find spurious classifiers that in general are not valid. There are several approaches for dealing with a high number of records including: sampling methods, aggregation, massively parallel processing, and efficient storage methods. This book presents some of the approaches in this direction.

4 The organization of the book

The book has sixteen chapters divided into four main parts, where the first three address the methods and topics that are most identified with soft computing, and then the last part adds advanced and promising methods and areas:

I. Neural network methods: Chapters 2 to 3
II. Evolutionary methods: Chapters 4 to 7

III. Fuzzy logic methods: Chapters 8 to 11
IV. Advanced soft computing methods and areas: Chapters 12 to 16 Including: Swarm intelligence (12), diffusion process (13), and agent technology (14); and the areas of: approximate frequent item-set mining (15), and finally the impact of over-fitting and over-generalization on the classification accuracy in Data Mining (16).

In the following, edited abstracts of the chapters in the book are presented, for the reader map and convenience:

4.1 Neural network methods

The first methodology addressed in the book is **Neural Networks**, which have become elaborated important tools for data mining. Chapter 2 provides an overview of neural network models and their applications to data mining tasks. It also provides historical development of the field of neural networks and present three important classes of neural models including feed forward multilayer networks, Hopfield networks, and Kohonen's self-organizing maps. Modeling issues and applications of these models for data mining are discussed as well.

Then Chapter 3 continues in this direction by specifically addressing **Self-Organizing Maps** (SOMs). SOMs have been useful in gaining insights about the information content of large volumes of data in various data mining applications. As a special form of neural networks, they have been attractive as a data mining tool because they are able to extract information from data even with very little user-intervention (though some is needed). This chapter proposes a methodical and semi-automatic SOM labeling procedure that does not require a set of labeled patterns, and shows an effective alternative. The effectiveness of the method is demonstrated on a data mining application involving customer-profiling based on an international market segmentation study.

4.2 Evolutionary methods

A new family of methods starts in Chapter 4 with a review of **Evolutionary Algorithms** (EAs) for Data Mining. Evolutionary Algorithms are stochastic search algorithms inspired by the process of neo-Darwinian evolution. The motivation for applying EAs to data mining is that they are robust, adaptive search techniques that perform a global search in the solution space. This chapter first presents a brief overview of EAs, focusing mainly on two kinds of EAs, viz. Genetic Algorithms (GAs) and Genetic Programming (GP). Then the chapter reviews the main concepts and principles used by EAs designed for solving several data mining tasks, namely: discovery of classification rules, clustering, attribute selection and attribute construction. Finally, it discusses

Multi-Objective EAs, based on the concept of Pareto dominance, and their use in several data mining tasks.

Then Chapter 5 continues this topic by specifically addressing **Genetic Clustering** for Data Mining. Genetic Algorithms (GAs) have been successfully applied to several complex data analysis problems in a wide range of domains, such as image processing, bioinformatics, and crude oil analysis. The need for organizing data into categories of similar objects has made the task of clustering increasingly important to those domains. This chapter presents a survey of the use of GAs for clustering applications. A variety of encoding (chromosome representation) approaches, fitness functions, and genetic operators are described, all of them customized to solve problems in such an application context.

Chapter 6 addresses the discovering of new rule by induction algorithms with **Grammar-Based Genetic Programming**. Rule induction is a data mining technique used to extract classification rules of the form IF (conditions) THEN (predicted class) from data. The majority of the rule induction algorithms found in the literature follow the sequential covering strategy, which essentially induces one rule at a time until (almost) all the training data is covered by the induced rule set. This strategy describes a basic algorithm composed by several key elements, which can be modified to generate new and better rule induction algorithms. With this in mind, this work proposes the use of a **Grammar-based Genetic Programming** (GGP) algorithm to automatically discover new sequential covering algorithms. The proposed system is evaluated using 20 data sets, and the automatically-discovered rule induction algorithms are compared with four well-known human-designed rule induction algorithms. Results showed that the GGP system is a promising approach to effectively discover new sequential covering algorithms

Another general aspect of data mining issues is introduced in Chapter 7 with **Evolutionary Design** of code-matrices for multi-class problems. Given a dataset containing data whose classes are known, Machine Learning algorithms can be employed for the induction of a classifier able to predict the class of new data from the same domain, performing the desired discrimination. Several machine learning techniques are originally conceived for the solution of problems with only two classes. In multi-class applications, an alternative frequently employed is to divide the original problem into binary subtasks, whose results are then combined. The decomposition can be generally represented by a code-matrix, where each row corresponds to a codeword assigned for one class and the columns represent the binary classifiers employed. This chapter presents a survey on techniques for multi-class problems code-matrix design. It also shows how evolutionary techniques can be employed to solve this problem.

4.3 Fuzzy logic methods

The role of **Fuzzy Sets** in Data Mining is introduced in Chapter 8. This chapter discusses how fuzzy logic extends the envelop of the main data mining tasks: clustering, classification, regression and association rules. The chapter begins by presenting a formulation of the data mining using fuzzy logic attributes. Then, for each task, the chapter provides a survey of the main algorithms and a detailed description (i.e. pseudo-code) of the most popular algorithms.

Continuing with the same area Chapter 9 addresses **Support Vector Machines and Fuzzy Systems.** Fuzzy set theory and fuzzy logic provide tools for handling uncertainties in data mining tasks. To design a fuzzy rule-based classification system (fuzzy classifier) with good generalization ability in a high dimensional feature space has been an active research topic for a long time. As a powerful machine learning approach for data mining and pattern recognition problems, support vector machine (SVM) is known to have good generalization ability. More importantly, an SVM can work very well on a high (or even infinite) dimensional feature space. This chapter presents a survey of the connection between fuzzy classifiers and kernel machines.

KDD in Marketing with **Genetic Fuzzy Systems** is addressed in Chapter 10. This chapter presents a new methodology to marketing (causal) modeling. Specifically it is applied to a consumer behavior model used for the experimentation. The characteristics of the problem (with uncertain data and available knowledge from a marketing expert) and the multi objective optimization make genetic fuzzy systems a good tool for this problem type. By applying this methodology useful information patterns (fuzzy rules) are obtained, which help to better understand the relations among the elements of the marketing system being analyzed (consumer model in this case).

In Chapter 11 the fuzzy theme is continued with a **Framework for Modeling with Words.** The learning of transparent models is an important and neglected area of data mining. The data mining community has tended to focus on algorithm accuracy with little emphasis on the knowledge representation framework. However, the transparency of a model will help practitioners greatly in understanding the trends and idea hidden behind the system. In this chapter a random set based knowledge representation framework for learning linguistic models is introduced. This framework is referred to as label semantics and a number of data mining algorithms are proposed. In this framework, a vague concept is modeled by a probability distribution over a set of appropriate fuzzy labels, which is called as mass assignment. The idea of mass assignment provides a probabilistic approach for modeling uncertainty based on pre-defined fuzzy labels.

4.4 Advanced soft computing methods and areas

A new soft computing methodology is introduced in Chapter 12, which addresses **Swarm Intelligence** algorithms for data clustering. Data mining tasks require fast and accurate partitioning of huge datasets, which may come with a variety of attributes or features. This, in turn, imposes severe computational requirements on the relevant clustering techniques. A family of bio-inspired algorithms, well-known as Swarm Intelligence (SI) has recently emerged that meets these requirements and has successfully been applied to a number of real world clustering problems. This chapter explores the role of SI in clustering different kinds of datasets. It finally describes a new SI technique for partitioning any dataset into an optimal number of groups through one run of optimization. Computer simulations undertaken in this research have also been provided to demonstrate the effectiveness of the proposed algorithm.

In Chapter 13 another type of method for soft computing is revealed, namely **Diffusion method**. This chapter describes a natural framework based on diffusion processes for the multi-scale analysis of high-dimensional data-sets. Many fields of research deal with high-dimensional data sets. Hyper spectral images in remote sensing and in hyper-spectral microscopy, transactions in banking monitoring systems are just a few examples for this type of sets. Revealing the geometric structure of these data-sets is a preliminary step to facilitate their efficient processing. Often, only a small number of parameters govern the structure of the data-set. This number is the true dimension of the data-set and is the motivation to reduce the dimensionality of the set. Dimensionality reduction algorithms try to discover the true dimension of a data set. The diffusion process scheme enables the description of the geometric structures of such sets by utilizing the Newtonian paradigm according to which a global description of a system can be derived by the aggregation of local transitions. Specifically, a Markov process is used to describe a random walk on the data set. The spectral properties of the Markov matrix that is associated with this process are used to embed the data-set in a low-dimensional space. This scheme also facilitates the parameterization of a data-set when the high dimensional data-set is not accessible and only a pair-wise similarity matrix is at hand.

Agent Technology as applied to Data Mining is introduced in Chapter 14. Today's applications are required to extract knowledge from large, often distributed, repositories of text, multimedia or hybrid content. The nature of this quest makes it impossible to use traditional deterministic computing techniques. Instead, various soft computing techniques are employed to meet the challenge for more sophisticated solutions in knowledge discovery. Most notably, Data Mining (DM) is thought of as one of the state-of-the-art paradigms. DM produces useful patterns and associations from large data repositories that can later be used as *knowledge nuggets*, within the context of any application. Individual facets of knowledge discovery, introduced by DM techniques, often need to be orchestrated, integrated and presented to end users in a unified way. Moreover, knowledge has to be exploited and embodied in autonomous software for learning purposes and, hence, a more increased performance. Agent Technology (AT) proves to be a promising paradigm that is suitable for modeling and implementing the unification of DM tasks, as well as for providing autonomous entity models that dynamically incorporate and use existing knowledge. Indeed, a plethora of multi-agent systems (MAS) and other agent-related solutions for knowledge-based systems can be found in the literature, and more specifically in the area of agent-based DM, as it is explained in detail in this chapter.

The issue of error-tolerant item-set is presented in Chapter 15, which addresses **Approximate Frequent Item-set Mining** in the presence of random noise. Frequent item-set mining has been a focused theme in data mining research and an important first step in the analysis of data arising in a broad range of applications. The traditional exact model for frequent item-set requires that every item occur in each supporting transaction. However, real application data is usually subject to random noise or measurement error, which poses new challenges for the efficient discovery of frequent item-set from the noisy data. Mining approximate frequent item-set in the presence of noise involves two key issues: the definition of a noise-tolerant mining model and the design of an efficient mining algorithm. This chapter gives an overview of the approximate item-set mining algorithms in the presence of random noise and examines several noise-tolerant mining approaches.

The impact of over fitting and over generalization on the classification accuracy in Data Mining is addressed in, Chapter 16, the last chapter of the book. Many classification studies often times conclude with a summary table, which presents performance results of applying various data mining approaches on different datasets. No single method outperforms all methods all the time. Further-more, the performance of a classification method in terms of its false-positive and false-negative rates may be totally unpredictable. Attempts to minimize any of the previous two rates, may lead to an increase on the other rate. If the model allows for new data to be deemed as unclassifiable when there is not adequate information to classify them, then it is possible for the previous two error rates to be very low. However, at the same time, the rate of having unclassifiable new examples may be very high. The root to the above critical problem is the over fitting and overgeneralization behaviors of a given classification approach when it is processing a particular dataset.

Although the above situation is of fundamental importance to data mining, it has not been studied from a comprehensive point of view. Thus, this chapter analyzes the above issues in depth. It also proposes a new approach called the Homogeneity-Based Algorithm (or HBA) for optimally controlling the previous three error rates. This is done by first formulating an optimization problem. The key development in this chapter is based on a special way for analyzing the space of the training data and then partitioning it according to the data density of different regions of this space. Next, the classification task is pursued based on the previous partitioning of the training space. In this way, the previous three error rates can be controlled in a comprehensive manner. Some preliminary computational results seem to indicate that the proposed approach has a significant potential to fill in a critical gap in current data mining methodologies.

.

Neural Networks For Data Mining

G. Peter Zhang

Georgia State University,
Department of Managerial Sciences,
gpzhang@gsu.edu

Summary. Neural networks have become standard and important tools for data mining. This chapter provides an overview of neural network models and their applications to data mining tasks. We provide historical development of the field of neural networks and present three important classes of neural models including feedforward multilayer networks, Hopfield networks, and Kohonen's self-organizing maps. Modeling issues and applications of these models for data mining are discussed.

Key words: neural networks, regression, classification, prediction, clustering

1 Introduction

Neural networks or artificial neural networks are an important class of tools for quantitative modeling. They have enjoyed considerable popularity among researchers and practitioners over the last 20 years and have been successfully applied to solve a variety of problems in almost all areas of business, industry, and science (Widrow, Rumelhart & Lehr, 1994). Today, neural networks are treated as a standard data mining tool and used for many data mining tasks such as pattern classification, time series analysis, prediction, and clustering. In fact, most commercial data mining software packages include neural networks as a core module.

Neural networks are computing models for information processing and are particularly useful for identifying the fundamental relationship among a set of variables or patterns in the data. They grew out of research in artificial intelligence; specifically, attempts to mimic the learning of the biological neural networks especially those in human brain which may contain more than 10^{11} highly interconnected neurons. Although the *artificial* neural networks discussed in this chapter are extremely simple abstractions of biological systems and are very limited in size, ability, and power comparing biological neural networks, they do share two very important characteristics: 1) parallel processing of information and 2) learning and generalizing from experience.

The popularity of neural networks is due to their powerful modeling capability for pattern recognition. Several important characteristics of neural networks make them suitable and valuable for data mining. First, as opposed to the traditional model-based methods, neural networks do not require several unrealistic *a priori* assumptions about the underlying data generating process and specific model structures. Rather, the modeling process is highly adaptive and the model is largely determined by the characteristics or patterns the network learned from data in the learning process. This data-driven approach is ideal for real world data mining problems where data are plentiful but the meaningful patterns or underlying data structure are yet to be discovered and impossible to be pre-specified.

Second, the mathematical property of the neural network in accurately approximating or representing various complex relationships has been well established and supported by theoretic work (Chen and Chen, 1995; Cybenko, 1989; Hornik, Stinchcombe, and White 1989). This universal approximation capability is powerful because it suggests that neural networks are more general and flexible in modeling the underlying data generating process than traditional fixed-form modeling approaches. As many data mining tasks such as pattern recognition, classification, and forecasting can be treated as function mapping or approximation problems, accurate identification of the underlying function is undoubtedly critical for uncovering the hidden relationships in the data.

Third, neural networks are nonlinear models. As real world data or relationships are inherently nonlinear, traditional linear tools may suffer from significant biases in data mining. Neural networks with their nonlinear and nonparametric nature are more cable for modeling complex data mining problems.

Finally, neural networks are able to solve problems that have imprecise patterns or data containing incomplete and noisy information with a large number of variables. This fault tolerance feature is appealing to data mining problems because real data are usually dirty and do not follow clear probability structures that typically required by statistical models.

This chapter aims to provide readers an overview of neural networks used for data mining tasks. First, we provide a short review of major historical developments in neural networks. Then several important neural network models are introduced and their applications to data mining problems are discussed.

2 A Brief History

Historically, the field of neural networks is benefited by many researchers in diverse areas such as biology, cognitive science, computer science, mathematics, neuroscience, physics, and psychology. The advancement of the filed, however, is not evolved steadily, but rather through periods of dramatic progress and enthusiasm and periods of skepticism and little progress.

The work of McCulloch and Pitts (1943) is the basis of modern view of neural networks and is often treated as the origin of neural network field. Their research is the first attempt to use mathematical model to describe how a neuron works. The main feature of their neuron model is that a weighted sum of input signals is compared to a threshold to determine the neuron output. They showed that simple neural networks can compute any arithmetic or logical function.

In 1949, Hebb (1949) published his book "The Organization of Behavior." The main premise of this book is that behavior can be explained by the action of neurons. He proposed one of the first learning laws that postulated a mechanism for learning in biological neurons.

In the 1950s, Rosenblatt and other researchers developed a class of neural networks called the perceptrons which are models of a biological neuron. The perceptron and its associated learning rule (Rosenblatt, 1958) had generated a great deal of interest in neural network research. At about the same time, Widrow and Hoff (1960) developed a new learning algorithm and applied it to their ADALINE (Adaptive Linear Neuron) networks which is very similar to perceptrons but with linear transfer function, instead of hard-limiting function typically used in perceptrons. The Widrow-Hoff learning rule is the basis of today's popular neural network learning methods. Although both perceptrons and ADALINE networks have achieved only limited success in pattern classification because they can only solve linearly-separable problems, they are still treated as important work in neural networks and an understanding of them provides the basis for understanding more complex networks.

The neural network research was hit by the book "Perceptrons" by Minsky and Papert (1969) who pointed out the limitation of the perceptrons and other related networks in solving a large class of nonlinearly separable problems. In addition, although Minsky and Papert proposed multilayer networks with hidden units to overcome the limitation, they were not able to find a way to train the network and stated that the problem of training may be unsolvable. This work causes much pessimism in neural network research and many researchers have left the filed. This is the reason that during the 1970s, the filed has been essentially dormant with very little research activity.

The renewed interest in neural network started in the 1980s when Hopfield (1982) used statistical mechanics to explain the operations of a certain class of recurrent network and demonstrated that neural networks could be trained as an associative memory. Hopfield networks have been used successfully in solving the Traveling Salesman Problem which is a constrained optimization problem (Hopfield and Tank, 1985). At about the same time, Kohonen (1982) developed a neural network based on self-organization whose key idea is to represent sensory signals as two-dimensional images or maps. Kohonen's networks, often called Kohonen's feature maps or self-organizing maps, organized neighborhoods of neurons such that similar inputs into the model are topologically close. Because of the usefulness of these two types of networks in solving real problems, more research was devoted to neural networks.

The most important development in the field was doubtlessly the invention of efficient training algorithms—called backpropagation—for multilayer perceptrons which have long been suspected to be capable of overcoming the linear separability limitation of the simple perceptron but have not been used due to lack of good training algorithms. The backpropagation algorithm, originated from Widrow and Hoff's learning rule, formalized by Werbos (1974), developed by Parker (1985), Rumelhart Hinton, and Williams (Rumelhart Hinton & Williams, 1986) and others, and popularized by Rumelhart, et al. (1986), is a systematic method for training multilayer neural networks. As a result of this algorithm, multilayer perceptrons are able to solve many important practical problems, which is the major reason that reinvigorated the filed of neural networks. It is by far the most popular learning paradigm in neural networks applications.

Since then and especially in the 1990s, there have been significant research activities devoted to neural networks. In the last 15 years or so, tens of thousands of papers have been published and numerous successful applications have been reported. It will not be surprising to see even greater advancement and success of neural networks in various data mining applications in the future.

3 Neural Network Models

As can be seen from the short historical review of development of the neural network field, many types of neural networks have been proposed. In fact, several dozens of different neural network models are regularly used for a variety of problems. In this section, we focus on three better known and most commonly used neural network models for data mining purposes: the multilayer feedforward network, the Hopfield network, and the Kohonen's map. It is important to point out that there are numerous variants of each of these networks and the discussions below are limited to the basic model formats.

3.1 Feedforward Neural Networks

The multilayer feedforward neural networks, also called multi-layer perceptrons (MLP), are the most widely studied and used neural network model in practice. According to Wong, Bodnovich, and Selvi (1997), about 95% of business applications of neural networks reported in the literature use this type of neural model. Feedforward neural networks are ideally suitable for modeling relationships between a set of predictor or input variables and one or more response or output variables. In other words, they are appropriate for any functional mapping problem where we want to know how a number of input variables affect the output variable(s). Since most prediction and classification tasks can be treated as function mapping problems, the MLP networks are very appealing to data mining. For this reason, we will focus more on feedforward networks and many issues discussed here can be extended to other types of neural networks.

Model Structure

An MLP is a network consisted of a number of highly interconnected simple computing units called neurons, nodes, or cells, which are organized in layers. Each neuron performs simple task of information processing by converting received inputs into processed outputs. Through the linking arcs among these neurons, knowledge can be generated and stored as arc weights regarding the strength of the relationship between different nodes. Although each neuron implements its function slowly and imperfectly, collectively a neural network is able to perform a variety of tasks efficiently and achieve remarkable results.

Figure 1 shows the architecture of a three-layer feedforward neural network that consists of neurons (circles) organized in three layers: input layer, hidden layer, and output layer. The neurons in the input nodes correspond to the independent or predictor variables that are believed to be useful for predicting the dependent variables which correspond to the output neurons. Neurons in the input layer are passive; they do not process information but are simply used to receive the data patterns and then pass them into the neurons into the next layer. Neurons in the hidden layer are connected to both input and output neurons and are key to learning the pattern in the data and mapping the relationship from input variables to the output variable. Although it is possible to have more than one hidden layer in a multilayer networks, most applications use only one layer. With nonlinear transfer functions, hidden neurons can process complex information received from input neurons and then send processed information to output layer for further processing to generate outputs. In feedforward neural networks, the information flow is one directional from the input to hidden then to output layer and there is no feedback from the output.

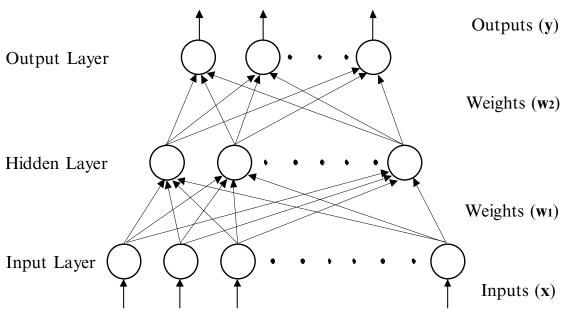


Fig. 1. Multi-layer feedforward neural network

Thus, a feedforward multilayer neural network is characterized by its architecture determined by the number of layers, the number of nodes in each layer, the transfer function used in each layer, as well as how the nodes in each layer connected to nodes in adjacent layers. Although partial connection between nodes in adjacent layers and direct connection from input layer to output layer are possible, the most commonly used neural network is so called fully connected one in that each node at one layer is fully connected only to all nodes in the adjacent layers.

To understand how the network in Figure 1 works, we need first understand the way neurons in the hidden and output layers process information. Figure 2 provides the mechanism that shows how a neuron processes information from several inputs and then converts it into an output. Each neuron processes information in two steps. In the first step, the inputs (x_i) are combined together to form a weighted sum of inputs and the weights (w_i) of connecting links. The 2^{nd} step then performs a transformation that converts the sum to an output via a transfer function. In other words, the neuron in Figure 2 performs the following operations:

$$Out_n = f\left(\sum_i w_i x_i\right), \tag{1}$$

where Out_n is the output from this particular neuron and $f$is the transfer function. In general, the transfer function is a bounded nondecreasing function. Although there are many possible choices for transfer functions, only a few of them are commonly used in practice. These include

1. the sigmoid (logistic) function, $f(x) = (1 + \exp(-x))^{-1}$,
2. the hyperbolic tangent function, $f(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$,
3. the sine and cosine function, $f(x) = \sin(x)$, $f(x) = \cos(x)$, and
4. the linear or identity function, $f(x) = x$.

Among them, the logistic function is the most popular choice especially for the hidden layer nodes due to the fact that it is simple, has a number of good characteristics (bounded, nonlinear, and monotonically increasing), and bears a better resemblance to real neurons (Hinton, 1992).

In Figure 1, let $\mathbf{x} = (x_1, x_2, ..., x_d)$ be a vector of d predictor or attribute variables, $\mathbf{y} = (y_1, y_2, ..., y_M)$be the M-dimensional output vector from the network, and $\mathbf{w_1}$ and $\mathbf{w_2}$ be the matrices of linking arc weights from input to hidden layer and from hidden to output layer, respectively. Then a three-layer neural network can be written as a nonlinear model of the form

$$\mathbf{y} = f_2(\mathbf{w_2} f_1(\mathbf{w_1 x})), \tag{2}$$

where f_1 and f_2 are the transfer functions for the hidden nodes and output nodes respectively. Many networks also contain node biases which are constants added to the hidden and/or output nodes to enhance the flexibility of neural network modeling. Bias terms act like the intercept term in linear regression.

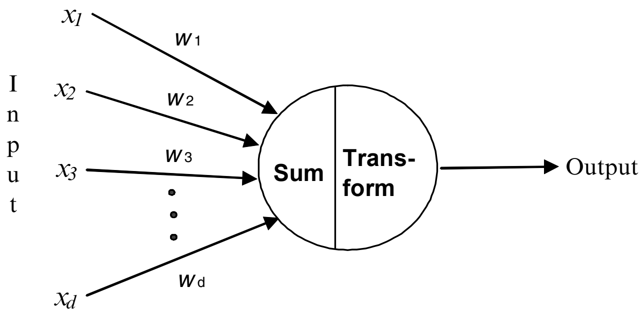


Fig. 2. Information processing in a single neuron

In classification problems where desired outputs are binary or categorical, logistic function is often used in the output layer to limit the range of the network outputs. On the other hand, for prediction or forecasting purposes, since output variables are in general continuous, linear transfer function is a better choice for output nodes. Equation (3) can have many different specifications depending on the problem type, the transfer function, and numbers of input, hidden, and output nodes employed. For example, the neural network structure for a general univariate forecasting problem with logistic function for hidden nodes and identity function for the output node can be explicitly expressed as

$$y_t = w_{10} + \sum_{j=1}^{q} w_{1j} f(\sum_{i=1}^{p} w_{ij} x_{it} + w_{0j}) \tag{3}$$

where y_t is the observation of forecast variable and $\{x_{it},\ i = \mathit{1},\ \mathit{2},\ \ldots,\ p\}$ are p predictor variables at time t, p is also the number of input nodes, q is the number of hidden nodes, $\{w_{1j}, j = 0, 1, ..., n\}$ are weights from the hidden to output nodes and $\{w_{ij}, i = 0, 1, ..., p;\ j = 1, 2, ..., q\}$ are weights from the input to hidden nodes; α_0 and β_{0j} are bias terms, and f is the logistic function defined above.

Network Training

The arc weights are the parameters in a neural network model. Like in a statistical model, these parameters need to be estimated before the network can be adopted for further use. Neural network training refers to the process in which these weights are determined, and hence is the way the network learns. Network training for classification and prediction problems is performed via supervised learning in which known outputs and their associated inputs are both presented to the network.

The basic process to train a neural network is as follows. First, the network is fed with training examples, which consist of a set of input patterns and their desired outputs. Second, for each training pattern, the input values are weighted and summed at each hidden layer node and the weighted sum is then transmitted by an appropriate transfer function into the hidden node's output value, which becomes the input to the output layer nodes. Then, the network output values are calculated and compared to the desired or target values to determine how closely the actual network outputs match the desired outputs. Finally, the weights of the connection are changed so that the network can produce a better approximation to the desired output. This process typically repeats many times until differences between network output values and the known target values for all training patterns are as small as possible.

To facilitate training, some overall error measure such as the mean squared errors (MSE) or sum of squared errors (SSE) is often used to serve as an objective function or performance metric. For example, MSE can be defined as

$$\mathrm{MSE} = \frac{1}{M}\frac{1}{N}\sum_{m=1}^{M}\sum_{j=1}^{N}(d_{mj} - y_{mj})^2, \tag{4}$$

where d_{mj} and y_{mj} represent the desired (target) value and network output at the mth node for the jth training pattern respectively, M is the number of output nodes, and N is the number of training patterns. The goal of training is to find the set of weights that minimize the objective function. Thus, network training is actually an unconstrained nonlinear optimization problem. Numerical methods are usually needed to solve nonlinear optimization problems.

The most important and popular training method is the backpropagation algorithm which is essentially a gradient steepest descent method. The idea of steepest descent method is to find the best direction in the multi-dimension error space to move or change the weights so that the objective function is reduced most. This requires partial derivative of the objective function with respect to each weight to be calculated because the partial derivative represents the rate of change of the objective function. The weight updating therefore follows the following rule

$$\begin{aligned} w_{ij}^{new} &= w_{ij}^{old} + \Delta w_{ij} \\ \Delta w_{ij} &= -\eta \tfrac{\partial E}{\partial w_{ij}} \end{aligned} \tag{5}$$

where $\Delta w_{ij}$is the gradient of objective function E with respect to weight w_{ij}, and η is called the learning rate which controls the size of the gradient descent step. The algorithm requires an iterative process and there are two versions of weight updating schemes: batch mode and on-line mode. In the batch mode, weights are updated after all training patterns are evaluated, while in the on-line learning mode, the weights are updated after each pattern presentation. The basic steps with the batch mode training can be summarized as
initialize the weights to small random values from, say, a uniform distribution

choose a pattern and forward propagate it to obtain network outputs
calculate the pattern error and back-propagate it to obtain partial derivative of this error with respect to all weights
add up all the single-pattern terms to get the total derivative
update the weights with equation (7)
repeat steps 2-5 for next pattern until all patterns are passed through.

Note that each one pass of all patterns is called an epoch. In general, each weight update reduces the total error by only a small amount so many epochs are often needed to minimize the error. For information on further detail of the backpropagation algorithm, readers are referred to Rumelhart et al. (1986) and Bishop (1995).

It is important to note that there is no algorithm currently available which can guarantee global optimal solution for general nonlinear optimization problems such as those in neural network training. In fact, all algorithms in nonlinear optimization inevitably suffer from the local optima problems and the most we can do is to use the available optimization method which can give the "best" local optima if the true global solution is not available. It is also important to point out that the steepest descent method used in the basic backpropagation suffers the problems of slow convergence, inefficiency, and lack of robustness. Furthermore, it can be very sensitive to the choice of the learning rate. Smaller learning rates tend to slow the learning process while larger learning rates may cause network oscillation in the weight space. Common modifications to the basic backpropagation include adding in the weight updating formula (2) an additional momentum parameter proportional to the last weight change the to control the oscillation in weight changes and (3) a weight decay term that penalizes the overly complex network with large weights.

In light of the weakness of the standard backpropagation algorithm, the existence of many different optimization methods (Fletcher, 1987) provides various alternative choices for the neural network training. Among them, the second-order methods such as BFGS and Levenberg-Marquardt methods are more efficient nonlinear optimization methods and are used in most optimization packages. Their faster convergence, robustness, and the ability to find good local minima make them attractive in neural network training. For example, De Groot and Wurtz (1991) have tested several well-known optimization algorithms such as quasi-Newton, BFGS, Levenberg-Marquardt, and conjugate gradient methods and achieved significant improvements in training time and accuracy.

Modeling Issues

Developing a neural network model for a data mining application is not a trivial task. Although many good software packages exist to ease users' effort in building a neural network model, it is still critical for data miners to understand many important issues around the model building process. It is important to point out that building a successful neural network is a combination of art and science and software alone is not sufficient to solve all problems in the process. It is a pitfall to blindly throw data into a software package and then hope it will automatically identify the pattern or give a satisfactory solution. Other pitfalls readers need to be cautious can be found in Zhang (2007).

An important point in building an effective neural network model is the understanding of the issue of learning and generalization inherent in all neural network applications. This issue of learning and generalization can be understood with the concepts of model bias and variance (Geman, Bienenstock & Doursat, 1992). Bias and variance are important statistical properties associated with any empirical model. Model bias measures the systematic error of a model in *learning* the underlying relations among variables or observations. Model variance, on the other hand, relates to the stability of a model built on different data samples and therefore offers insights on *generalizability* of the model. A pre-specified or parametric model, which is less dependent on the data, may misrepresent the true functional relationship and hence cause a large bias. On the other hand, a flexible, data-driven model may be too dependent on the specific data set and hence have a large variance. Bias and variance are two important terms that impact a model's usefulness. Although it is desirable to have both low bias and low variance, we may not be able to reduce both terms at the same time for a given data set because these goals are conflicting. A model that is less dependent on the data tends to have low variance but high bias if the pre-specified model is incorrect. On the other hand, a model that fits the data well tends to have low bias but high variance when applied to new data sets. Hence a good predictive model should have an "appropriate" balance between model bias and model variance.

As a data-driven approach to data mining, neural networks often tend to fit the training data well and thus have low bias. But the potential price to pay is the overfitting effect that causes high variance. Therefore, attentions should be paid to address issues of overfitting and the balance of bias and variance in neural network model building.

The major decisions in building a neural network model include data preparation, input variable selection, choice of network type and architecture, transfer function, and training algorithm, as well as model validation, evaluation, and selection procedures. Some of these can be solved during the model building process while others must be considered before actual modeling starts.

Neural networks are data-driven techniques. Therefore, data preparation is a critical step in building a successful neural network model. Without an adequate and representative data set, it is impossible to develop a useful data mining model.

There are several practical issues around the data requirement for a neural network model. The first is the data quality. As data sets used for typical data mining tasks are massive and may be collected from multiple sources, they may suffer many quality problems such as noises, errors, heterogeneity, and missing observations. Results reported in Klein and Rossin (1999) suggest that data error rate and its magnitude can have substantial impact on neural network performance. Klein and Rossion believe that an understanding of errors in a dataset should be an important consideration to neural network users and efforts to lower error rates are well deserved. Appropriate treatment of these problems to clean the data is critical for successful application of any data mining technique including neural networks (Dasu and Johnson, 2003).

Another one is the size of the sample used to build a neural network. While there is no specific rule that can be followed for all situations, the advantage of having large samples should be clear because not only do neural networks have typically a large number of parameters to estimate, but also it is often necessary to split data into several portions for overfitting prevention, model selection, evaluation, and comparison. A larger sample provides better chance for neural networks to adequately approximate the underlying data structure.

The third issue is the data splitting. Typically for neural network applications, all available data are divided into an in-sample and an out-of-sample. The in-sample data are used for model fitting and selection, while the out-of-sample is used to evaluate the predictive ability of the model. The in-sample data often are further split into a training sample and a validation sample. The training sample is used for model parameter estimation while the validation sample is used to monitor the performance of neural networks and help stop training and select the final model. For a neural network to be useful, it is critical to test the model with an independent out-of-sample which is not used in the network training and model selection phase. Although there is no consensus on how to split the data, the general practice is to allocate more data for model building and selection although it is possible to allocate 50% vs. 50% for in-sample and out-of-sample if the data size is very large. Typical split in data mining applications reported in the literature uses convenient ratio varying from 70%:30% to 90%:10%.

Data preprocessing is another issue that is often recommended to highlight important relationships or to create more uniform data to facilitate neural network learning, meet algorithm requirements, and avoid computation problems. For time series forecasting, Azoff (1994) summarizes four methods typically used for input data normalization. They are along channel normalization, across channel normalization, mixed channel normalization, and external normalization. However, the necessity and effect of data normalization on network learning and forecasting are still not universally agreed upon. For example, in modeling and forecasting seasonal time series, some researchers (Gorr, 1994) believe that data preprocessing is not necessary because the neural network is a universal approximator and is able to capture all of the underlying patterns well. Recent empirical studies (Nelson, Hill, Remus & O'Connor, 1999; Zhang and Qi, 2002), however, find that pre-deseasonalization of the data is critical in improving forecasting performance.

Neural network design and architecture selection are important yet difficult tasks. Not only are there many ways to build a neural network model and a large number of choices to be made during the model building and selection process, but also numerous parameters and issues have to be estimated and experimented before a satisfactory model may emerge. Adding to the difficulty is the lack of standards in the process. Numerous rules of thumb are available but not all of them can be applied blindly to a new situation. In building an appropriate model, some experiments with different model structures are usually necessary. Therefore, a good experiment design is needed. For further discussions of many aspects of modeling issues for classification and forecasting tasks, readers may consult Bishop (1995), Zhang, Patuwo, and Hu (1998), and Remus and O'Connor (2001).

For network architecture selection, there are several decisions to be made. First, the size of output layer is usually determined by the nature of the problem. For example, in most time series forecasting problems, one output node is naturally used for one-step-ahead forecasting, although one output node can also be employed for multi-step-ahead forecasting in which case, iterative forecasting mode must be used. That is, forecasts for more than two-step ahead in the time horizon must be based on earlier forecasts. On the other hand, for classification problems, the number of output nodes is determined by the number of groups into which we classify objects. For a two-group classification problem, only one output node is needed while for a general M-group problem, M binary output nodes can be employed.

The number of input nodes is perhaps the most important parameter in an effective neural network model. For classification or causal forecasting problems, it corresponds to the number of feature (attribute) variables or independent (predictor) variables that data miners believe important in predicting the output or dependent variable. These input variables are usually pre-determined by the domain expert although variable selection procedures can be used to help identify the most important variables. For univariate forecasting problems, it is the number of past lagged observations. Determining an appropriate set of input variables is vital for neural networks to capture the essential relationship that can be used for successful prediction. How many and what variables to use in the input layer will directly affect the performance of neural network in both in-sample fitting and out-of-sample prediction.

Neural network model selection is typically done with the basic cross-validation process. That is the in-sample data is split into a training set and a validation set. The neural network parameters are estimated with the training sample, while the performance of the model is monitored and evaluated with the validation sample. The best model selected is the one that has the best performance on the validation sample. Of course, in choosing competing models, we must also apply the principle of parsimony. That is, a simpler model that has about the same performance as a more complex model should be preferred. Model selection can also be done with all of the in-sample data. This can be done with several in-sample selection criteria that modify the total error function to include a penalty term that penalizes for the complexity of the model. Some in-sample model selection approaches are based on criteria such as Akaike's information criterion (AIC) or Schwarz information criterion (SIC). However, it is important to note the limitation of these criteria as empirically demonstrated by Swanson and White (1995) and Qi and Zhang (2001). Other in-sample approaches are based on pruning methods such as node and weight pruning (see a review by Reed, 1993) as well as constructive methods such as the upstart and cascade correlation approaches (Fahlman and Lebiere, 1990; Frean, 1990).

After the modeling process, the finally selected model must be evaluated using data not used in the model building stage. In addition, as neural networks are often used as a nonlinear alternative to traditional statistical models, the performance of neural networks needs be compared to that of statistical methods. As Adya and Collopy (1998) point out, "if such a comparison is not conducted it is difficult to argue that the study has taught us much about the value of neural networks." They further propose three evaluation criteria to objectively evaluate the performance of a neural network: (2) comparing it to well-accepted (traditional) models; (3) using true out-of-samples; and (4) ensuring enough sample size in the out-of-sample (40 for classification problems and 75 for time series problems). It is important to note that the test sample served as out-of-sample should not in any way be used in the model building process. If the cross-validation is used for model selection and experimentation, the performance on the validation sample should not be treated as the true performance of the model.

Relationships with Statistical Methods

Neural networks especially the feedforward multilayer networks are closely related to statistical pattern recognition methods. Several articles that illustrate their link include Ripley (1993, 1994), Cheng and Titterington (1994), Sarle (1994), and Ciampi and Lechevallier (1997). This section provides a summary of the literature that links neural networks, particularly MLP networks to statistical data mining methods.

Bayesian decision theory is the basis for statistical classification methods. It provides the fundamental probability model for well known classification procedures. It has been shown by many researchers that for classification problems, neural networks provide the direct estimation of the posterior probabilities under a variety of situations (Richard and Lippmann, 1991). Funahashi (1998) shows that for the two-group d-dimensional Gaussian classification problem, neural networks with at least $2d$ hidden nodes have the capability to approximate the posterior probability with arbitrary accuracy when infinite data is available and the training proceeds ideally. Miyake and Kanaya (1991) shows that neural networks trained with a generalized mean-squared error objective function can yield the optimal Bayes rule.

As the statistical counterpart of neural networks, discriminant analysis is a well-known supervised classifier. Gallinari, Thiria, Badran, and Fogelman-Soulie (1991) describe a general framework to establish the link between discriminant analysis and neural network models. They find that in quite general conditions the hidden layers of an MLP project the input data onto different clusters in a way that these clusters can be further aggregated into different classes. The discriminant feature extraction by the network with nonlinear hidden nodes has also been demonstrated in Webb and Lowe (1990) and Lim, Alder and Hadingham (1992).

Raudys (1998a, b) presents a detailed analysis of nonlinear single layer perceptron (SLP). He shows that by purposefully controlling the SLP classifier complexity during the adaptive training process, the decision boundaries of SLP classifiers are equivalent or close to those of seven statistical classifiers. These statistical classifiers include the Euclidean distance classifier, the Fisher linear discriminant function, the Fisher linear discriminant function with pseudo-inversion of the covariance matrix, the generalized Fisher linear discriminant function, the regularized linear discriminant analysis, the minimum empirical error classifier, and the maximum margin classifier.

Logistic regression is another important data mining tool. Schumacher, Robner and Vach (1996) make a detailed comparison between neural networks and logistic regression. They find that the added modeling flexibility of neural networks due to hidden layers does not automatically guarantee their superiority over logistic regression because of the possible overfitting and other inherent problems with neural networks (Vach Schumacher & Robner, 1996).

For time series forecasting problems, feedforward MLP are general nonlinear autoregressive models. For a discussion of the relationship between neural networks and general ARMA models, see Suykens, Vandewalle, and De Moor (1996).

3.2 Hopfield Neural Networks

Hopfield neural networks are a special type of neural networks which are able to store certain memories or patterns in a manner similar to the brain—the full pattern can be recovered if the network is presented with only partial or noisy information. This ability of brain is often called associative or content-addressable memory. Hopfield networks are quite different from the feedforward multilayer networks in several ways. From the model architecture perspective, Hopfield networks do not have a layer structure. Rather, a Hopfield network is a single layer of neurons with complete interconnectivity. That is, Hopfield networks are autonomous systems with all neurons being both inputs and outputs and no hidden neurons. In addition, unlike in feedforward networks where information is passed only in one direction, there are looping feedbacks among neurons.

Figure 3 shows a simple Hopfield network with only three neurons. Each neuron is connected to every other neuron and the connection strengths or weights are symmetric in that the weight from neuron i to neuron j (w_{ij}) is the same as that from neuron j to neuron $i(w_{ji})$. The flow of the information is not in a single direction as in the feedforward network. Rather it is possible for signals to flow from a neuron back to itself via other neurons. This feature is often called feedback or recurrent because neurons may be used repeatedly to process information.

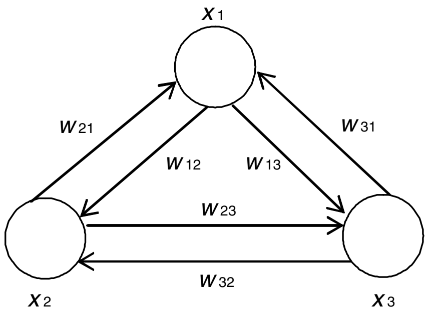


Fig. 3. A three-neuron Hopfield network

The network is completely described by a state vector which is a function of time t. Each node in the network contributes one component to the state vector and any or all of the node outputs can be treated as outputs of the network. The dynamics of neurons can be described mathematically as the following equations:

$$u_i(t) = \sum_{j=1}^{n} w_{ij} x_j(t) + v_i \tag{6}$$

where $u_i(t)$ is the internal state of the ith neuron, $x_i(t)$ is the output activation or output state of the ith neuron, v_i is the threshold to the ith neuron, n is the number of neurons, and *sign* is the sign function defined as $sign(x)$=1, if x >0 and -1 otherwise. Given a set of initial conditions $\mathbf{x}(0)$, and appropriate restrictions on the weights (such as symmetry), this network will converge to a fixed equilibrium point.

For each network state at any time, there is an energy associated with it. A common energy function is defined as

$$E(t) = -\frac{1}{2}\mathbf{x}(t)^T \mathbf{W}\mathbf{x}(t) - \mathbf{x}(t)^T \mathbf{v} \tag{7}$$

where $\mathbf{x}$(t) is the state vector, $\mathbf{W}$ is the weight matrix, $\mathbf{v}$ is the threshold vector, and T denote transpose. The basic idea of the energy function is that it always decreases or at least remains constant as the system evolves over time according to its dynamic rule in equations 6 and 7. It can be shown that the system will converge from an arbitrary initial energy to eventually a fixed point (a local minimum) on the surface of the energy function. These fixed points are stable states which correspond to the stored patterns or memories.

The main use of Hopfield's network is as associative memory. An associative memory is a device which accepts an input pattern and generates an output as the stored pattern which is most closely associated with the input. The function of the associate memory is to recall the corresponding stored pattern, and then produce a clear version of the pattern at the output. Hopfield networks are typically used for those problems with binary pattern vectors and the input pattern may be a noisy version of one of the stored patterns. In the Hopfield network, the stored patterns are encoded as the weights of the network.

There are several ways to determine the weights from a training set which is a set of known patterns. One way is to use a prescription approach given by Hopfield (1982). With this approach, the weights are given by

$$\mathbf{w} = \frac{1}{n} \sum_{i=1}^{p} z_i z_i^T \tag{8}$$

where z_i, $i \quad = 1, 2, \ldots, p$ are $p$patterns that are to be stored in the network. Another way is to use an incremental, iterative process called Hebbian learning rule developed by Hebb (1949). It has the following learning process:
choose a pattern from the training set at random
present a pair of components of the pattern at the outputs of the corresponding nodes of the network
if two nodes have the same value then make a small positive increment to the interconnected weight. If they have opposite values then make a small negative decrement to the weight. The incremental size can be expressed as $\Delta w_{ij} = \alpha z_i^p z_j^p$, where α is a constant rate in between 0 and 1 and z_i^p is the ith component of pattern p.

Hopfield networks have two major limitations when used as a content addressable memory. First, the number of patterns that can be stored and accurately recalled is fairly limited. If too many patterns are stored, the network may converge to a spurious pattern different from all programmed patterns. Or, it may not converge at all. The second limitation is that the network may become unstable if the common patterns it shares are too similar. An example pattern is considered unstable if it is applied at time zero and the network converges to some other pattern from the training set.

3.3 Kohonen's Self-organizing Maps

Kohonen's self-organizing maps (SOM) are important neural network models for dimension reduction and data clustering. SOM can learn from complex, multidimensional data and transform them into a topological map of much fewer dimensions typically one or two dimensions. These low dimension plots provide much improved visualization capabilities to help data miners visualize the clusters or similarities between patterns.

SOM networks represent another neural network type that is markedly different from the feedforward multilayer networks. Unlike training in the feedforward MLP, the SOM training or learning is often called the unsupervised because there are no known target outputs associated with each input pattern in SOM and during the training process, the SOM processes the input patterns and learns to cluster or segment the data through adjustment of weights. A two-dimensional map is typically created in such a way that the orders of the interrelationships among inputs are preserved. The number and composition of clusters can be visually determined based on the output distribution generated by the training process. With only input variables in the training sample, SOM aims to learn or discover the underlying structure of the data.

A typical SOM network has two layers of nodes, an input layer and output layer (sometimes called the Kohonen layer). Each node in the input layer is fully connected to nodes in the two-dimensional output layer. Figure 4 shows an example of an SOM network with several input nodes in the input layer and a two dimension output layer with a 4x4 rectangular array of 16 neurons. It is also possible to use hexagonal array or higher dimensional grid in the Kohonen layer. The number of nodes in the input layer is corresponding to the number of input variables while the number of output nodes depends on the specific problem and is determined by the user. Usually, this number of neurons in the rectangular array should be large enough to allow a sufficient number of clusters to form. It has been recommended that this number is ten times the dimension of the input pattern (Deboeck and Kohonen, 1998)

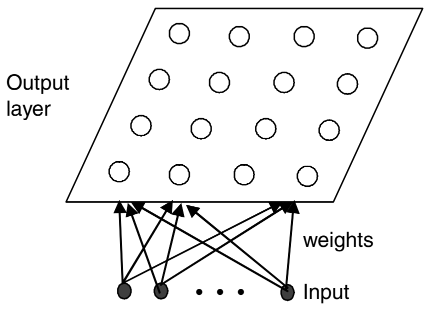


Fig. 4. A 4x4 SOM network

During the training process, input patterns are presented to the network. At each training step when an input pattern $\mathbf{x}$ randomly selected from the training set is presented, each neuron i in the output layer calculates how similar the input is to its weights $\mathbf{w}_i$. The similarity is often measured by some distance between $\mathbf{x}$ and $\mathbf{w}_i$. As the training proceeds, the neurons adjust their weights according to the topological relations in the input data. The neuron with the minimum distance is the winner and the weights of the winning node as well as its neighboring nodes are strengthened or adjusted to be closer to the value of input pattern. Therefore, the training with SOM is unsupervised and competitive with winner-take-all strategy.

A key concept in training SOM is the neighborhood N_k around a winning neuron, k, which is the collection of all nodes with the same radial distance. Figure 5 gives an example of neighborhood nodes for a 5x5 Kohonen layer at radius of 1 and 2.

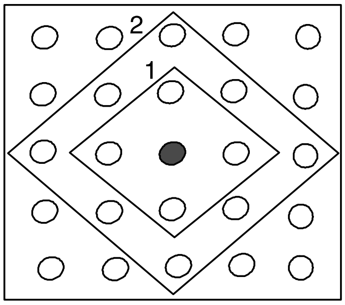


Fig. 5. A 5x5 Kohonen Layer with two neighborhood sizes

The basic procedure in training an SOM is as follows:
initialize the weights to small random values and the neighborhood size large enough to cover half the nodes
select an input pattern **x** randomly from the training set and present it to the network
find the best matching or "winning" node k whose weight vector w_k is closest to the current input vector **x** using the vector distance. That is:

$$\|x - w_k\| = \min_i \|x - w_i\|$$

where $\|.\|$ represents the Euclidean distance
update the weights of nodes in the neighborhood of k using the Kohonen learning rule:
$w_i^{new} = w_i^{old} + \alpha h_{ik}(x - w_i)$if i is in N_k
$w_i^{new} = w_i^{old}$if $i$is not in $N_k(10)$
where α is the learning rate between 0 and 1 and h_{ik} is a neighborhood kernel centered on the winning node and can take Gaussian form as

$$h_{ik} = \exp\left[-\frac{\|r_i - r_k\|^2}{2\sigma^2}\right] \tag{9}$$

where r_i and r_k are positions of neurons i and k on the SOM grid and σ is the neighborhood radius.

decrease the learning rate slightly
repeat Steps 1—5 with a number of cycles and then decrease the size of the neighborhood. Repeat until weights are stabilized.

As the number of cycles of training (epochs) increases, better formation of the clusters can be found. Eventually, the topological map is fine-tuned with finer distinctions of clusters within areas of the map. After the network has been trained, it can be used as a visualization tool to examine the data structure. Once clusters are identified, neurons in the map can be labeled to indicate their meaning. Assignment of meaning usually requires knowledge on the data and specific application area.

4 Data Mining Applications

Neural networks have been used extensively in data mining for a wide variety of problems in business, engineering, industry, medicine, and science. In general, neural networks are good at solving the following common data mining problems such as classification, prediction, association, and clustering. This section provides a short overview on the application areas.

Classification is one of the frequently encountered data mining tasks. A classification problem occurs when an object needs to be assigned into a predefined group or class based on a number of observed attributes related to that object. Many problems in business, industry, and medicine can be treated as classification problems. Examples include bankruptcy prediction, credit scoring, medical diagnosis, quality control, handwritten character recognition, and speech recognition. Feed-forward multilayer networks are most commonly used for these classification tasks although other types of neural networks can also be used.

Forecasting is central to effective planning and operations in all business organizations as well as government agencies. The ability to accurately predict the future is fundamental to many decision activities in finance, marketing, production, personnel, and many other business functional areas. Increasing forecasting accuracy could facilitate the saving of millions of dollars to a company. Prediction can be done with two approaches: causal and time series analysis, both of which are suitable for feedforward networks. Successfully applications include predictions of sales, passenger volume, market share, exchange rate, futures price, stock return, electricity demand, environmental changes, and traffic volume.

Clustering involves categorizing or segmenting observations into groups or clusters such that each cluster is as homogeneous as possible. Unlike classification problems, the groups or clusters are usually unknown to or not predetermined by data miners. Clustering can simplify a complex large data set into a small number of groups based on the natural structure of data. Improved understanding of the data and subsequent decisions are major benefits of clustering. Kohonen or SOM networks are particularly useful for clustering

Table 1. Data mining applications of neural networks

Data Mining Task	Application Area
Classification	bond rating (Dutta and shenkar, 1993)
	corporation failure (Zhang et al., 1999; Mckee and Greenstein, 2000)
	credit scoring (West, 2000)
	customer retention (Mozer and Wolniewics, 2000; Smith et al., 2000)
	customer satisfaction (Temponi et al., 1999)
	fraud detection (He et al., 1997)
	inventory (Partovi and Anandarajan, 2002)
	project (Thieme et al., 2000; Zhang et al., 2003)
	target marketing (Zahavi and Levin, 1997)
Prediction	air quality (Kolehmainen et al., 2001)
	business cycles and recessions (Qi, 2001)
	consumer expenditures (Church and Curram, 1996)
	consumer choice (West et al., 1997)
	earnings surprises (Dhar and Chou, 2001)
	economic crisis (Kim et al., 2004)
	exchange rate (Nag and Mitra, 2002)
	market share (Agrawal and Schorling, 1996)
	ozone concentration level (Prybutok et al., 2000)
	sales (Ansuj et al., 1996; Kuo, 2001; Zhang and Qi, 2002)
	stock market (Qi, 1999; Chen et al., 2003; Leung et al., 2000; Chun and Kim, 2004)
	tourist demand (Law, 2000)
	traffic (Dia, 2001; Qiao et al., 2001)
Clustering	bankruptcy prediction (Kiviluoto, 1998)
	document classification (Dittenbach et al., 2002)
	enterprise typology (Petersohn, 1998)
	fraud uncovering (Brockett et al., 1998)
	group technology (Kiang et al., 1995)
	market segmentation (Ha and Park, 1998; Vellido et al., 1999; Reutterer and Natter, 2000; Boone and Roehm, 2002)
	process control (Hu and Rose, 1995)
	property evaluation (Lewis et al., 1997)
	quality control (Chen and Liu, 2000)
	webpage usage (Smith and Ng, 2003)
Association/Pattern Recognition	defect recognition (Kim and Kumara, 1997)
	facial image recognition (Dai and Nakano, 1998)
	frequency assignment (Salcedo-Sanz et al., 2004)
	graph or image matching (Suganthan et al., 1995; Pajares et al., 1998)
	image restoration (Paik and Katsaggelos, 1992; Sun and Yu, 1995)
	imgage segmentation (Rout et al., 1998; Wang et al., 1992)
	landscape pattern prediction (Tatem et al., 2002)
	market basket analysis (Evans, 1997)
	object recognition (Huang and Liu, 1997; Young et al., 1997; Li and Lee, 2002)
	on-line marketing (Changchien and Lu, 2001)
	pattern sequence recognition (Lee, 2002)
	semantic indexing and searching (Chen et al., 1998)

tasks. Applications have been reported in market segmentation, customer targeting, business failure categorization, credit evaluation, document retrieval, and group technology.

With association techniques, we are interested in the correlation or relationship among a number variables or objects. Association is used in several ways. One use as in market basket analysis is to help identify the consequent items given a set of antecedent items. An association rule in this way is an implication of the form: IF $\mathbf{x}$, THEN Y, where $\mathbf{x}$ is a set of antecedent items and Y is the consequent items. This type of association rule has been used in a variety of data mining tasks including credit card purchase analysis, merchandise stocking, insurance fraud investigation, market basket analysis, telephone calling pattern identification, and climate prediction. Another use is in pattern recognition. Here we train a neural network first to *remember* a number of patterns, so that when a distorted version of a stored pattern is presented, the network associates it with the closest one in its memory and returns the original version of the pattern. This is useful for restoring noisy data. Speech, image, and character recognitions are typical application areas. Hopfield networks are useful for this purpose.

Given an enormous amount of applications of neural networks in data mining, it is difficult if not impossible to give a detailed list. Table 1 provides a sample of several typical applications of neural networks for various data mining problems. It is important to note that studies given in Table 1 represent only a very small portion of all the applications reported in the literature, but we should still get an appreciation of the capability of neural networks in solving wide range of data mining problems. For real-world industrial or commercial applications, readers are referred to Widrow et al. (1994), Soulie and Gallinari (1998), Jain and Vemuri (1999), and Lisboa, Edisbury, and Vellido (2000).

5 Conclusions

Neural networks are standard and important tools for data mining. Many features of neural networks such as nonlinear, data-driven, universal function approximating, noise-tolerance, and parallel processing of large number of variables are especially desirable for data mining applications. In addition, many types of neural networks functionally are similar to traditional statistical pattern recognition methods in areas of cluster analysis, nonlinear regression, pattern classification, and time series forecasting. This chapter provides an overview of neural networks and their applications to data mining tasks. We present three important classes of neural network models: Feedforward multilayer networks, Hopfield networks, and Kohonen's self-organizing maps, which are suitable for a variety of problems in pattern association, pattern classification, prediction, and clustering.

Neural networks have already achieved significant progress and success in data mining. It is, however, important to point out that they also have limitations and may not be a panacea for every data mining problem in every situation. Using neural networks require thorough understanding of the data, prudent design of modeling strategy, and careful consideration of modeling issues. Although many rules of thumb exist in model building, they are not necessarily always useful for a new application. It is suggested that users should not blindly rely on a neural network package to "automatically" mine the data, but rather should study the problem and understand the network models and the issues in various stages of model building, evaluation, and interpretation.

References

Adya M., Collopy F. (1998), How effective are neural networks at forecasting and prediction? a review and evaluation. Journal of forecasting ; 17:481-495.

Agrawal D., Schorling C. (1996), Market share forecasting: an empirical comparison of artificial neural networks and multinomial logit model. Journal of Retailing ; 72:383-407.

Ahn H., Choi E., Han I. (2007), Extracting underlying meaningful features and canceling noise using independent component analysis for direct marketing Expert Systems with Applications, ; 33: 181-191

Azoff E. M. (1994), Neural Network Time Series Forecasting of Financial Markets. Chichester: John Wiley & Sons, .

Bishop M. (1995), Neural Networks for Pattern Recognition. Oxford: Oxford University Press, .

Boone D., Roehm M. (2002), Retail segmentation using artificial neural networks. International Journal of Research in Marketing ; 19:287-301.

Brockett P.L., Xia X.H., Derrig R.A. (1998), Using Kohonen's self-organizing feature map to uncover automobile bodily injury claims fraud. The Journal of Risk and Insurance ; 65: 24

Changchien S.W., Lu T.C. (2001), Mining association rules procedure to support on-line recommendation by customers and products fragmentation. Expert Systems with Applications ; 20:

Chen T., Chen H. (1995), Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, Neural Networks ; 6:911-917.

Chen F.L., Liu S.F. (2000), A neural-network approach to recognize defect spatial pattern in semiconductor fabrication. IEEE Transactions on Semiconductor Manufacturing ; 13:366-37

Chen S.K., Mangiameli P., West D. (1995), The comparative ability of self-organizing neural networks to define cluster structure. Omega ; 23:271-279.

Chen H., Zhang Y., Houston A.L. (1998), Semantic indexing and searching using a Hopfield net. Journal of Information Science ; 24:3-18.

Cheng B., Titterington D. (1994), Neural networks: a review from a statistical perspective. Statistical Sciences ; 9:2-54.

Chen K.Y., Wang, C.H. (2007), Support vector regression with genetic algorithms in forecasting tourism demand. Tourism Management ; 28:215-226.

Chiang W.K., Zhang D., Zhou L. (2006), Predicting and explaining patronage behavior toward web and traditional stores using neural networks: a comparative analysis with logistic regression. Decision Support Systems ; 41:514-531.

Church K. B., Curram S. P. (1996), Forecasting consumers' expenditure: A comparison between econometric and neural network models. International Journal of Forecasting ; 12:255-267

Ciampi A., Lechevallier Y. (1997), Statistical models as building blocks of neural networks. Communications in Statistics: Theory and Methods ; 26:991-1009.

Crone S.F., Lessmann S., Stahlbock R. (2006), The impact of preprocessing on data mining: An evaluation of classifier sensitivity in direct marketing. European Journal of Operational Research ; 173:781-800

Cybenko G. (1989), Approximation by superpositions of a sigmoidal function. Mathematical Control Signals Systems ; 2:303–314.

Dai Y., Nakano Y. (1998), Recognition of facial images with low resolution using a Hopfield memory model. Pattern Recognition ; 31:159-167.

Dasu T., Johnson T. (2003), Exploratory Data Mining and Data Cleaning. New Jersey: Wiley, .

De Groot D., Wurtz D. (1991), Analysis of univariate time series with connectionist nets: A case study of two classical examples. Neurocomputing ;3:177-192.

Deboeck G., Kohonen T. (1998), Visual Explorations in Finance with Self-organizing Maps. London: Springer-Verlag, .

Delen D., Sharda R., Bessonov M. (2006), Identifying significant predictors of injury severity in traffic accidents using a series of artificial neural networks Accident Analysis and Prevention ; 38:434-444.

Dhar V., Chou D. (2001), A comparison of nonlinear methods for predicting earnings surprises and returns. IEEE Transactions on Neural Networks ; 12:907-921.

Dia H. (2001), An object-oriented neural network approach to short-term traffic forecasting. European Journal of Operation Research ; 131:253-261.

Dittenbach M., Rauber A., Merkl, D. (2002), Uncovering hierarchical structure in data using the growing hierarchical self-organizing map. Neurocompuing ; 48:199-216.

Doganis P., Alexandridis A., Patrinos P., Sarimveis H. (2006), Time series sales forecasting for short shelf-life food products based on artificial neural networks and evolutionary computing. Journal of Food Engineering ; 75:196-204.

Dutot A.L., Rynkiewicz J., Steiner F.E., Rude J. (2007), A 24-h forecast of ozone peaks and exceedance levels using neural classifiers and weather predictions Modelling and Software; 22:1261-1269.

Dutta S., Shenkar S. (1993), "Bond rating: a non-conservative application of neural networks." In Neural Networks in Finance and Investing, Trippi, R., and Turban, E., eds. Chicago: Probus Publishing Company.

Enke D., Thawornwong S. (2005), The use of data mining and neural networks for forecasting stock market returns. Expert Systems with Applications ; 29:927-940.

Evans O.V.D. (1997), Discovering associations in retail transactions using neural networks. ICL Systems Journal ; 12:73-88.

Fahlman S., Lebiere C. (1990), "The cascade-correlation learning architecture." In Advances in Neural Information Processing Systems, Touretzky, D., ed. .

Fletcher R. (1987), Practical Methods of Optimization 2^{nd}. Chichester: John Wiley & Sons, .

Frean M. (1990), The Upstart algorithm: a method for constructing and training feed-forward networks. Neural Computations ; 2:198-209.

Funahashi K. (1998), Multilayer neural networks and Bayes decision theory. Neural Networks ; 11:209-213.

Gallinari P., Thiria S., Badran R., Fogelman-Soulie, F. (1991), On the relationships between discriminant analysis and multilayer perceptrons. Neural Networks ; 4:349-360.

Geman S., Bienenstock E., Doursat T. (1992), Neural networks and the bias/variance dilemma. Neural Computation ; 5:1-58.

Gorr L. (1994), Research prospective on neural network forecasting. International Journal of Forecasting ; 10:1-4.

He H., Wang J., Graco W., Hawkins S. (1997), Application of neural networks to detection of medical fraud. Expert Systems with Applications ; 13:329-336.

Hebb D.O. (1949), The Organization of Behavior. New York: Wiley.

Hinton G.E. (1992), How neural networks learn from experience. Scientific American ;9:145-151.

Hornik K., Stinchcombe M., White H. (1989), Multilayer feedforward networks are universal approximators. Neural Networks ; 2:359–366.

Hopfield J.J. (2558), (1982), Neural networks and physical systems with emergent collective computational abilities. Proceedings of National Academy of Sciences; 79:2554-.

Hopfield J.J., Tank D.W. (1985), Neural computation of decisions in optimization problems. Biological Cybernetics ; 52:141-152.

Hu J.Q., Rose, E. (1995), On-line fuzzy modeling by data clustering using a neural network. Advances in Process Control. , 4, 187-194.

Huang J.S., Liu H.C. (2004), Object recognition using genetic algorithms with a Huang Z. Chen, H., Hsu, C.J. Chen, W.H. and Wu, S., Credit rating analysis with support vector machines and neural networks: a market comparative study. Decision Support Systems ; 37:543-558

Hopfield's neural model (1997). Expert Systems with Applications 1997; 13:191-199.

Jain L.C., Vemuri V.R. (1999), Industrial Applications of Neural Networks. Boca Raton: CRC Press, .

Kiang M.Y., Hu, M.Y., Fisher D.M. (2006), An extended self-organizing map network for market segmentation—a telecommunication example Decision Support Systems ; 42:36-47.

Kiang M.Y., Kulkarni U.R., Tam K.Y. (1995), Self-organizing map network as an interactive clustering tool-An application to group technology. Decision Support Systems ; 15:351-374.

Kim T., Kumara S.R.T., (1997), Boundary defect recognition using neural networks. International Journal of Production Research; 35:2397-2412.

Kim T.Y., Oh K.J., Sohn K., Hwang C. (2004), Usefulness of artificial neural networks for early warning system of economic crisis. Expert Systems with Applications ; 26:583-590.

Kirkos E., Spathis C., Manolopoulos Y., (2007), Data Mining techniques for the detection of fraudulent financial statements. Expert Systems with Applications ; 32: 995-1003.

Kiviluoto K. (1998), Predicting bankruptcy with the self-organizing map. Neurocomputing ; 21:203-224.

Klein B.D., Rossin D. F. (1999), Data quality in neural network models: effect of error rate and magnitude of error on predictive accuracy. Omega ; 27:569-582.

Kohonen T. (1982), Self-organized formation of topologically correct feature maps. Biological Cybernetics ; 43:59-69.

Kolehmainen M., Martikainen H., Ruuskanen J. (2001), Neural networks and periodic components used in air quality forecasting. Atmospheric Environment ; 35:815-825.

Law R. (2000), Back-propagation learning in improving the accuracy of neural network-based tourism demand forecasting. Tourism Management ; 21:331-340.

Lee D.L. (2002), Pattern sequence recognition using a time-varying Hopfield network. IEEE Transactions on Neural Networks ; 13:330-343.

Lewis O.M., Ware J.A., Jenkins D. (1997), A novel neural network technique for the valuation of residential property. Neural Computing and Applications ; 5:224-229.

Li W.J., Lee T., (2002), Object recognition and articulated object learning by accumulative Hopfield matching. Pattern Recognition; 35:1933-1948.

Lim G.S., Alder M., Hadingham P. (1992), Adaptive quadratic neural nets. Pattern Recognition Letters ; 13: 325-329.

Lisboa P.J.G., Edisbury B., Vellido A. (2000), Business Applications of Neural Networks : The State-of-the-art of Real-world Applications. River Edge: World Scientific, .

McCulloch W., Pitts W. (1943), A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics ; 5:115-133.

Min S.H., Lee J., Han I. (2006), Hybrid genetic algorithms and support vector machines for bankruptcy prediction. Expert Systems with Applications ; 31: 652-660.

Minsky M. L., Papert S. A. (1969), Perceptrons. MA: MIT press, .

Miyake S., Kanaya F. (1991), A neural network approach to a Bayesian statistical decision problem. IEEE Transactions on Neural Networks ; 2:538-540.

Mozer M.C., Wolniewics R. (2000), Predicting subscriber dissatisfaction and improving retention in the wireless telecommunication. IEEE Transactions on Neural Networks ; 11:690-696

Nag A.K., Mitra A. (2002), Forecasting daily foreign exchange rates using genetically optimized neural networks. Journal of Forecasting ; 21:501-512.

Nelson M., Hill T., Remus T., O'Connor, M. (1999), Time series forecasting using neural networks: Should the data be deseasonalized first? Journal of Forecasting ; 18:359-367.

O'Connor N., Madden M.G. (2006), A neural network approach to predicting stock exchange movements using external factors. Knowledge-Based Systems ; 19:371-378.

Paik J.K., Katsaggelos, A.K. (1992), Image restoration using a modified Hopfield neural network. IEEE Transactions on Image Processing ; 1:49-63.

Pajares G., Cruz J.M., Aranda, J. (1998), Relaxation by Hopfield network in stereo image matching. Pattern Recognition ; 31:561-574.

Panda C., Narasimhan V. (2007), Forecasting exchange rate better with artificial neural network. Journal of Policy Modeling ; 29:227-236.

Parker D.B. (1985), Learning-logic: Casting the cortex of the human brain in silicon, Technical Report TR-47, Center for Computational Research in Economics and Management Science, MIT.

Palmer A., Montaño J.J., Sesé, A. (2006), Designing an artificial neural network for forecasting tourism time series. Tourism Management ; 27: 781-790.

Partovi F.Y., Anandarajan M. (2002), Classifying inventory using an artificial neural network approach. Computers and Industrial Engineering ; 41:389-404.

Petersohn H. (1998), Assessment of cluster analysis and self-organizing maps. International Journal of Uncertainty Fuzziness and Knowledge-Based Systems. ; 6:139-149.

Prybutok V.R., Yi J., Mitchell D. (2000), Comparison of neural network models with ARIMA and regression models for prediction of Houston's daily maximum ozone concentrations. European Journal of Operational Research ; 122:31-40.

Qi M. (2001), Predicting US recessions with leading indicators via neural network models. International Journal of Forecasting ; 17:383-401.

Qi M., Zhang G.P. (2001), An investigation of model selection criteria for neural network time series forecasting. European Journal of Operational Research ; 132:666-680.

Qiao F., Yang H., Lam, W.H.K. (2001), Intelligent simulation and prediction of traffic flow dispersion. Transportation Research, Part B ; 35:843-863.

Raudys S. (1998), Evolution and generalization of a single neuron: I., Single-layer perceptron as seven statistical classifiers Neural Networks ; 11:283-296.

Raudys S. (1998), Evolution and generalization of a single neuron: II., Complexity of statistical classifiers and sample size considerations. Neural Networks ; 11:297-313.

Raviwongse R. Allada V., Sandidge T. (2000), Plastic manufacturing process selection methodology using self-organizing map (SOM)/fuzzy analysis. International Journal of Advanced Manufacturing Technology; 16:155-161.

Reed R. (1993), Pruning algorithms-a survey. IEEE Transactions on Neural Networks ; 4:740-747.

Remus W., O'Connor M. (2001), "Neural networks for time series forecasting." In Principles of Forecasting: A Handbook for Researchers and Practitioners, Armstrong, J. S. ed. Norwell:Kluwer Academic Publishers, 245-256.

Reutterer T., Natter M. (2000), Segmentation based competitive analysis with MULTICLUS and topology representing networks. Computers and Operations Research; 27:1227-1247.

Richard, M. (1991), D., Lippmann, R., Neural network classifiers estimate Bayesian *aposteriori* probabilities. Neural Computation ; 3:461-483.

Ripley A. (1993), "Statistical aspects of neural networks." In Networks and Chaos - Statistical and Probabilistic Aspects, Barndorff-Nielsen, O. E., Jensen J. L. and Kendall, W. S. eds. London: Chapman and Hall, 40-123.

Ripley A. (1994), Neural networks and related methods for classification. Journal of Royal Statistical Society, Series B ; 56:409-456.

Roh T. H. (2007), Forecasting the volatility of stock price index. Expert Systems with Applications ; 33:916-922.

Rosenblatt F. (1958), The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review ; 65:386-408.

Rout S., Srivastava, S.P., Majumdar, J. (1998), Multi-modal image segmentation using a modified Hopfield neural network. Pattern Recognition ; 31:743-750.

Rumelhart D.E., Hinton G.E., Williams R.J. (1986), "Learning internal representation by back-propagating errors." In Parallel Distributed Processing: Explorations in the Microstructure of Cognition Press, Rumelhart, D.E., McCleland, J.L. and the PDP Research Group, eds. MA: MIT.

Saad E.W., Prokhorov D.V., Wunsch, D.C. II. (1998), Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks. IEEE Transactions on Neural Networks; 9:456-1470.

Salcedo-Sanz S., Santiago-Mozos R.,Bousono-Calzon, C. (2004), A hybrid Hopfield network-simulated annealing approach for frequency assignment in satellite communications systems. IEEE Transactions on System, Man and Cybernetics, Part B:108-116.

Sarle W.S. (1994), Neural networks and statistical models. Poceedings of the Nineteenth Annual SAS Users Group International Conference, Cary, NC: SAS Institute, .

Schumacher M., Robner R., Vach W. (1996), Neural networks and logistic regression: Part I., Computational Statistics and Data Analysis ; 21:661-682.

Smith K.A., Ng, A. (2003), Web page clustering using a self-organizing map of user navigation patterns. Decision Support Systems ; 35:245-256.

Smith K.A., Willis R.J., Brooks M. (2000), An analysis of customer retention and insurance claim patterns using data mining: a case study. Journal of the Operational Research Society; 51:532-541.

Soulie F.F., Gallinari P. (1998), Industrial Applications of Neural Networks. River Edge, NJ: World Scientific.

Suganthan P.N., Teoh E.K., Mital D.P. (1995), Self-organizing Hopfield network for attributed relational graph matching. Image and Vision Computing; 13:61-71.

Sun Z.Z., Yu S. (1995), Improvement on performance of modified Hopfield neural network for image restoration. IEEE Transactions on Image processing; 4:683-692.

Suykens J.A.K., Vandewalle J.P.L., De Moor B.L.R. (1996), Artificial Neural Networks for Modeling and Control of Nonlinear Systems. Boston: Kluwer.

Swanson N.R., White H. (1995), A model-selection approach to assessing the information in the term structure using linear models and artificial neural networks. Journal of Business and Economic Statistics; 13;265-275.

Tatem A.J., Lewis H.G., Atkinson P.M., Nixon M.S. (2002), Supre-resolution land cover pattern prediction using a Hopfield neural network. Remote Sensing of Environment; 79:1-14.

Temponi C., Kuo Y.F., Corley H.W. (1999), A fuzzy neural architecture for customer satisfaction assessment. Journal of Intelligent & Fuzzy Systems; 7:173-183.

Thieme R.J., Song M., Calantone R.J. (2000), Artificial neural network decision support systems for new product developement project selection. Journal of Marketing Research ; 37:543-558.

Vach W., Robner R., Schumacher M. (1996), Neural networks and logistic regression: Part I. Computational Statistics and Data Analysis; 21:683-701.

Wang T., Zhuang X., Xing X. (1992), Robust segmentation of noisy images using a neural network model. Image Vision Computing; 10:233-240.

Webb A.R., Lowe D., (1990), The optimized internal representation of multilayer classifier networks performs nonlinear discriminant analysis. Neural Networks; 3:367-375.

Werbos P.J., (1974), Beyond regression: New tools for prediction and analysis in the behavioral sciences. Ph.D. thesis, Harvard University, 1974.

West D., (2000), Neural network credit scoring models. Computers and Operations Research; 27:1131-1152.

West P.M., Brockett P.L., Golden L.L., (1997), A comparative analysis of neural networks and statistical methods for predicting consumer choice. Marketing Science; 16:370-391.

Widrow B., Hoff M.E., (1960), Adaptive switching circuits, 1960 IRE WESCON Convention Record, New York: IRE Part 4 1960:96-104.

Widrow B., Rumelhart D.E., Lehr M.A., (1994), Neural networks: applications in industry, business and science, Communications of the ACM; 37:93-105.

Wong B.K., Bodnovich T.A., Selvi Y., (1997), Neural network applications in business: A review and analysis of the literature (1988-1995). Decision Support Systems; 19:301-320.

Young S.S., Scott P.D., Nasrabadi, N.M., (1997), Object recognition using multilayer Hopfield neural network. IEEE Transactions on Image Processing; 6:357-372.

Zhang G.P., (2007), Avoiding Pitfalls in Neural Network Research. IEEE Transactions on Systems, Man, and Cybernetics; 37:3-16.

Zhang G.P., Hu M.Y., Patuwo B.E., Indro D.C., (1999), Artificial neural networks in bankruptcy prediction: general framework and cross-validation analysis. European Journal of Operational Research; 116:16-32.

Zhang G.P., Keil M., Rai A., Mann J., (2003), Predicting information technology project escalation: a neural network approach. European Journal of Operational Research 2003; 146:115–129.

Zhang G.P., Qi M. (2002), “Predicting consumer retail sales using neural networks.” In Neural Networks in Business: Techniques and Applications, Smith, K. and Gupta, J.eds. Hershey: Idea Group Publishing, 26-40.

Zhang G.P., Patuwo E.P., Hu M.Y., (1998), Forecasting with artificial neural networks: the state of the art. International Journal of Forecasting; 14:35-62.

Zhang W., Cao Q., Schniederjans M.J., (2004), Neural Network Earnings per Share Forecasting Models: A Comparative Analysis of Alternative Methods. Decision Sciences; 35: 205–237.

Zhu Z., He H., Starzyk J.A., Tseng, C., (2007), Self-organizing learning array and its application to economic and financial problems. Information Sciences; 177:1180-1192.

Improved SOM Labeling Methodology for Data Mining Applications

Arnulfo Azcarraga[1], Ming-Huei Hsieh[2], Shan-Ling Pan[3], and Rudy Setiono[4]

[1] College of Computer Studies, De La Salle University, Manila, the Philippines
azcarragaa@canlubang.dlsu.edu.ph
[2] Department of International Business, National Taiwan University, Taiwan
mhhsieh@management.ntu.edu.tw
[3] School of Computing, National University of Singapore, Singapore
pansl@comp.nus.edu.sg
[4] School of Computing, National University of Singapore, Singapore
rudys@comp.nus.edu.sg

Summary. Self-Organizing Maps (SOMs) have been useful in gaining insights about the information content of large volumes of data in various data mining applications. As a special form of neural networks, they have been attractive as a data mining tool because they are able to extract information from data even with very little user-intervention. However, although learning in self-organizing maps is considered unsupervised because training patterns do not need desired output information to be supplied by the user, a trained SOM often has to be labeled prior to use in many real-world applications. Unfortunately, this labeling phase is usually supervised as patterns need accompanying output information that have to be supplied by the user. Because labeled patterns are not always available or may not even be possible to construct, the supervised nature of the labeling phase restricts the deployment of SOM to a wider range of potential data mining applications. This work proposes a methodical and semi-automatic SOM labeling procedure that does not require a set of labeled patterns. Instead, nodes in the trained map are clustered and subsets of training patterns associated to each of the clustered nodes are identified. Salient dimensions per node cluster, that constitute the basis for labeling each node in the map, are then identified. The effectiveness of the method is demonstrated on a data mining application involving customer-profiling based on an international market segmentation study.

Key words: self-organizing maps,neural networks, classification, clustering

1 Introduction

In many data mining applications, there is limited knowledge about what might be contained in the input data, and very often, the dataset has no veritable structure that will allow for easy searching of information. As such, neural networks are important tools for data mining applications because they are able to learn just from being shown examples of the data, without explicitly being told what to look for, or how the information is structured in the input data.

Neural network models are often categorized as either "supervised" or "unsupervised", based on whether or not the learning method they employ requires some supplementary "desired output" information to accompany each of the training patterns. The most well-known supervised neural network model is the multi-layered perceptron (MLP), with retro-propagation of error as the underlying learning mechanism (Haykin, 1998, Rumelhart *et al.*, 1986). In such systems, network parameters are adjusted during training based on the difference (i.e. error) between the system's response for a given training pattern and the desired output for this pattern.

Among the unsupervised neural network models, the most popular are the Kohonen maps or Self-Organizing Maps (SOMs) (Kohonen, 1995, Kohonen, 1999). In the class of unsupervised neural network models, the underlying learning mechanism is Competitive Learning, the general framework for which is described in (Rumelhart and Zipser, 1986). Other models in this class include the Adaptive Resonance Theory (ART) (Carpenter and Grossberg, 1991) and the Neocognitron (Fukushima, 1980). All these models do not require any supplementary information other than the input data. The input space is partitioned into non-overlapping regions which are delineated based on a process of competition among the nodes in the neural network. For every training pattern that is shown to the network, the single node that is closest to the input pattern, based on some distance measure, earns the right to assimilate the training pattern by adapting its weights as well as the weights of those nodes in its neighborhood.

The SOM methodology dates back to the early 80's (Kohonen, 1982) and has been applied to a wide variety of applications (Kohonen, 1990), which include data mining (Kiang and Kumar, 2001), marketing (Schmitt and Deboeck, 1998, Kuo *et al.*, 2002), investment banking (Kiviluto and Bergius, 1998, Shumsky and Yarovoy, 1998); speech processing (Kohonen, 1990), robotics (Ritter *et al.*, 1992), finance engineering (Serrano-Cinca, 1998, Deboeck, 1998a, Deboeck, 1998b, Resta, 1998), text organization and retrieval (Kohonen *et al.*, 2000, Merkl, 1998), and real-estate applications (Carlson, 1998, Tulkki, 1998).

Prior to its use in some real-world application, a trained SOM has to be labeled - typically using labeling patterns that have accompanying category information. In a bankruptcy analysis application, for example, a set of known cases is used to determine which nodes are sensitive to profiles of companies that have financial difficulties and which nodes are sensitive to profiles of companies that are solvent (Serrano-Cinca, 1998). On the basis of these node labels, a new company is evaluated in terms of its chances of going bankrupt based on its profile. In such a typical SOM application, even if training is unsupervised, the labeling phase is supervised.

Because such labeled patterns are not always available or may not even be possible to construct, this supervised labeling phase of the SOM methodology hinders the deployment of SOM to an even wider range of potential domains of application. Take for example the case of a market survey of potential consumers, with questions pertaining to what consumers look for from a specific line of products. After the set of respondent records is fed to a SOM for training, a separate set of labeled consumer records would be needed to associate each node with some tangible purchasing pattern or behavior. For example, if we want the SOM to assist in identifying which types of responses correspond to what kinds of purchasing behavior, it would be convenient to have access to a set of respondent records with an indication of the type of product that they have actually purchased. With such “labeled” respondent records, we can check each node in the map to see which kinds of respondent records are associated with it, and to label them accordingly. This would then allow the user to do various post-processing tasks, like analyzing the socio-demographic profile of each cluster of nodes (i.e. each cluster of associated respondent records). In turn, this makes for clusters that are identifiable and actionable, with the presumption that clusters become meaningful only when they can be identified and can be acted upon (Wedel and Kamakura, 1998).

Indeed, several special-cases of self-organizing maps that have been deployed for very specific tasks are designed such that the labeling of the nodes does not require a set of pre-labeled patterns. This is the case for SOMs used in text-processing and classification (Merkl, 1998,Kohonen *et al.*, 2000,Azcarraga *et al.*, 2002). In these systems, the words associated with certain dimensions in the weight vectors of the trained SOM are used to label the nodes.

Building on the ideas underlying such systems, we improve on the SOM methodology by proposing a methodical and automatic SOM labeling procedure that does not require any set of labeled patterns. The proposed method is quite general. It can be applied to numerous other areas where self-organizing maps are being employed. For illustration purposes, however, we shall focus the discussions on a SOM-based international market segmentation study.

The rest of the chapter is organized as follows. Section 2 describes in greater detail the SOM methodology and includes some of the most frequently used training and labeling methods. Section 3 describes the unsupervised SOM labeling method that we propose. The application of this labeling method to a SOM-based international market segmentation study is discussed in section 4. Section 5 introduces some non-SOM techniques to validate the results. This is followed by the conclusions and recommendations for future work in section 6.

2 SOM methodology

The SOM methodology is depicted in Figure 1. Pre-processing would typically include dealing with missing data and normalizing input fields within a consistent input range, e.g. 0 to 1. Various other pre-processing steps are done depending on the specific domain of application. When preparing documents as input patterns in text processing and classification, for example, common words known as “stop words” are removed to reduce the dimensionality of the input space. Furthermore, words are reduced to their root form, through a process called “stemming”. In the digital archiving of music, where SOMs may be used as an innovative interface for searching through and surfing over a large archive of music files, music files are pre-processed to extract various “features” that characterize the beat, rhythm, timbre, and higher level characteristics such as music genre, vocal quality, types of instruments used, etc. (Mayer *et al.*, 2006).

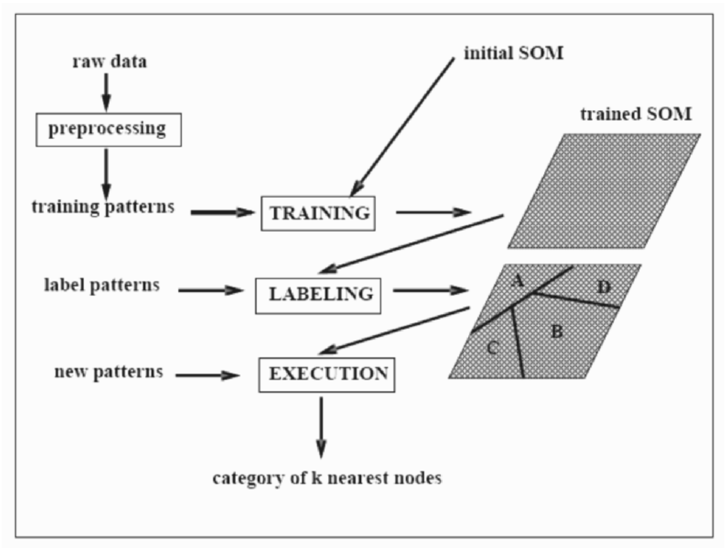


Fig. 1. A general SOM methodology

Once the raw data have been adequately represented in the input space, SOM training is performed. In a SOM system, a map is usually a rectangular grid of nodes, although some SOMs use hexagonal grid structures (Kohonen, 1999,Kohonen, 1990). All input units are connected to each node in the map, and the connection from each input unit i to a node j is represented by a connection weight w_{ij}. Each input unit corresponds to one input field, and typically, all input units draw their values from a binary set (0 or 1), bipolar set (-1 or +1), or from a uniform range of real values (e.g. 0 to 1). The set of values assumed by the individual input units at a certain training cycle

t is denoted by an input vector x^t, with x_i^t referring to the specific value of input unit i at cycle t. Training of the map consists of successively presenting input patterns through the input units and of adapting the various connection weights of each node in the map. At each training cycle t, one training sample x_t is selected at random. Each node then computes its distance/similarity to the current input, using some appropriate distance or similarity measure (e.g. Euclidean distance or cosine of angle between input and node weight vector). The weights w_{ij}^t of all nodes n_i in the neighborhood of the node with the smallest distance (the winning node n_c) are then updated using the following learning rule (Clark and Ravishankar, 1990):

$$w_{ij}^{t+1} = w_{ij}^t + \alpha(t)(x_j^t - w_{ij}^t) \tag{1}$$

The gain parameter $\alpha(t)$ and the size of the neighborhood decrease with the number of cycles, according to some parameter adjustment function (Ritter *et al.*, 1992).

The more recent version of the training algorithm does away with the neighborhood region. In its place, a gaussian function $G(c, i, t)$ is used so that nodes near the winning node n_c have larger weight changes than those further away (Kohonen, 1999):

$$w_{ij}^{t+1} = w_{ij}^t + \alpha(t)G(c, i, t)(x_j^t - w_{ij}^t) \tag{2}$$

The function $G(c, i, t)$ is defined by the formula below, where $\sigma(t)$ is a parameter to control the size of the neighborhood of nodes that would have substantial weight changes, and $D(c, i)$ is the grid distance between a given node n_i and the winning node:

$$G(c, i, t) = \exp\left(\frac{-D(c, i)^2}{\sigma(t)^2}\right) \tag{3}$$

By the end of the training phase, a self-organized map would have emerged. This map is often not useful until each node in the map is labeled. Some SOM applications, however, make do with just visualizing the individual component planes of the map. This is done by rendering the weight of each node's reference vector in 2D as shown in Figure 2. In the figure, each plane corresponds to a certain car feature. These car features are the following: fun to drive, acceleration and speed, dealer service, fuel economy, styling, level of technology, luxury features, made to last, prestige, reliability, safety in accidents, sportiness, quality, passenger space, cargo/luggage space.

A total of 2,385 potential customers have been asked to select which three of the 15 car features above are most important to them. Based on the survey responses, a 16 × 16 SOM was trained and the weights on a per dimension basis are shown. In the SOM literature, these are referred to as "component planes". Each plane has 16 × 16 smaller squares, with each square representing a node in the 16 × 16 map. The grey level of each square denotes the weight value of the node for the given dimension. Black squares correspond to zero

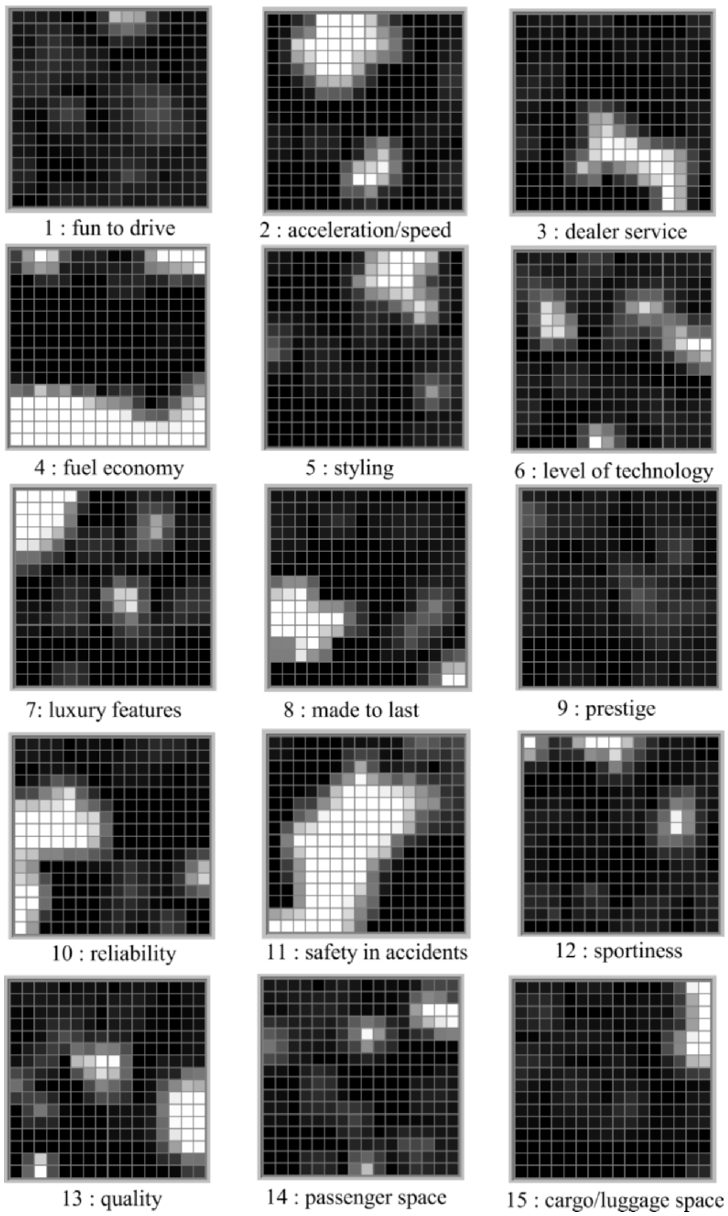


Fig. 2. Component planes for the trained 16×16 SOM, with one component plane for each dimension. Each dimension corresponds to a car feature that potential car buyers look for when buying a car. Lighter shades of grey indicate higher weight values for the given dimension.

or near zero weight values, while white squares denote higher weight values. A high weight value of a node for a given dimension (i.e. car feature) indicates that most of the respondents associated to the node have selected the given car feature as one of three car features they value most.

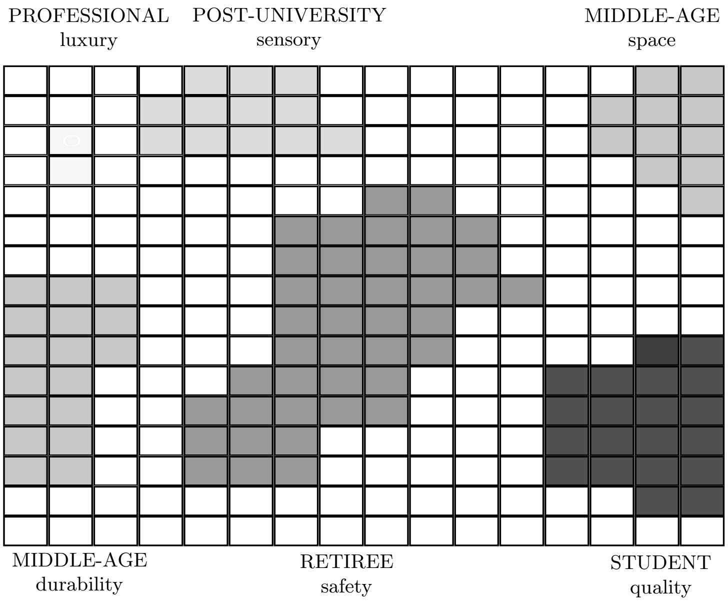


Fig. 3. A hypothetical SOM with clusters of nodes that have been labeled according to the age bracket to which most of its associated label patterns belong.

Deboeck and Kohonen (DeBoeck and Kohonen, 1998) present numerous examples on how component planes are visualized. Basically, the user takes note of the distribution patterns in the map of the weight values for every dimension (or component) and interprets them based on their relation to the clusters observed. To illustrate, suppose that the component planes of Figure 2 correspond to a trained SOM (refer to Figure 3) that has been labeled according to the age brackets of the majority of the associated respondent records. The nodes of the trained SOM have been labeled as: college student (age 18-22), post-university (23-30), early-professional (31-40), mid-age (41-60), and retiree (61-75). The aim of this labeling scheme would be to understand what types of car features attract customers of certain age groups[5].

[5] The labeling scheme of Figure 3 is fictitious aimed only at illustrating the use of component planes in relation to labeled maps. However, the component planes of Figure 2 are actual component planes generated for the market segmentation study discussed in section 4.

In this example, the user visually inspects the component planes and notes, for example, that component plane 11 reveals high preference for "safety in accidents" as a car feature among the retiree age-group, as manifested by the patch of white squares corresponding to those nodes associated with the retiree group in the mid-section of the map. Likewise, the young out-of-college age group ("post university" in Figure 3) is attracted to sensory car benefits like "acceleration and speed" and "sportiness" whose component planes show white patches at around the area of nodes that corresponds to the post-university cluster in Figure 3. Respondents in the early-professional age group are attracted to "luxury features", while college students go for "quality" and "dealer service". As for the mid-age customers, some are attracted to durability features like "reliability" and "made to last", while others go for space features like "cargo/luggage space" and "passenger space".

As we will see in the next section, the automatic labeling method proposed in this paper is a methodical and statistics-based version of the way component planes are studied and visualized. Before we present the unsupervised labeling method, we first go through some of the existing ways by which trained maps can be labeled.

The usual way is to employ a separate set of label patterns which have been individually pre-classified into designated categories. To label a given node in the map, its distance to each of the label patterns is computed. Based on the categories of the label pattern(s) to which the node yields the smallest distance(s), the given node is assigned a label. This labeling method is applied to all the nodes in the map.

Several options are available for determining the final label(s) of a node, given its distance d_i to each of the label patterns p_i. If the labels are categorical (e.g. "vowel a" for a phonetic map, "terrorism" for news classification), then given the set β of k label patterns which yield the k smallest distances to node j, some of the labeling methods for assigning a label c to the given node j are as follows:

1. c is any label among patterns in β. If $k = 1$, then c is just the label of pattern p_m that yields the lowest distance d_m;
2. c is the most common label among patterns in β;
3. c is any label appearing at least r times among patterns in β where r is a pre-set percentage of k; and
4. c is a label determined as follows: rank the patterns in β from the nearest to the farthest from node j, and assign them with weights $(k - r + 1)$, where r is their rank. The sum of weights for each label is tallied and the label with the highest total weight is chosen.

There are many variants to these labeling methods. Note that methods 2 and 4 yield a single label for each node, while methods 1 and 3 can assign multiple labels. Furthermore, if labels are assigned continuous values (e.g. "age": 22.3, "grade point average": 3.57), then method 2 is modified by computing the average label instead of choosing the most common label, while method 4 is modified by taking the weighted average of the labels.

Once the map is fully labeled, it is ready for use. Although it varies depending on the task a SOM is supposed to support, one frequent role of the SOM is to assist in the classification of some unknown pattern which has the same input fields as those of the training patterns. Classification is done by computing the distances of the unknown pattern with respect to each of the nodes in the map. The relative distances of the nodes plus their associated categories (labels) would then serve as basis for classifying the unknown pattern. The final classification of the unknown pattern can be as simple as just assigning the category of the node that registers the smallest distance to the unknown pattern.

Depending on the application, the process can be more complicated than a simple classification of an unknown pattern. In some applications, the distances and categories of a whole region of nodes (sometimes the entire map) that are nearest to the unknown pattern are fed to another classification system, along with the associated distances of each node. Or, as in the case of the phonetic map, for example, the trajectory of the nearest nodes while phonemes are fed to the SOM one after another is the basis for segmenting the speech data and for recognizing the spoken words (Kohonen, 1990). In Serrano-Cinca's work (Serrano-Cinca, 1998), this same type of trajectory of nearest nodes is the basis for determining the general liquidity (solvency) and financial health of banks.

3 Unsupervised SOM labeling

We propose a general method for labeling self-organizing maps that does away with pre-labeled patterns. Indeed, in many applications, pre-labeled patterns are not easy to obtain. In fact, if pre-labeled patterns are available, then the supervised neural network models could have been used instead. Unsupervised neural network models are attractive options because they do not require training patterns to be accompanied with the desired classification outcome. But if these unsupervised models would require labeled patterns for labeling the resultant neural network, then the applicability of these models becomes limited. Such is the drawback of self-organizing maps that we are able to fix with a novel labeling method that does not require pre-labeled patterns - not during training, and not even during labeling.

At the onset, we clarify that this unsupervised SOM labeling method is not applicable to every conceivable SOM application. For example, this proposed method may not be used in some applications in image processing, where the input dimensions refer to identical input features (i.e. light intensity) for pixels in different locations of the image. Nor would the method be applicable to some applications in speech processing, where all dimensions might refer to the amplitude at different frequencies of a set of voice signals. Since we use the dimensions as basis for differentiating the clusters, the physical interpretation of each individual dimension must then be distinguishable from each other.

The unsupervised labeling method we describe here would be useful in most other types of applications where each input dimension corresponds to a tangible feature that has some concrete meaning in the application domain. In text processing for example, each input dimension may correspond to some unique word or phrase (bi-grams or tri-grams). In various finance engineering applications, each input dimension may refer to some country-specific macro-economic variable like GNP or inflation rate. Or the input dimensions may correspond to normalized values of company-specific factors such as price-equity ratio, stock price, and market capitalization. In market segmentation studies, input dimensions may correspond to scaled responses to market survey questions such as whether a customer values prompt waiter service or whether a consumer prefers cars that have large passenger space, for example.

The general idea of our proposed unsupervised SOM labeling method consists of five main steps (refer to Figure 3), each of which would be elaborated further below:

1. group all nodes that have similar reference weight vectors using some clustering method;
2. for each cluster of nodes, prune out outlier nodes that are very different from their cluster centroids;
3. for each cluster of nodes (minus the outliers), classify the set of (unlabeled) training patterns as either in-patterns or out-patterns depending on whether or not their nearest node in the map is in the cluster;
4. based on the set of in-patterns and out-patterns of a given cluster, identify the salient dimensions; and
5. on the basis of the salient dimensions, assign a descriptive label to each cluster of nodes that is meaningful in the context of the application domain.

3.1 Step 1: Clustering of node weight vectors

In grouping the nodes into clusters of similar reference weight vectors, one major problem is determining the appropriate number of clusters. This can be resolved by doing a hierarchical clustering of the weight vectors. Various hierarchical methods can be used and these are discussed in standard clustering textbooks (Everitt, 1974,Hartigan, 1975,Spath, 1980). A recent survey was done by Xu and Wunsch (Xu and Wunsch, 2005).

There are basically two types of hierarchical methods: agglomerative and divisive. In agglomerative methods, each node weight vector starts off as individual clusters. At every step, the two most similar clusters are merged into a single cluster and the new cluster center, or centroid, is computed. The merging of clusters continues until the quality of the clusters is satisfactory, that is, no two distinct clusters may be merged and result in a significant increase in the quality of the groupings, according to some measure of clustering quality.

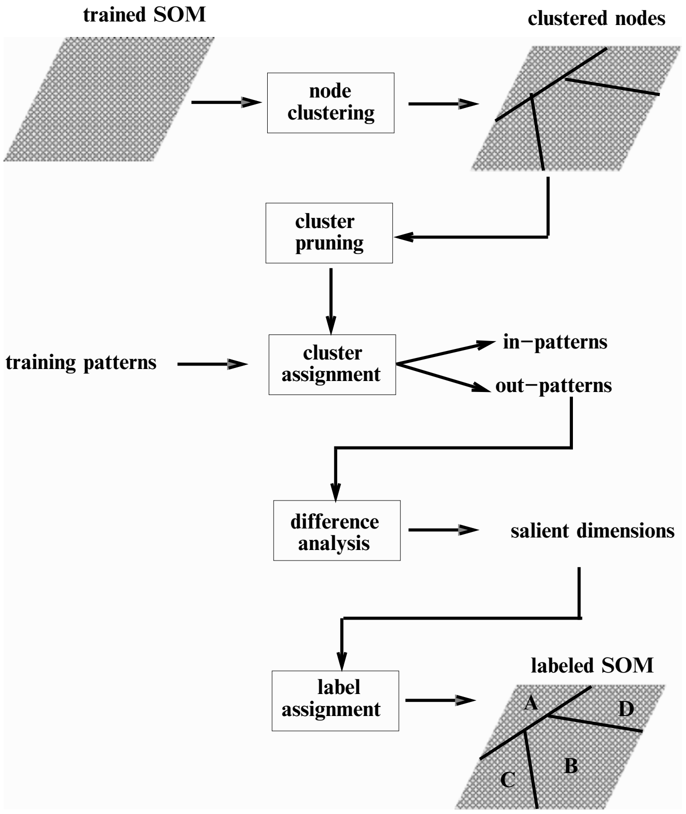


Fig. 4. General unsupervised SOM labeling method.

In divisive methods, all the node weight vectors start off as a single cluster. At each cycle, a partitioning system selects the cluster with the highest variance and breaks it up into two. Weight vectors in the cluster that has just been split are redistributed to the two new clusters. Again, the breaking up of clusters continues until the quality of the groupings has become satisfactory, according to some clustering quality measure.

The quality (usually based on the within-cluster variance) of the resultant groupings is computed every time two clusters are merged in the case of agglomerative methods, or whenever a cluster is split into two in the case of divisive methods. For agglomerative methods, a relatively large increase in the variance among the patterns within a cluster is a good indicator that the two clusters are quite distinct and should not be merged. When this happens, the hierarchical clustering may be stopped at this point, and the number of remaining clusters is a good estimate of the suitable number of clusters to be specified for the k-means clustering to be done later. For divisive methods, the splitting stops when none of the clusters can be split to gain a significant decrease in variance among patterns in the clusters.

Once the suitable number of clusters is determined by either an agglomerative or divisive method, we can proceed with k-means clustering of the SOM node weight vectors. Many variants of k-means exist, and there is abundant literature on the subject (Everitt, 1974, Hartigan, 1975, Spath, 1980). This form of clustering frequently derives better quality clusters, since it is more robust to poor quality data, and is less affected by outliers. Its main drawback is that the number of clusters k has to be known a priori, which is why hierarchical clustering is performed first (Punj and Steward, 1983).

If there is a simpler way of determining a suitable value for k, then hierarchical clustering may be omitted. In fact, in most SOM applications, the number of clusters that can be visually inspected and analyzed has to be small. In such cases, the user may just opt to perform k-means clustering using different values of k, say from 3 to 10, and select the k value that produces the best clustering results. Since labeling is unsupervised and automatic, this is feasible to do.

3.2 Step 2: Pruning of outlier nodes

We next try to remove some nodes from their clusters if they are too different from the cluster centers. To do this, we compute the centroid χ^k of each node cluster Γ^k as follows :

$$\chi_j^k = \frac{\sum\limits_{n_i \in \Gamma^k} w_{ij}}{|\Gamma^k|}, \quad j = 1, 2, \ldots, D \tag{4}$$

where D is the dimensionality of the data and w_{ij} is the jth component of the reference weight vector of node n_i, one of the nodes in Γ^k. The function $|A|$ returns the cardinality of set A. We then compute the distance d_i from each node n_i in Γ^k to its centroid χ^k as :

$$d_i = \sqrt{\sum_{j=1}^{D} \left(w_{ij} - \chi_j^k\right)^2} \tag{5}$$

With the individual node distances to their respective cluster centroids, we are ready to compute the mean μ_d^k of these distances and the standard deviation σ_d^k for each cluster of nodes. Using $z = 1$, we retain a node n_i in its original cluster if

$$\mu_d^k - z \times \sigma_d^k < d_i < \mu_d^k + z \times \sigma_d^k \tag{6}$$

Those nodes with distance from the centroid that differ from the mean by more than one standard deviation are considered *outliers* and are excluded from the cluster. All these nodes that have been dropped from their original clusters can be collectively referred to as "unlabeled nodes". In some applications, this special set of nodes may have some concrete role to play.

3.3 Step 3: Separating in-patterns and out-patterns

Once each node in the map is assigned to a given cluster, the individual training patterns are re-used. Each of these patterns is assigned to the cluster of the node to which the distance is smallest. On the basis of these training pattern assignments, we construct an *in-patterns* set and an *out-patterns* set for each cluster. The in-patterns are those patterns belonging to the given cluster, while the out-patterns are all the other patterns not belonging to the cluster, including those patterns associated with "unlabeled nodes" if any.

3.4 Step 4: Identifying salient dimensions

The next step is to identify those dimensions in a given cluster, referred to as salient dimensions, whose values are significantly different in a statistical sense compared to those in the other clusters. For each cluster, we determine if the mean input value among the in-patterns for a given dimension is significantly higher or lower than the corresponding mean input value among the out-patterns. To identify salient dimensions of each cluster Γ^k, we do the following

1. for each dimension v, compute $\mu_{in}(k,v)$ and $\mu_{out}(k,v)$ as the mean input value for the set of in-patterns $\Phi_{in}(k)$ and out-patterns $\Phi_{out}(k)$, respectively, where p_i is training pattern i and x_{iv} is the v-th component of the input vector of p_i:

$$\mu_{in}(k,v) = \frac{\sum\limits_{p_i \in \Phi_{in}(k)} x_{iv}}{|\Phi_{in}(k)|} \tag{7}$$

$$\mu_{out}(k,v) = \frac{\sum\limits_{p_i \in \Phi_{out}(k)} x_{iv}}{|\Phi_{out}(k)|} \tag{8}$$

2. compute the *difference factor* $df(k,v)$ of each dimension v as

$$df(k,v) = \frac{\mu_{in}(k,v) - \mu_{out}(k,v)}{\mu_{out}(k,v)} \tag{9}$$

3. compute the difference factors mean $\mu_{df}(k)$ and standard deviation $\sigma_{df}(k)$ over all dimensions v; to avoid possible mix-up in the indices, we give the formula for the mean and standard deviation as follows:

$$\mu_{df}(k) = \frac{\sum_{v=1}^{D} df(k,v)}{D} \tag{10}$$

$$\sigma_{df}(k) = \left(\sum_{v=1}^{D} \left(df(k,v) - \mu_{df}(k,v) \right)^2 / D \right)^{\frac{1}{2}} \tag{11}$$

Using Equations 12, 13 and 14, we are ready to precisely define a salient dimension. A dimension v is a salient dimension for cluster Γ^k if its corresponding difference factor deviates from the mean by at least one standard deviation; that is if

$$df(k,v) \leq \mu_{df}(k) - z \times \sigma_{df}(k) \tag{12}$$

or

$$df(k,v) \geq \mu_{df}(k) + z \times \sigma_{df}(k) \tag{13}$$

with $z = 1$.

3.5 Step 5: Assignment of descriptive labels

Once the salient dimensions are identified for each cluster of nodes, we manually interpret the label combinations and assign domain-specific descriptions as final cluster labels. This step is inherently supervised in most domains of applications. A user has to inspect the salient dimensions and must provide the required descriptive labels. It is interesting to note that in text processing applications, even this final step can be unsupervised, since the words that correspond to the salient dimensions can in fact take the place of a "descriptive label". Note that the association between dimension and words is done automatically during the pre-processing stage.

In cases where there is more than one significant dimension for a given cluster, the absolute value of the difference factors is used to rank the salient dimensions, which would then aid in deciding on an appropriate descriptive label for the cluster. It should also be noted that in some applications, only dimensions that have positive difference factors (i.e. $\mu_{in}(k,v) > \mu_{out}(k,v)$) are meaningful. In such cases, the user may just ignore the negative difference factors when choosing appropriate labels for the cluster. Examples will be given below to illustrate this.

On the issue of input representation, we explained at the onset, that we are assuming dimensions are distinct and they individually represent some tangible feature in the application domain. Note, however, that even if this is not the case, our method would be able to detect the salient dimensions. For some applications, this is all that is needed (i.e. descriptive labels are not necessary).

Another point regarding input representation is that our method is sensitive to "high values" and "low values" when evaluating each dimension for purposes of spotting salient dimensions. Therefore, if the domain of application is such that several ranges of values may have important connotations, then assigning one dimension to each range of values is a better encoding scheme. In the case of age or income brackets, for example, representing each range of values as a separate dimension would allow for labels to be deduced for the specific age or income bracket. Otherwise, if data are entered as normalized values within a certain range (e.g. 0 to 1), then the label will only be in terms of high and low age or income, and will not pertain to specific income or age brackets.

4 Customer profiling: an illustration

Being visual renderings of the input set, self-organized maps open up opportunities for gaining insights and mining critical information from an otherwise unstructured set of data. We illustrate here how a SOM is labeled with the appropriate labels and, once labeled, is used to do an automatic profiling of potential car buyers. The user of the SOM results could, for example, design detailed marketing strategies for very specific niche markets, given the features and qualities that the specific market is attracted to.

We trained a SOM using data collected by MORPACE International in a cross-national consumer survey. The data set covers the top twenty automobile markets in the world consisting of 4,320 eligible new vehicle buyers who bought a car within the past six months during the period September-October 1997. Although the dataset consists of 4,332 samples, only respondents who had purchased or intended to purchase a passenger car were selected for analysis. Furthermore, Chinese-, Russian-, Turkish-, and Indian samples were removed from the dataset for analysis due to the relatively modest qualified sample sizes in those countries. Consequently, a total of 2,385 respondent records from 16 countries were included in the study.

In the survey, automobile benefit-sought behavior was measured by asking respondents to choose up to three benefits (out of 15) that they considered as most important benefits when purchasing a new car. The list of benefits includes "fun to drive", "good acceleration and speed", "good dealer service", "good fuel economy", "good styling", "level of technology", "luxury features", "made to last", "prestige", "reliability", "safety in accidents", "sportiness", "high quality", "passenger space", and "cargo/luggage space".

According to a study (Hsieh, 2002), the dimensionality of the benefits listed above corresponds approximately to the brand concepts[6] proposed by

[6] Brand concepts are defined as brand-unique abstract meanings that typically originate from a particular configuration of product features and a firm's efforts to create meaning from these arrangements (Park *et al.*, 1991).

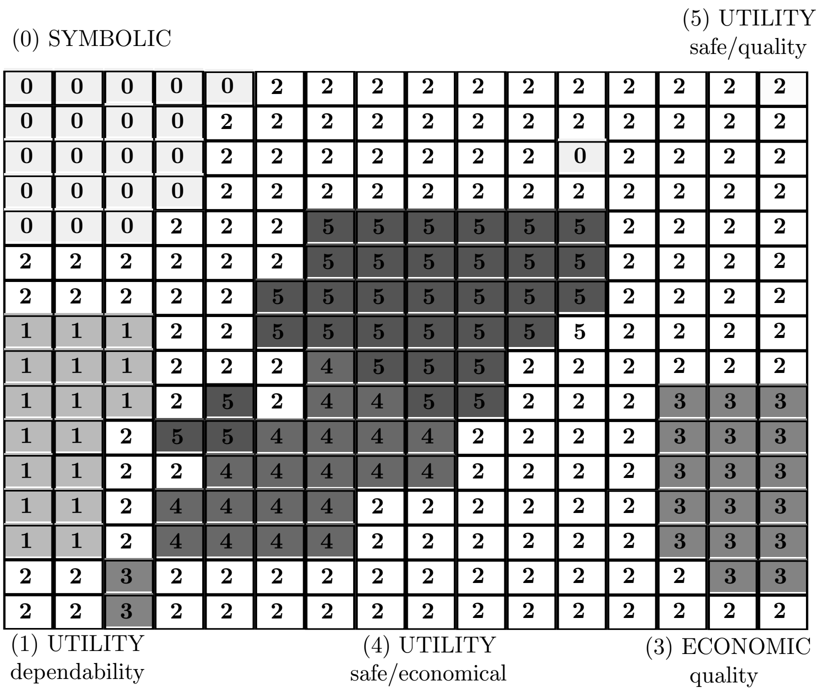
Park, Jaworski and MacInnis (Park *et al.*, 1986). The four dimensions extracted are: (1) the symbolic dimension including prestige, luxury features, styling and quality, (2) the sensory dimension including good acceleration and speed, fun to drive, and sportiness (3) the utilitarian dimension including reliability, durability and safety in accidents, and (4) the economic dimension consisting of fuel economy and dealer service. The relationship between functional/utilitarian, social/symbolic and experiential/sensory needs and consumption has been proven to be significant in various studies (Holbrook and Schindler, 1994). The set of benefits covering the three universal needs along with specific product benefits serve as a rather comprehensive set of benefits that consumers are likely to be seeking.

A 16 × 16 SOM was trained using the converted binary data from the global samples. By doing an agglomerative hierarchical clustering on the data, we observed that when the number of clusters was fewer than six, relatively distinct clusters were being merged (big increase in inter-cluster variance). But there was really no specific number of clusters that was evidently ideal, so we proceeded with doing k-means clustering using different values of k. We did not probe more than eight clusters since, from a market segmentation point of view, a large number of clusters would be counter-productive. Of the clusters generated, we zeroed in on the initial 6-cluster solution shown in Figure 5.

By virtue of SOM's well-studied characteristics (Wu and Chow, 2005), it can be surmised that clusters 4 and 5 are somewhat related because the nodes that constitute these clusters are positioned spatially close together in the map. On the other hand, clusters 0 and 3 are positioned at opposite ends of the map, indicating that responses to the survey vary more significantly between these two clusters than between other pairs of clusters in the map. It is also worth noting that even if our clustering method does not force nodes in the same cluster to be contiguous in the map, the clusters we derived are patches of nodes that are mostly contiguous in the map, except for the lone cluster 0 node somewhere above cluster 5. The fact that nodes in a cluster tend to be contiguous in the trained map is a result of the weight update procedure done on the nodes reference vectors during training.

Each respondent record in our dataset is matched to the node in the map with a reference weight vector that has the smallest (Euclidean) distance with respect to it. The cluster number of the nearest node is assigned to the respondent record accordingly. Thus, the entire dataset is now subdivided into subsets of respondent records for each cluster (i.e. the set of in-patterns per cluster). The number of respondent records assigned to each cluster is shown in Table 1. In this study, we skipped the pruning step and retained all the nodes in the different clusters as part of the their respective clusters. As will be discussed in the next section, we will be conducting a separate study focused on the pruned off nodes.

Since cluster 2 is a significantly sized cluster, we probed it further by doing a further k-means clustering on just the nodes in this cluster 2. At $k = 3$, we


(0) SYMBOLIC
(5) UTILITY
safe/quality
(1) UTILITY
dependability
(4) UTILITY
safe/economical
(3) ECONOMIC
quality

0	0	0	0	0	2	2	2	2	2	2	2	2	2	2	2
0	0	0	0	2	2	2	2	2	2	2	2	2	2	2	2
0	0	0	0	2	2	2	2	2	2	2	0	2	2	2	2
0	0	0	0	2	2	2	2	2	2	2	2	2	2	2	2
0	0	0	2	2	2	5	5	5	5	5	5	2	2	2	2
2	2	2	2	2	2	5	5	5	5	5	5	2	2	2	2
2	2	2	2	2	5	5	5	5	5	5	5	2	2	2	2
1	1	1	2	2	5	5	5	5	5	5	5	2	2	2	2
1	1	1	2	2	2	4	5	5	5	2	2	2	2	2	2
1	1	1	2	5	2	4	4	5	5	2	2	2	3	3	3
1	1	2	5	5	4	4	4	4	2	2	2	2	3	3	3
1	1	2	2	4	4	4	4	4	2	2	2	2	3	3	3
1	1	2	4	4	4	4	2	2	2	2	2	2	3	3	3
1	1	2	4	4	4	4	2	2	2	2	2	2	3	3	3
2	2	3	2	2	2	2	2	2	2	2	2	2	2	3	3
2	2	3	2	2	2	2	2	2	2	2	2	2	2	2	2

Fig. 5. The trained 16 × 16 SOM with a clustering of nodes. Cluster 2 nodes have subsequently been clustered into 3 smaller sub-clusters.

are able to break-up cluster 2 into meaningful sub-clusters. This portion of our market segmentation study, which does a second k-means clustering on the relatively very large cluster from the initial clustering, deviates somewhat from the method outlined in the preceding section. However, the goal is to reach a final clustering of the SOM nodes, and we are consistent as far as this goal is concerned. Note that if we simply do a one-step k-means clustering using $k = 8$, we would not be able to obtain the same quality of clusters that we obtained here.

We then inspect the respondent records in each cluster to generate the profile of benefits-sought in each of the clusters. The frequency distribution of each selected benefit in the entire survey set is first computed. We then compute the frequency distributions for just the individual clusters, after which we compute the difference factors between in-patterns and out-patterns on a per dimension basis. The difference factors between the set of in-patterns and the set of out-patterns for each of the benefit dimensions are shown in Table 2. For each cluster, we also compute the mean and the standard deviation of these difference factors. These would be the basis for deciding which dimensions are salient for each cluster.

Table 1. Distribution of respondents to the different clusters based on six and eight clusters. Because of its large size, the original cluster 2 was subdivided further into 3 sub-clusters.

	6 clusters		8 clusters	
Cluster	# respondents	(%)	# respondents	(%)
0	248	10.4%	248	10.4%
1	170	7.1%	170	7.1%
2	1334	55.9%		
2-a			704	29.5%
2-b			139	5.8%
2-c			491	20.6%
3	201	8.4%	201	8.4%
4	152	6.4%	152	6.4%
5	280	11.7%	280	11.7%

Table 2. Difference factors of each dimension (benefit) for every cluster. Bold figures denote difference factors that are more than one standard deviation from the mean.

	Clusters							
Dimension	0	1	2-a	2-b	2-c	3	4	5
fun to drive	-0.11	-0.25	-0.49	-0.34	**1.84**	-0.20	-0.78	0.10
acceleration/speed	0.51	-0.87	0.43	-0.64	0.55	-0.75	-0.89	0.10
dealer service	-0.76	-0.59	-0.09	-1.00	0.55	0.10	**3.87**	-0.71
fuel economy	-0.39	-0.02	**1.63**	1.08	-0.99	0.59	0.56	-1.00
styling	-0.52	-0.42	**0.92**	-0.51	0.46	-0.11	-0.85	-0.44
level of technology	0.18	-0.50	-0.36	0.12	**1.47**	-0.83	-0.95	0.41
luxury features	**13.25**	-0.90	-0.72	-1.00	-0.58	-0.72	-1.00	0.23
made to last	-0.73	**4.80**	-0.54	-0.62	-0.51	0.02	**2.34**	-0.23
prestige	1.46	-0.56	-0.49	-0.68	0.87	-0.16	-0.71	0.13
reliability	-0.77	**5.18**	0.01	-0.90	-0.09	0.04	-0.66	-0.63
safety in accident	-0.97	-0.56	0.44	-0.68	-0.84	-0.55	**2.05**	**2.49**
sportiness	1.74	-0.40	-0.27	-0.67	0.96	-0.59	-0.45	-0.59
quality	-0.79	-0.54	-0.96	-0.83	-0.09	**11.00**	-1.00	**0.89**
passenger space	-0.24	-0.87	-0.18	**3.98**	0.06	-0.68	-0.66	0.34
cargo/luggage space	-0.42	-0.85	0.12	**8.56**	-0.25	-0.79	-0.83	-0.68
μ	0.76	0.18	-0.04	0.39	0.23	0.42	0.00	0.03
σ	3.55	1.97	0.67	2.59	0.83	0.95	1.52	0.86
$\mu+\sigma$	4.31	2.15	0.64	2.98	1.06	3.38	1.52	0.89

Since this is a market-segmentation study, we refer to the salient dimensions as the "primary benefit(s)" sought by respondents in each cluster. We only considered positive difference factors, because of the nature of the study. We are mainly trying to establish what car features each cluster of consumers is seeking when buying a car, and not so much what they are least interested in. The distinctive meaning of each segment is then determined by assessing the combination of benefits and the importance respondents attach to individual benefits. Eight types of benefit segments were identified, as listed in

Table 3. In the table, we also present "secondary benefits" since they aid in providing a better profile of the kind of benefits the customers are seeking for each cluster. We consider all positive difference factors that are less than one standard deviation from the mean as "secondary benefits".

Table 3. Primary and secondary benefits sought for each cluster with corresponding difference factors.

Cluster	Label	Primary benefits	Secondary benefits
0	SYMBOLIC	luxury features (13.25)	sportiness (1.74) prestige (1.46) acceleration/speed (0.51) level of technology (0.18)
1	UTILITARIAN (dependability)	reliability (5.18) made to last (4.80)	
2-a	ECONOMIC (fuel economy)	fuel economy (1.63) styling (0.92)	safety in accidents (0.44) acceleration/speed (0.43) cargo/luggage space (0.12) reliability (0.01)
2-b	UTILITARIAN (larger space)	cargo/luggage space (8.56) passenger space (3.98)	fuel economy (1.08) level of technology (0.12)
2-c	SENSORY	fun to drive (1.84) level of technology (1.47)	sportiness (0.96) prestige (0.87) acceleration/speed (0.55) dealer service (0.55) styling (0.46) passenger space (0.06)
3	ECONOMIC (quality)	high quality (11.00)	fuel economy (0.59) dealer service (0.10) reliability (0.04) made to last (0.02)
4	UTILITARIAN (safe/economical)	good dealer service (3.87) made to last (2.34) safety in accidents (2.05)	fuel economy (0.56)
5	UTILITARIAN (safe/high quality)	safety in accidents (2.49) high quality (0.89)	level of technology (0.41) passenger space (0.34) luxury features (0.23) prestige (0.13) acceleration/speed (0.10) fun to drive (0.10)

Cluster 2-a, an economic-oriented segment, is dominated by "good fuel economy" and supplemented by "good styling", "safety in accidents", "good acceleration and speed", "cargo/luggage space" and "reliability". Cluster 2-c, which is a "sensory" segment, values benefits such as "fun to drive", "level of technology", "sportiness", "prestige", "good acceleration and speed" and "styling". Respondents who fall under the "symbolic" segment (cluster 0) are those who value “luxury features” and appreciate other symbol-oriented benefits such as “prestige”, “sportiness”, “good acceleration and speed” and “level of technology”. Cluster 3 is the other economic-oriented segment with benefits such as “high quality”, “fuel economy”, and “good dealer service”.

The remaining four segments differ from each other, but all reflect various utilitarian needs. Cluster 1 represents utilitarian benefit seekers who are after “reliability” and “made to last”. Clusters 4 and 5 are both concerned with “safety in accidents”, except that their respective secondary benefits point to two distinct types of needs. Whereas cluster 4 is a grouping of safety-conscious consumers who value “good dealer service”, “made to last” and “good fuel economy”, cluster 5 consumers are concerned with “high quality”, “level of technology”, “passenger space”, “luxury features”, “prestige”, etc. Finally, cluster 2-b is another utilitarian segment that is mainly focused on “space” benefits, including both “passenger space“ and “cargo/luggage space“.

Note that the component planes shown earlier in Figure 2 are based on the actual trained weights of the SOM we have generated for this study. Knowing the primary and secondary benefits from Table 3, it is easy to work backwards and verify our results. Indeed, when a dimension is a primary benefit for a given cluster, the component plane corresponding to that dimension has a white patch (high weight values) in the section of the map that corresponds to the nodes that make up the given cluster. However, had we relied on just the visual inspection of component planes the way it is usually done, we might be able to manually deduce some of the primary benefits, but the level of detail of Table 3 will be very difficult to match.

We are now ready to construct the demographic profile of each of the eight clusters, including age, gender, marital status, and whether or not a respondent has children under 18. We have done this at the global level as well as at the level of groupings of countries (i.e. continental Europe, Latin America, Anglo-America, East Asia). Interested readers are referred to (Azcarraga *et al.*, 2003) for the marketing context of the study.

Table 4 gives socio-demographic profile of each cluster of car features at the global level in terms of the percent distribution of respondents in each of the car-benefits clusters that were previously identified. From Table 5, we observe that the symbolic segment has a significantly higher proportion of younger consumers in the under-30 age bracket. We expected the sensory segment to be dominated by the younger consumers as well, but this trend is not significant at the global level.

A quite unexpected result is the significantly higher proportion of female consumers who value passenger and cargo space. In the global sample, only 37% of the respondents are female. This proportion increased significantly to 50% for cluster 2-b. Also, married consumers prefer low maintenance attributes and good dealer service. Furthermore, the economic cluster 3 shows a significantly dominant middle age consumer bracket from 30 to 40 years old, and slightly older.

Table 4. Global socio-demographic profile of each benefits-sought cluster (figures are in %).

Clusters		Demographics								
		gender		status		age				
		female	male	single	married	< 30	30-39	40-49	50-59	≥ 60
0	symbolic	35	65	47	53	33	30	23	12	8
1	utilitarian (dependability)	34	66	29	71	19	32	25	14	10
2-a	utilitarian (fuel economy)	38	62	39	61	26	29	23	14	8
2-b	utilitarian (space)	50	50	42	58	24	30	24	12	9
2-c	sensory	33	67	39	61	31	29	20	11	9
3	economic (quality)	38	62	39	61	24	36	24	11	4
4	utilitarian (safe/economical)	41	59	25	75	15	34	24	17	11
5	utilitarian (safe/quality)	38	62	34	66	25	26	26	11	10
All		**37**	**63**	**38**	**62**	**26**	**30**	**23**	**12**	**8**

Table 5. Global socio-demographic profile of each benefits-sought cluster based on net deviation of % proportion from mean. Items labeled as ++ and −− have one standard deviation of positive or negative deviation from the mean. Others are labeled as + or − when the net deviation is more than 5%.

Clusters		Demographics								
		gender		status		age				
		female	male	single	married	< 30	30-39	40-49	50-59	≥ 60
0	symbolic	−	+	+	−	++		−	−	−
1	utilitarian (dependability)	−	+	−	+	−−	+	+	+	+
2-a	utilitarian (fuel economy)						−		+	
2-b	utilitarian (space)	++	−−	+	−	−		+	−	
2-c	sensory	−	+			+		−	−	
3	economic (quality)					−	++	+	−	−
4	utilitarian (safe/economical)	+	−	−−	++	−−	+		+	+
5	utilitarian (safe/quality)			−	+	−	−	+	−	+

The socio-demographic profiles of each cluster are more pronounced when studied at a regional level than at the global level, as can be seen from Table 6. At the global level, the profiles of the various niche markets in the regional levels tend to cancel each other.

For the purpose of illustrating how we are able to mine for insights that may be very useful in marketing applications, for example, we will highlight here a few of the marketing-related results for Continental Europe and Latin America. In Continental Europe, gender does not matter much compared to the other regional-cultural blocs. Whether a consumer is male or female matters only in cluster 2-b, associated with passenger and cargo space. Latin America is where socio-demographics matter the most. For example, proportionately more unmarried (single) consumers go for symbolic, utilitarian (space), and economic benefits, while the married consumers go for dependability and safe-and-economical. Gender matters in all clusters as well, even significantly with dependability and sensory benefits for males, and safe-and-economical and space benefits for females. Age is significantly pronounced in seven of the eight Latin American clusters. It should be noted that like the Anglo-American bloc, the proportion for the above-60 age bracket (retired segment) is significantly higher for three benefits clusters. However, only one

Table 6. Regional socio-demographic profile of each benefits-sought cluster.

Clusters		**Demographics (Anglo-America)**								
		gender		status		age				
		female	male	single	married	< 30	30-39	40-49	50-59	≥ 60
0	symbolic	−−	++	+	−	+	++		−	++
1	utilitarian (dependability)	−	+	−	+	−−	+	+	+	+
2-a	utilitarian (fuel economy)						−		+	
2-b	utilitarian (space)	++	−−	+	−	−		+	−	
2-c	sensory	−	+			+		−	−	
3	economic(quality)					−	++	+	−	−
4	utilitarian (safe/economical)	+	−	−−	++	−−	+		+	+
5	utilitarian (safe/quality)			−	+	−	−	+	−	+

Clusters		**Demographics (Continental Europe)**								
		gender		status		age				
		female	male	single	married	< 30	30-39	40-49	50-59	≥ 60
0	symbolic			+	−	++	−	+	−	−
1	utilitarian (dependability)	−	+	−−	++	−−		+	+	+
2-a	utilitarian (fuel economy)					−				+
2-b	utilitarian (space)	++	−−			−	+	−	−	+
2-c	sensory	−	+	+	−	++		−	−	
3	economic (quality)	−	+	+	−	−	++		−	−
4	utilitarian (safe/economical)			−	+	−−	−	+	++	−
5	utilitarian (safe/quality)	+	−			+	−	+	+	−

Clusters		**Demographics (Latin America)**								
		gender		status		age				
		female	male	single	married	< 30	30-39	40-49	50-59	≥ 60
0	symbolic	−	+	++	−−	++		−−	−	−
1	utilitarian (dependability)	−−	++	−−	++	−−	−	+		++
2-a	utilitarian (fuel economy)	++	−−	−	+		−		−	++
2-b	utilitarian (space)	++	−−	++	−−	++	−−	−	−−	−−
2-c	sensory	−−	++	−	+	+	+	−	−−	−
3	economic (quality)	+	−	++	−−		++	−		−
4	utilitarian (safe/economical)	+	−	−−	++	−−	++			++
5	utilitarian (safe/quality)	−	+	−	+	−−	−	++	−	−

Clusters		**Demographics (East Asia)**								
		gender		status		age				
		female	male	single	married	< 30	30-39	40-49	50-59	≥ 60
0	symbolic			++	−−	++		−−		−
1	utilitarian (dependability)	−	+	−	+	−−	++		−	−
2-a	utilitarian (fuel economy)	++	−	+	−	+	−−		+	
2-b	utilitarian (space)	++	−−			+	−−	++	−	−
2-c	sensory	−−	++	−	+		+	−	−	+
3	economic (quality)	++	−−			+	+			−
4	utilitarian (safe/economical)	+	−	−−	++	−−	+	+		+
5	utilitarian (safe/quality)	−	+			+	−−	+		+

of the three clusters represents the same group of benefits sought, namely "safe and economical" (cluster 4).

In a very succinct manner, a SOM rendering of the marketing information contains a lot of relevant information that are more readily understood when shown in a picture form (Figures 6 and 7) than in the form of a table, as in Table 6. The SOM, once appropriately labeled, does provide various insights depending on the use of the information revealed by the SOM. In Figure 6, we note that luxury features and economic benefits are positioned at opposite corners of the map, reflecting the distinct types of benefits they include. In the middle portion are most of the utilitarian benefits, with the two safety-in-accidents clusters positioned side-by-side each other in the center of the map.

For the Anglo-American bloc, the males are mainly attracted to the symbolic benefit, while a significantly higher proportion of women are attracted to two utilitarian clusters (clusters 2-b and 4) in the mid-section of the map. The married consumers are likewise attracted to cluster 4 (i.e., safety and economic related benefits). As for the age, the young consumers are attracted to the symbolic and sensory clusters, as expected, while the middle-aged consumers are attracted to utilitarian and economic clusters. There is a distinct market among the 60-over consumers in the Anglo-American bloc (which is not evident in Continental Europe and East Asia), in that they gravitate significantly towards the symbolic benefit as well as the two safety-in-accidents benefits (i.e., clusters 4 and 5).

In Figure 7 we use another visual SOM-rendering of the socio-demographic segmentation of a group of countries (East Asia) to reinforce the claim that such "pictures" can be more insightful than the usual tabular presentation of Table 6. Notice how much easier it is to see the over-all picture when the demographics are presented as shown in Figure 7. In addition, the socio-demographic profiles of the Anglo-American and East Asian blocs can be readily compared. In the East-Asian bloc, the males are mainly attracted to the sensory benefits ("level of technology", "fun to drive", etc.), while a significantly higher proportion of women is attracted cluster 2-b (space) and cluster 3 "high quality". The married consumers are attracted to cluster 4 (i.e., safety and economic related benefits). The young consumers under 30 years old are attracted to the symbolic cluster ("luxury features"), while those in the 30-39 range go for "durability", and those in the 40-59 range go for "space".

5 Assessing the quality of the cluster labels

Although the method works well for the market segmentation study discussed above, one may wonder whether other known methods might yield similar results or might provide further evidence that the results are indeed satisfactory. To address this question, we assessed the quality of the clusters and of the identified salient dimensions using various statistical tests and machine learning techniques.

First, we applied Wilks' Lambda F test using MANOVA to test the effect of each of the factors (i.e., eight clusters) on the dependent variables (i.e., 15 benefits) on a pair-wise basis. Wilks' Lambda is a test statistic frequently used in multivariate analysis for mean differences among more than two groups. In our case, this test would establish whether in fact the eight clusters that we have generated are distinct in terms of benefits desired by respondents in each cluster. At significance level .01, the centroid vector representing each cluster was found to be pair-wise different from each of the centroids vectors of all the other clusters. As described earlier, we also saw that each cluster can be associated with a unique set of benefits that would establish its distinctive

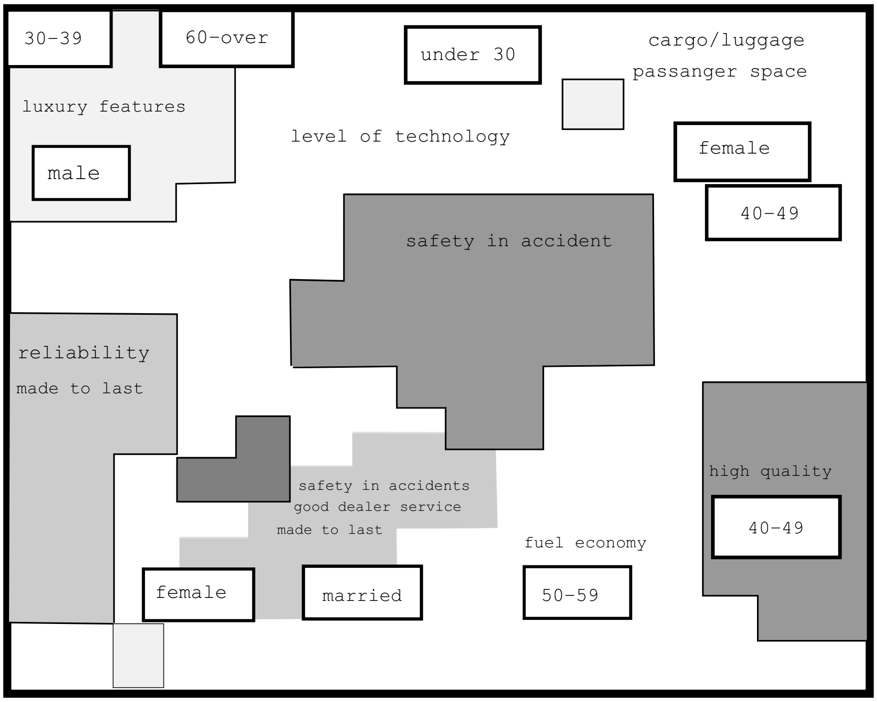


Fig. 6. Labeled SOM with superimposed socio-demographics for the Anglo-American bloc.

character. That we are able to do so further supports our claim that the eight clusters are distinct.

To validate the assignment of primary and secondary benefits to each cluster, we relied on the tests of between-subject effects generated from the same MANOVA to check for significant effect of cluster on each benefit across different clusters. We noted for each cluster all those benefits which are different at significance level .05 from all the seven other clusters. From among these, we further identify those benefits whose means are higher than the over-all mean (all clusters included). Table 7 shows that the benefits identified as outlined above are exactly those listed as either primary or secondary benefits in Table 3, which used difference factors as basis for identifying them. Furthermore, almost all of the primary benefits for each cluster in Table 3 are significantly different from all seven other clusters as shown in Table 7. The only exception is the "fuel economy" primary benefit of cluster 2-a. It has a mean value that is not significantly different from that of cluster 2-b, which also has "fuel economy" as a secondary benefit.

These tests for independence that help us determine which of the dimensions are significantly different for a given cluster are only useful for validating the selection of significant dimensions that we have previously done. If it were

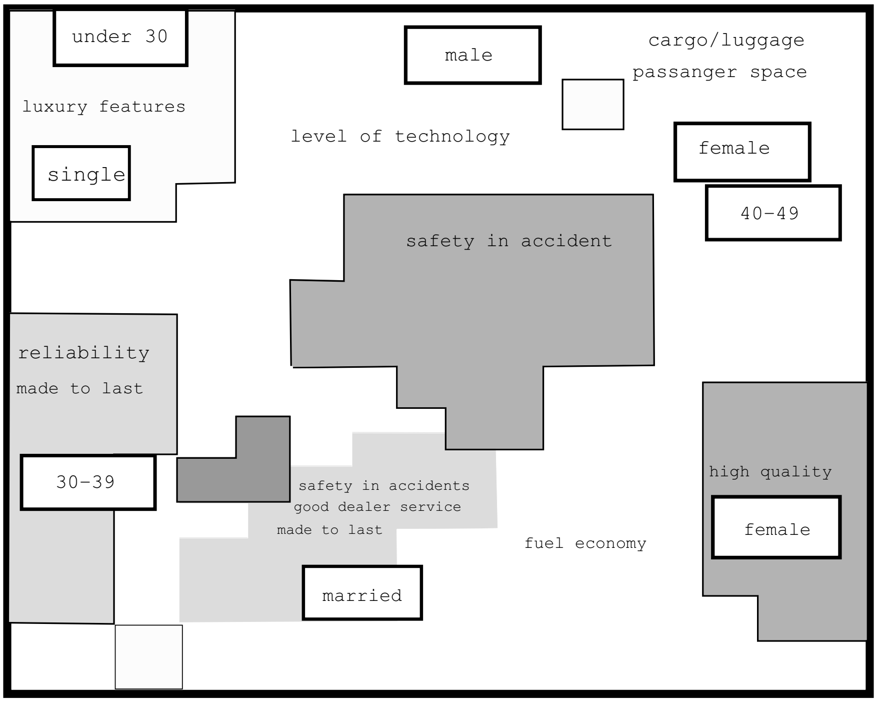


Fig. 7. Labeled SOM with superimposed socio-demographics for the East-Asian bloc.

used as the main technique for identifying the significant dimensions, it is not clear what criteria (cut-off point) should be used to identify the salient dimensions. For example, a given dimension that is extremely high for three and very low for five out of eight clusters may in fact be a "salient dimension" for labeling purposes. Yet, if the cut-off is set at six out of seven other clusters, such a dimension may not be selected as a salient dimension. Also, secondary benefits would have been difficult to identify.

A final validation technique involves the use of C4.5, a very well-established decision tree classifier (Quinlan, 1993). We use C4.5 to automatically build a decision tree that would classify respondent patterns as either belonging to the in-patterns set or the out-patterns set. A very useful feature of C4.5 is that aside from building a decision tree, it is able to extract a set of rules that mimics the decision-making process of the tree. Figure 8 is an example of the extracted rules for cluster 0, after C4.5 has built a decision tree based on the sets of in-patterns and out-patterns for cluster 0. In the rules of Figure 8, class 0 refers to the out-patterns set while class 1 refers to the in-patterns set. Note that C4.5 automatically drops certain rules, which explains why rules 3 to 6 are missing.

Table 7. Salient dimensions, not marked by (*), are significantly different at $\alpha = .05$ from all other seven clusters based on the F Lambda test. Those marked by (*) are significantly different from only six out of the seven other clusters.

Cluster	Label	Salient Dimensions
0	SYMBOLIC	luxury features sportiness prestige (*)
1	UTILITARIAN (dependability)	reliability made to last
2	ECONOMIC (fuel economy)	safety in accidents fuel economy (*) cargo/luggage space (*)
2-a	UTILITARIAN (larger space)	cargo/luggage space passenger space fuel economy (*)
2-b	SENSORY	level of technology fun to drive sportiness
3	ECONOMIC (quality)	high quality fuel economy (*)
4	UTILITARIAN (safe and economical)	good dealer service safety in accidents made to last fuel economy (*)
5	UTILITARIAN (safe and high quality)	safety in accidents high quality luxury features passenger space (*)

We basically would want the extracted rules to confirm that cluster 0 is a "symbolic" cluster, with "luxury features" as the primary benefit. Indeed, this is what C4.5 is able to extract from the decision tree it has built for cluster 0. Clearly, the rules extracted by C4.5 are the kinds of information we need when we want to assign descriptive labels to each cluster in the map. As a validation technique, it does well in confirming that the significant dimensions identified by the proposed methodology are in fact what the C4.5 rules are using in deciding that a pattern belongs to the in-patterns set. In fact, all the significant dimensions we have identified have also appeared prominently in the respective rules extracted by C4.5 (in the interest of space, we are not showing them all here).

Just like for the F tests for significance, it is tempting to conclude that C4.5 can replace the dimension-selection technique based on difference factors. Just to illustrate how it can be quite complicated to do so, we refer to Figure 9 containing the list of extracted C4.5 rules for cluster 2-b. The rules set for cluster 2-b is already the next to the simplest, after that of cluster

```
Rule 1:
        Q7-Luxury-Features = 0
        ->  class 0  [99.9%]

Rule 7:
        Q11-Safety-in-Accidents = 1
        ->  class 0  [99.4%]

Rule 2:
        Q3-Dealer-Service = 0
        Q5-Styling = 0
        Q7-Luxury-Features = 1
        Q10-Reliability = 0
        Q11-Safety-in-Accidents = 0
        Q13-Quality = 0
        ->  class 1  [95.9%]
```

Fig. 8. Extracted rules using C4.5 given the sets of in-patterns (class 1) and out-patterns (class 0) for cluster 0. Clearly, the dimension "luxury features" is the salient dimension for this "symbolic" cluster, thus validating the salient dimension (primary benefit) extracted by our method as shown in Table 3.

0. Those for the other clusters have many more rules with many more dimensions included in the rule conditions. Again, knowing that our method has tagged "passenger space" and "luggage space" as the primary benefits for this cluster, we can confirm the validity of our extracted primary labels. However, if C4.5 were directly used to isolate these two primary benefits, there is no straightforward method for isolating these two dimensions from "fuel economy" for example, which also appears prominently in the rules. Only by assessing the rules together as a set, are we able to discern the truly important variables. This obviously becomes extremely difficult when the set of rules is complex. Furthermore, there is no clear way of ranking the dimensions from the most significant to the least significant, the way it can be done in a very neat manner using difference factors.

There remains one more important point. Although it is quite clear that the two validation methods employed here - statistical and decision-tree - cannot be used as alternatives to our method based on difference factors, it remains a methodological issue whether one or both methods ought to be included in the general unsupervised labeling methodology that we are proposing.

We prefer to leave validation out as an optional phase and to use it only when 1) the user has absolutely no idea as to what constitutes a "reasonable" set of findings; and 2) the results appear counter-intuitive. Basically, the validation techniques, particularly the rule extraction method using C4.5, would be useful to detect some procedural or computational mistakes related to the computation of difference factors, means and standard deviations. However,

```
Rule 5:
        Q4-Fuel-Economy = 1
        Q15-Cargo-Luggage-Space = 1
        ->  class 1  [82.9%]

Rule 3:
        Q11-Safety-in-Accidents = 0
        Q14-Passenger-Space = 1
        Q15-Cargo-Luggage-Space = 1
        ->  class 1  [71.8%]

Rule 2:
        Q4-Fuel-Economy = 0
        Q14-Passenger-Space = 0
        ->  class 0  [99.9%]

Rule 4:
        Q4-Fuel-Economy = 0
        Q11-Safety-in-Accidents = 1
        ->  class 0  [99.7%]

Rule 1:
        Q15-Cargo-Luggage-Space = 0
        ->  class 0  [97.9%]
```

Fig. 9. Extracted rules using C4.5 given the sets of in-patterns (class 1) and out-patterns (class 0) for cluster 2-b. Although "passenger space" and "cargo/luggage space" which are the cluster's primary benefits from Table 3, do appear prominently in the rules, it is not clear how these two dimensions could have been isolated from the other dimensions (e.g. "fuel economy") just based on the rules extracted for cluster 2-b.

it must be emphasized that since C4.5 comes into the picture after the sets of in-patterns and out-patterns of each cluster have been determined, any error prior to this step would remain undetected.

6 Conclusion

Neural networks are potent data mining models, as they are able to learn just from being shown examples of the data, without explicitly being told what to look for, or how the information is structured in the input heap. Indeed, data mining tools become very useful precisely when there is little knowledge about what might be contained in the input data, and often times, the dataset has no veritable structure to speak of. Among the neural network models, self-organizing maps, which belong to the class of unsupervised neural network learning models, become doubly interesting for data mining applications because this model does not require training data to have accompanying desired-output information that typically would need some tedious user-intervention (which is the case for supervised neural network learning models).

It must be emphasized that although training (learning) in self-organizing maps (SOMs) is unsupervised, the labeling phase is very often a supervised process. The labeling process is supervised in that we rely on labeled patterns that have accompanying desired-output information. Since such labeled patterns are not always available or may not even be possible to construct, the supervised labeling phase of the SOM methodology hinders the deployment of SOMs to a wider range of potential domains of application.

We improved on the SOM methodology by devising a methodical and automatic SOM labeling procedure that does not require a set of labeled patterns. Nodes of the trained map are clustered and the so-called *salient dimensions* in each cluster are automatically identified. Out of these salient dimensions, a “descriptive label” is assigned by the user. Assignment of a descriptive label is still a form of user intervention, but this is no longer at the level of individual labeled patterns.

We have illustrated the effectiveness of the method by applying the unsupervised labeling method to a SOM-based customer-profiling study. The market segmentation application illustrates the usefulness of SOM as a methodology for data mining, through clustering and visualization of unstructured data.

In the market segmentation study, clustering of the benefits-sought data could have been done by a multitude of clustering techniques, and a number of these would probably generate a similar segmentation. However, SOM provides an additional feature: visualization of the clusters on a simple 2D grid that would position the clusters in such a way that those that are near each other, in a spatial sense, pertain to benefits groupings that are fairly similar (i.e. the Euclidean distance of their associated input vectors is small). In addition, we are able to superimpose on this cluster distribution the primary benefits for the different clusters, and the various socio-demographic patterns for the different market niches represented by the clusters.

Acknowledgment

The authors acknowledge the contribution of MORPACE International Inc. in providing the dataset for analysis.

References

Azcarraga AP, Hsieh M, Setiono R, (2003), Visualizing globalization: A SOM approach to customer profiling. In:Proceedings of 24th International Conference on Information Systems (ICIS), Seattle, WA.

Azcarraga A, Yap TN, Tan J, Chua TS, (2002), Evaluating keyword selection methods for WEBSOM text archives, IEEE Transactions on Knowledge and Data Engineering, 16(3): 380–383.

Carlson E, (1998), Real estate investment appraisal of land properties using SOM. In: Deboeck G, Kohonen T (eds), Visual explorations infinance with self-organizing maps, Springer-Verlag, London.

Carpenter GA, Grossberg S, (1991), Pattern-recognition by self-organizing neural networks. MIT Press, Cambridge, MA.

Clark D, Ravishankar K, (1990), A convergence theorem for Grossberg learning, Neural Networks 3(1): 87–92.

Deboeck G, Kohonen T, (1998), Visual explorations in finance with self-organizing maps,Springer-Verlag, London.

Deboeck G, (1998), Picking mutual funds with self-organizing maps. In:Deboeck G, Kohonen T (eds), Visual explorations in finance with self-organizing maps, Springer-Verlag, London.

Deboeck G, (1998), Investment maps of emerging markets. In: Deboeck G, Kohonen T (eds), Visual explorations in finance with self-organizing maps, Springer-Verlag, London.

Everitt B, (1974), Cluster analysis, Heinemann Educational Books, London.

Fukushima K, (1980), Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift inposition, Biological Cybernetics 36: 121–136.

Hartigan JA, (1975), Clustering algorithms, Wiley-Interscience, New York.

Haykin S, (1998), Neural networks: a comprehensive foundation. Prentice-Hall International, 2nd Edition, Upper Saddle River, NewJersey.

Holbrook MB, Schindler RM, (1994), Age, sex, and attitude toward the pastas predictors of consumers' aesthetic taste for cultural products. Journal of Consumer Research 31: 412–22.

Hsieh MH, (2002), Identifying brand image dimensionality and measuring degree of brand globalization: a cross-national study. Journal of International Marketing 10(2): 46–67.

Kiang MY, Kumar A, (2001), An evaluation of self-organizing map networks as a robust alternative to factor analysis in data mining applications, Information Systems Research 12: 177–194.

Kiviluto K, Bergius P, (1998), Maps for analyzing failures of small andmedium-sized enterprises. In: Deboeck G, Kohonen T (eds), Visual explorations in finance with self-organizing maps,Springer-Verlag, London.

Kohonen T, (2000), Self-organization of a massive document collection, IEEE Transactions on Neural Networks 11(3): 574–585.

Kohonen T, (1982), Self-organized formation of topologically-correct feature maps, Biological Cybernetics 43: 59–69.

Kohonen T, (1990), The self-organizing map, Proceedings of the IEEE 78:1464–1480.

Kohonen T, (1995), Self-organizing maps, Springer-Verlag, Berlin.

Kohonen T, (1999), Kohonen maps, Elsevier, New York.

Kuo RJ, Ho LM, Hu CM, (2002), Integration of self-organizing feature mapand k-means algorithm for market segmentation, Computers and Operations Research 29:1475–1493.

Mayer R, Lidy T, Rauber A, (2006), The map of Mozart, Proc 7th International Conference on Music Information Retrieval, Victoria,Canada, Oct 8-12.

Merkl D, (1998), Text classification with self-organizing maps: some lessons learned, Neurocomputing 21: 61–77.

Park CW, Jaworski BJ, MacInnis DJ, (1986), Strategic brand concept-imagemanagement. Journal of Marketing 50: 135–145.

Park CW, Milberg S, Lawson R, (1991), Evaluation of brand extension:the role of product level similarity and brand concept consistency. Journal of Consumer Research 18: 185–193.

Punj G, Steward DW, (1983), Cluster analysis in marketing research: review and suggestions for applications. Journal of Marketing Research 20: 134–148.

Quinlan R, (1993), C4.5: Programs for machine learning, Morgan Kaufman,San Mateo, CA.

Resta M, (1998), A hybrid neural network system for trading financial markets. In: Deboeck G, Kohonen T (eds), Visual explorations infinance with self-organizing maps, Springer-Verlag, London.

Ritter H, Martinetz T, Schulten K, (1992), Neural computation and self-organizing maps (translated from German), Addison-Wesley, Reading MA.

Rumelhart DE, Zipser D, (1986), Feature discovery by competitive learning. In: Rumelhart DE and McClelland JL (eds) Parallel and Distributed Processing, Vol 1, 151-193. MIT Press, Cambridge, CA.

Rumelhart DE, Hinton GE, Williams RJ, (1986), Learning internal representations by error propagation. In: Rumelhart DE, McClelland JL (eds) Parallel and Distributed Processing, Vol 1. 318-362. MITPress, Cambridge, MA.

Schmitt B, Deboeck G, (1998), Differential patterns in consumer purchase preferences using self-organizing maps: a case study of China. In:Deboeck G, Kohonen T (eds), Visual explorations in finance withself-organizing maps, Springer-Verlag, London.

Serrano-Cinca C, (1998), Let financial data speak for themselves. In:Deboeck G, Kohonen T (eds), Visual explorations in finance with self-organizing maps, Springer-Verlag, London.

Shumsky S, Yarovoy AV, (1998), Self-organizing atlas of Russian banks. In:Deboeck G, Kohonen T (eds), Visual explorations in finance with self-organizing maps, Springer-Verlag, London.

Spath H, (1980), Cluster analysis algorithms, Ellis Horwood, Chichester,England.

Tulkki A, (1998), Real estate investment appraisal of buildings using SOM. In: Deboeck G, Kohonen T (eds), Visual explorations in financewith self-organizing maps, Springer-Verlag, London.

Wedel M, Kamakura W, (1998), Market segmentation: conceptual and methodological foundations, Kluwer Academic Publishers, Boston,MA.

Wu S, Chow T, (2005), PRSOM: A new visualization method by hybridizing multi dimensional scaling and self-organizing Map, IEEE Trans on Neural Networks 16(6): 1362–1380.

Xu R, Wunsch D, (2005), Survey of cluster algorithms, IEEE Trans on Neural Networks, 16(3): 645–678.

A Review of Evolutionary Algorithms for Data Mining

Alex A. Freitas

University of Kent, UK, Computing Laboratory, A.A.Freitas@kent.ac.uk

Summary. Evolutionary Algorithms (EAs) are stochastic search algorithms inspired by the process of neo-Darwinian evolution. The motivation for applying EAs to data mining is that they are robust, adaptive search techniques that perform a global search in the solution space. This chapter first presents a brief overview of EAs, focusing mainly on two kinds of EAs, viz. Genetic Algorithms (GAs) and Genetic Programming (GP). Then the chapter reviews the main concepts and principles used by EAs designed for solving several data mining tasks, namely: discovery of classification rules, clustering, attribute selection and attribute construction. Finally, it discusses Multi-Objective EAs, based on the concept of Pareto dominance, and their use in several data mining tasks.

Key words: genetic algorithm, genetic programming, classification, clustering, attribute selection, attribute construction, multi-objective optimization

1 Introduction

The paradigm of Evolutionary Algorithms (EAs) consists of stochastic search algorithms inspired by the process of neo-Darwinian evolution (Back et al. 2000; De Jong 2006; Eiben & Smith 2003). EAs work with a population of individuals, each of them a candidate solution to a given problem, that “evolve” towards better and better solutions to that problem. It should be noted that this is a very generic search paradigm. EAs can be used to solve many different kinds of problems, by carefully specifying what kind of candidate solution an individual represents and how the quality of that solution is evaluated (by a “fitness” function).

In essence, the motivation for applying EAs to data mining is that EAs are robust, adaptive search methods that perform a global search in the space of candidate solutions. In contrast, several more conventional data mining methods perform a local, greedy search in the space of candidate solutions. As a result of their global search, EAs tend to cope better with attribute interactions than greedy data mining methods (Freitas 2002a; Dhar et al. 2000; Papagelis & Kalles 2001; Freitas 2001, 2002c). Hence, intuitively EAs can discover interesting knowledge that would be missed by a greedy method.

The remainder of this chapter is organized as follows. Section 2 presents a brief overview of EAs. Section 3 discusses EAs for discovering classification rules. Section 4 discusses EAs for clustering. Section 5 discusses EAs for two data preprocessing tasks, namely attribute selection and attribute construction. Section 6 discusses multi-objective EAs. Finally, Section 7 concludes the chapter. This chapter is an updated version of (Freitas 2005).

2 An Overview of Evolutionary Algorithms

An Evolutionary Algorithm (EA) is essentially an algorithm inspired by the principle of natural selection and natural genetics. The basic idea is simple. In nature individuals are continuously evolving, getting more and more adapted to the environment. In EAs each "individual" corresponds to a candidate solution to the target problem, which could be considered a very simple "environment". Each individual is evaluated by a fitness function, which measures the quality of the candidate solution represented by the individual. At each generation (iteration), the best individuals (candidate solutions) have a higher probability of being selected for reproduction. The selected individuals undergo operations inspired by natural genetics, such as crossover (where part of the genetic material of two individuals are swapped) and mutation (where part of the generic material of an individual is replaced by randomly-generated genetic material), producing new offspring which will replace the parents, creating a new generation of individuals. This process is iteratively repeated until a stopping criterion is satisfied, such as until a fixed number of generations has been performed or until a satisfactory solution has been found.

There are several kinds of EAs, such as Genetic Algorithms, Genetic Programming, Classifier Systems, Evolution Strategies, Evolutionary Programming, Estimation of Distribution Algorithms, etc. (Back et al. 2000; De Jong 2006; Eiben & Smith 2003). This chapter will focus on Genetic Algorithms (GAs) and Genetic Programming (GP), which are probably the two kinds of EA that have been most used for data mining.

Both GA and GP can be described, at a high level of abstraction, by the pseudocode of Algorithm 1. Although GA and GP share this basic pseudocode, there are several important differences between these two kinds of algorithms. One of these differences involves the kind of solution represented by each of these kinds of algorithms. In GAs, in general a candidate solution consists mainly of values of variables – in essence, data. By contrast, in GP the candidate solution usually consists of both data and functions. Therefore, in GP one works with two sets of symbols that can be represented in an individual, namely the terminal set and the function set. The terminal set typically contains variables (or attributes) and constants; whereas the function set contains functions which are believed to be appropriate to represent good solutions for the target problem. In the context of data mining, the explicit use of a function set is interesting because it provides GP with potentially powerful means of changing the original data representation into a representation that is more suitable for knowledge discovery purposes, which is not so naturally done when using GAs or another EA where only attributes (but not functions) are represented by an individual. This ability of changing the data representation will be discussed particularly on the section about GP for attribute construction.

Note that in general there is no distinction between terminal set and function set in the case of GAs, because GAs' individuals usually consist only of data, not functions. As a result, the representation of GA individuals tend to be simpler than the representation of GP individuals. In particular, GA individuals are usually represented by a fixed-length linear genome, whereas the genome of GP individuals is often represented by a variable-size tree genome – where the internal nodes contain functions and the leaf nodes contain terminals.

Algorithm 1: Generic Pseudocode for GA and GP

```
1: Create initial population of individuals
2: Compute the fitness of each individual
3: repeat
4:    Select individuals based on fitness
5:    Apply genetic operators to selected individuals, creating new individuals
6:    Compute fitness of each of the new individuals
7:    Update the current population (new individuals replace old individuals)
8: until (stopping criteria)
```

When designing a GP algorithm, one must bear in mind two important properties that should be satisfied by the algorithm, namely closure and sufficiency (Banzhaf et al. 1998; Koza 1992). Closure means that every function in the function set must be able to accept, as input, the result of any other function or any terminal in the terminal set. Some approaches to satisfy the closure property in the context of attribute construction will be discussed in Subsection 5.2. Sufficiency means that the function set should be expressive enough to allow the representation of a good solution to the target problem. In practice it is difficult to know *a priori* which functions should be used to guarantee the sufficiency property, because in challenging real-world problems one often does not know the shape of a good solution for the problem. As a practical guideline, (Banzhaf et al. 1998) (p. 111) recommends:

"An approximate starting point for a function set might be the arithmetic and logic operations: PLUS, MINUS, TIMES, DIVIDE, OR, AND, XOR. . . . Good solutions using only this function set have been obtained on several different classification problems,. . . ,and symbolic regression problems."

We have previously mentioned some differences between GA and GP, involving their individual representation. Arguably, however, the most important difference between GAs and GP involves the fundamental nature of the solution that they represent. More precisely, in GAs (like in most other kinds of EA) each individual represents a solution to one particular instance of the problem being solved. In contrast, in GP a candidate solution should represent a generic solution – a program or an algorithm – to the kind of problem being solved; in the sense that the evolved program should be generic enough to be applied to any instance of the target kind of problem.

To quote (Banzhaf et al. 1998), p. 6:

> it is possible to define genetic programming as the direct evolution of *programs or algorithms* [our italics] for the purpose of inductive learning.

In practice, in the context of data mining, most GP algorithms evolve a solution (say, a classification model) *specific for a single data set*, rather than a *generic program* that can be applied to different data sets from different application domains. An exception is the work of (Pappa & Freitas 2006), proposing a grammar-based GP system that automatically evolves full rule induction algorithms, with loop statements, generic procedures for building and pruning classification rules, etc. Hence, in this system the output of a GP run is a *generic* rule induction algorithm (implemented in Java), which can be run on virtually any classification data set – in the same way that a manually-designed rule induction algorithm can be run on virtually any classification data set. An extended version of the work presented in (Pappa & Freitas 2006) is discussed in detail in another chapter of this book (Pappa & Freitas 2007).

3 Evolutionary Algorithms for Discovering Classification Rules

Most of the EAs discussed in this section are Genetic Algorithms, but it should be emphasized that classification rules can also be discovered by other kinds of EAs. In particular, for a review of Genetic Programming algorithms for classification-rule discovery, see (Freitas 2002a); and for a review of Learning Classifier Systems (a type of algorithm based on a combination of EA and reinforcement learning principles), see (Bull 2004; Bull & Kovacs 2005).

3.1 Individual Representation for Classification-Rule Discovery

This Subsection assumes that the EA discovers classification rules of the form "IF (conditions) THEN (class)" (Witten & Frank 2005). This kind of knowledge representation has the advantage of being intuitively comprehensible to the user – an important point in data mining (Fayyad et al. 1996). A crucial issue in the design of an individual representation is to decide whether the candidate solution represented by an individual will be a rule set or just a single classification rule (Freitas 2002a, 2002b).

The former approach is often called the "Pittsburgh approach", whereas the later approach is often called the "Michigan-style approach". This latter term is an extension of the term "Michigan approach", which was originally used to refer to one particular kind of EA called Learning Classifier Systems (Smith 2000; Goldberg 1989). In this chapter we use the extended term "Michigan-style approach" because, instead of discussing Learning Classifier Systems, we discuss conceptually simpler EAs sharing the basic characteristic that an individual represents a single classification rule, regardless of other aspects of the EA.

The difference between the two approaches is illustrated in Figure 1. Figure 1(a) shows the Pittsburgh approach. The number of rules, m, can be either variable, automatically evolved by the EA, or fixed by a user-specified parameter. Figure 1(b) shows the Michigan-style approach, with a single rule per individual. In both Figure 1(a) and 1(b) the rule antecedent (the "IF part" of the rule) consists of a conjunction of conditions. Each condition is typically of the form <Attribute, Operator, Value>, also known as attribute-value (or propositional logic) representation. Examples are the conditions: "Gender = Female" and "Age $<$ 25". In the case of continuous attributes it is also common to have rule conditions of the form <LowerBound, Operator, Attribute, Operator, UpperBound>, e.g.: "30K $\leq$ Salary $\leq$ 50K".

In some EAs the individuals can only represent rule conditions with categorical (nominal) attributes such as Gender, whose values (male, female) have no ordering – so that the only operator used in the rule conditions is "$=$", and sometimes "$\neq$". When using EAs with this limitation, if the data set contains continuous attributes – with ordered numerical values – those attributes have to be discretized in a preprocessing stage, before the EA is applied. In practice it is desirable to use an EA where individuals can represent rule conditions with both categorical and continuous attributes. In this case the EA is effectively doing a discretization of continuous values "on-the-fly", since by creating rule conditions such as "30K $\leq$ Salary $\leq$ 50K" the EA is effectively producing discrete intervals. The effectiveness of an EA that directly copes with continuous attributes can be improved by using operators that enlarge or shrink the intervals based on concepts and methods borrowed from the research area of discretization in data mining (Divina & Marchiori 2005).

It is also possible to have conditions of the form <Attribute, Operator, Attribute>, such as "Income $>$ Expenditure". Such conditions are associated with relational (or first-order logic) representations. This kind of relational representation has considerably more expressiveness power than the conventional attribute-value representation, but the former is associated with a much larger search space – which often requires a more complex EA and a longer processing time. Hence, most EAs for rule discovery use the attribute-value, propositional representation. EAs using the relational, first-order logic representation are described, for instance, in (Neri & Giordana 1995; Hekanaho 1995; Woung & Leung 2000; Divina & Marchiori 2002).

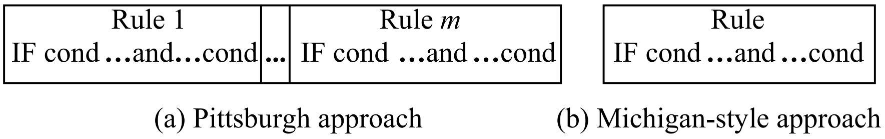


Fig. 1. Pittsburgh vs. Michigan-style approach for individual representation

Note that in Figure 1 the individuals are representing only the rule antecedent, and not the rule consequent (predicted class). It would be possible to include the predicted class in each individual's genome and let that class be evolved along with its corresponding rule antecedent. However, this approach has one significant drawback, which can be illustrated with the following example. Suppose an EA has just generated an individual whose rule antecedent covers 100 examples, 97 of which have class c_1. Due to the stochastic nature of the evolutionary process and the "blind-search" nature of the generic operators, the EA could associate that rule antecedent with class c_2, which would assign a very low fitness to that individual – a very undesirable result. This kind of problem can be avoided if, instead of evolving the rule consequent, the predicted class for each rule is determined by other (non-evolutionary) means. In particular, two such means are as follows.

First, one can simply assign to the individual the class of the majority of the examples covered by the rule antecedent (class c_1 in the above example), as a conventional, non-evolutionary rule induction algorithm would do. Second, one could use the "sequential covering" approach, which is often used by conventional rule induction algorithms (Witten & Frank 2005). In this approach, the EA discovers rules for one class at a time. For each class, the EA is run for as long as necessary to discover rules covering all examples of that class. During the evolutionary search for rules predicting that class, all individuals of the population will be representing rules predicting the same fixed class. Note that this avoids the problem of crossover mixing genetic material of rules predicting different classes, which is a potential problem in approaches where different individuals in the population represent rules predicting different classes. A more detailed discussion about how to represent the rule consequent in an EA can be found in (Freitas 2002a).

The main advantage of the Pittsburgh approach is that an individual represents a complete solution to a classification problem, i.e., an entire set of rules. Hence, the evaluation of an individual naturally takes into account rule interactions, assessing the quality of the rule *set*. In addition, the more complete information associated with each individual in the Pittsburgh approach can be used to design "intelligent", task-specific genetic operators. An example is the "smart" crossover operator proposed by (Bacardit & Krasnogor 2006), which heuristically selects, out of the N sets of rules in N parents (where $N \geq 2$), a good subset of rules to be included in a new child individual. The main disadvantage of the Pittsburgh approach is that it leads to long individuals and renders the design of genetic operators (that will act on selected individuals in order to produce new offspring) more difficult.

The main advantage of the Michigan-style approach is that the individual representation is simple, without the need for encoding multiple rules in an individual. This leads to relatively short individuals and simplifies the design of genetic operators. The main disadvantage of the Michigan-style approach is that, since each individual represents a single rule, a standard evaluation of the fitness of an individual ignores the problem of rule interaction. In the classification task, one usually wants to evolve a *good set* of rules, rather than a set of *good rules*. In other words, it is important to discover a rule set where the rules "cooperate" with each other. In particular, the rule set should cover the entire data space, so that each data instance should be covered by at least one rule. This requires a special mechanism to discover a diverse set of rules, since a standard EA would typically converge to a population where almost all the individuals would represent the same best rule found by the evolutionary process.

In general the previously discussed approaches perform a "direct" search for rules, consisting of initializing a population with a set of rules and then iteratively modifying those rules via the application of genetic operators. Due to a certain degree of randomness typically present in both initialization and genetic operations, some bad quality rules tend to be produced along the evolutionary process. Of course such bad rules are likely to be eliminated quickly by the selection process, but in any case an interesting alternative and "indirect" way of searching for rules has been proposed, in order to minimize the generation of bad rules. The basic idea of this new approach, proposed in (Jiao et al. 2006), is that the EA searches for good groups (clusters) of data instances, where each group consists of instances of the same class. A group is good to the extent that its data instances have similar attribute values and those attribute values are different from attribute values of the instances in other groups. After the EA run is over and good groups of instances have been discovered by the EA, the system extracts classification rules from the groups. This seems a promising new approach, although it should be noted that the version of the system described in (Jiao et al. 2006) has the limitation of coping only with categorical (not continuous) attributes.

In passing, it is worth mentioning that the above discussion on rule representation issues has focused on a generic classification problem. Specific kinds of classification problems may well be more effectively solved by EAs using rule representations "tailored" to the target kind of problem. For instance, (Hirsch et al. 2005) propose a rule representation tailored to document classification (i.e., a *text* mining problem), where strings of characters – in general fragments of words, rather than full words – are combined via Boolean operators to form classification rules.

3.2 Searching for a Diverse Set of Rules

This subsection discusses two mechanisms for discovering a diverse set of rules. It is assumed that each individual represents a single classification rule (Michigan-style approach). Note that the mechanisms for rule diversity discussed below are not normally used in the Pittsburgh approach, where an individual already represents a set of rules whose fitness implicitly depends on how well the rules in the set cooperate with each other.

First, one can use a niching method. The basic idea of niching is to avoid that the population converges to a single high peak in the search space and to foster the EA to create stable subpopulations of individuals clustered around each of the high peaks. In general the goal is to obtain a kind of "fitness-proportionate" convergence, where the size of the subpopulation around each peak is proportional to the height of that peak (i.e., to the quality of the corresponding candidate solution).

For instance, one of the most popular niching methods is fitness sharing (Goldberg & Richardson 1987; Deb & Goldberg 1989). In this method, the fitness of an individual is reduced in proportion to the number of similar individuals (neighbors), as measured by a given distance metric. In the context of rule discovery, this means that if there are many individuals in the current population representing the same rule or similar rules, the fitness of those individuals will be considerably reduced, and so they will have a considerably lower probability of being selected to produce new offspring. This effectively penalizes individuals which are in crowded regions of the search space, forcing the EA to discover a diverse set of rules.

Note that fitness sharing was designed as a generic niching method. By contrast, there are several niching methods designed specifically for the discovery of classification rules. An example is the "universal suffrage" selection method (Giordana et al. 1994; Divina 2005) where – using a political metaphor – individuals to be selected for reproduction are "elected" by the training data instances. The basic idea is that each data instance "votes" for a rule that covers it in a probabilistic fitness-based fashion. More precisely, let R be the set of rules (individuals) that cover a given data instance i, i.e., the set of rules whose antecedent is satisfied by data instance i. The better the fitness of a given rule r in the set R, the larger the probability that rule r will receive the vote of data instance i. Note that in general only rules covering the same data instances are competing with each other. Therefore, this selection method implements a form of niching, fostering the evolution of different rules covering different parts of the data space. For more information about niching methods in the context of discovering classification rules the reader is referred to (Hekanaho 1996; Dhar et al. 2000).

Another kind of mechanism that can be used to discover a diverse set of rules consists of using the previously-mentioned "sequential covering" approach – also known as "separate-and-conquer". The basic idea is that the EA discovers one rule at a time, so that in order to discover multiple rules the EA has to be run multiple times. In the first run the EA is initialized with the full training set and an empty set of rules. After each run of the EA, the best rule evolved by the EA is added to the set of discovered rules and the examples correctly covered by that rule are removed from the training set, so that the next run of the EA will consider a smaller training set. The process proceeds until all examples have been covered. Some examples of EAs using the sequential covering approach can be found in (Liu & Kwok 2000; Zhou et al. 2003; Carvalho & Freitas 2004). Note that the sequential covering approach is not specific to EAs. It is used by several non-evolutionary rule induction algorithms, and it is also discussed in data mining textbooks such as (Witten & Frank 2005).

3.3 Fitness Evaluation

One interesting characteristic of EAs is that they naturally allow the evaluation of a candidate solution, say a classification rule, as a whole, in a global fashion. This is in contrast with some data mining paradigms, which evaluate a partial solution. Consider, for instance, a conventional, greedy rule induction algorithm that incrementally builds a classification rule by adding one condition at a time to the rule. When the algorithm is evaluating several candidate conditions, the rule is still incomplete, being just a partial solution, so that the rule evaluation function is somewhat shortsighted (Freitas 2001, 2002a; Furnkranz & Flach 2003).

Another interesting characteristic of EAs is that they naturally allow the evaluation of a candidate solution by simultaneously considering different quality criteria. This is not so easily done in other data mining paradigms. To see this, consider again a conventional, greedy rule induction algorithm that adds one condition at a time to a candidate rule, and suppose one wants to favor the discovery of rules which are both accurate and simple (short). As mentioned earlier, when the algorithm is evaluating several candidate conditions, the rule is still incomplete, and so its size is not known yet. Hence, intuitively is better to choose the best candidate condition to be added to the rule based on a measure of accuracy only. The simplicity (size) criterion is better considered later, in a pruning procedure.

The fact that EAs evaluate a candidate solution as a whole and lend themselves naturally to simultaneously consider multiple criteria in the evaluation of the fitness of an individual gives the data miner a great flexibility in the design of the fitness function. Hence, not surprisingly, many different fitness functions have been proposed to evaluate classification rules. Classification accuracy is by far the criterion most used in fitness functions for evolving classification rules. This criterion is already extensively discussed in many good books or articles about classification, e.g. (Hand 1997; Caruana & Niculescu-Mizil 2004), and so it will not be discussed here – with the exception of a brief mention of overfitting issues, as follows. EAs can discover rules that overfit the training set – i.e. rules that represent very specific patterns in the training set that do not generalize well to the test set (which contains data instances unseen during training). One approach to try to mitigate the overfitting problem is to vary the training set at every generation, i.e., at each generation a subset of training instances is randomly selected, from the entire set of training instances, to be used as the (sub-)training or validation set from which the individuals' fitness values are computed (Bacardit et al. 2004; Pappa & Freitas 2006; Sharpe & Glover 1999; Bhattacharyya 1998). This approach introduces a selective pressure for evolving rules with a greater generalization power and tends to reduce the risk of overfitting, by comparison with the conventional approach of evolving rules for a training set which remains fixed throughout evolution. In passing, if the (sub)-training or validation set used for fitness computation is significantly smaller than the original training set, this approach also has the benefit of significantly reducing the processing time of the EA.

Hereafter this section will focus on two other rule-quality criteria (not based on accuracy) that represent different desirables properties of discovered rules in the context of data mining, namely: comprehensibility (Fayyad et al. 1996), or simplicity; and surprisingness, or unexpectedness (Liu et al. 1997; Romao et al. 2004; Freitas 2006).

The former means that ideally the discovered rule(s) should be comprehensible *to the user*. Intuitively, a measure of comprehensibility should have a strongly subjective, user-dependent component. However, in the literature this subjective component is typically ignored (Pazzani 2000; Freitas 2006), and comprehensibility is usually evaluated by a measure of the syntactic simplicity of the classifier, say the size of the rule set. The latter can be measured in an objective manner, for instance, by simply counting the total number of rule conditions in the rule set represented by an individual.

However, there is a natural way of incorporating a subjective measure of comprehensibility into the fitness function of an EA, namely by using an *interactive* fitness function. The basic idea of an interactive fitness function is that the user directly evaluates the fitness of individuals during the execution of the EA (Banzhaf 2000). The evaluation of the user is then used as the fitness measure for the purpose of selecting the best individuals of the current population, so that the EA evolves solutions that tend to maximize the subjective preference of the user.

An interactive EA for attribute selection is discussed e.g. in (Terano & Ishino 1998, 2002). In that work an individual represents a selected subset of attributes, which is then used by a classification algorithm to generate a set of rules. Then the user is shown the rules and selects good rules and rule sets according to her/his subjective preferences. Next the individuals having attributes that occur in the selected rules or rule sets are selected as parents to produce new offspring. The main advantage of interactive fitness functions is that intuitively they tend to favor the discovery of rules that are comprehensible and considered "good" by the user. The main disadvantage of this approach is that it makes the system considerably slower. To mitigate this problem one often has to use a small population size and a small number of generations.

Another kind of criterion that has been used to evaluate the quality of classification rules in the fitness function of EAs is the surprisingness of the discovered rules. First of all, it should be noted that accuracy and comprehensibility do not imply surprisingness. To show this point, consider the following classical hypothetical rule, which could be discovered from a hospital's database: IF (patient is pregnant) THEN (gender is female). This rule is very accurate and very comprehensible, but it is useless, because it represents an obvious pattern.

One approach to discover surprising rules consists of asking the user to specify a set of general impressions, specifying his/her previous knowledge and/or believes about the application domain (Liu et al. 1997). Then the EA can try to find rules that are surprising in the sense of contradicting some general impression specified by the user. Note that a rule should be reported to the user only if it is found to be both surprising and at least reasonably accurate (consistent with the training data). After all, it would be relatively easy to find rules which are surprising and inaccurate, but these rules would not be very useful to the user.

An EA for rule discovery taking this into account is described in (Romao et al. 2002, 2004). This EA uses a fitness function measuring both rule accuracy and rule surprisingness (based on general impressions). The two measures are multiplied to give the fitness value of an individual (a candidate prediction rule).

4 Evolutionary Algorithms for Clustering

There are several kinds of clustering algorithm, and two of the most popular kinds are iterative-partitioning and hierarchical clustering algorithms (Aldenderfer & Blashfield 1984; Krzanowski & Marriot 1995). In this section we focus mainly on EAs that can be categorized as iterative-partitioning algorithms, since most EAs for clustering seem to belong to this category.

4.1 Individual Representation for Clustering

A crucial issue in the design of an EA for clustering is to decide what kind of individual representation will be used to specify the clusters. There are at least three major kinds of individual representation for clustering (Freitas 2002a), as follows.

Cluster description-based representation – In this case each individual explicitly represents the parameters necessary to precisely specify each cluster. The exact nature of these parameters depends on the shape of clusters to be produced, which could be, e.g., boxes, spheres, ellipsoids, etc. In any case, each individual contains K sets of parameters, where K is the number of clusters, and each set of parameters determines the position, shape and size of its corresponding cluster. This kind of representation is illustrated, at a high level of abstraction, in Figure 2, for the case where an individual represents clusters of spherical shape. In this case each cluster is specified by its center coordinates and its radius. The cluster description-based representation is used, e.g., in (Srikanth et al. 1995), where an individual represents ellipsoid-based cluster descriptions; and in (Ghozeil and Fogel 1996; Sarafis 2005), where an individual represents hyperbox-shaped cluster descriptions. In (Sarafis 2005), for instance, the individuals represent rules containing conditions based on discrete numerical intervals, each interval being associated with a different attribute. Each clustering rule represents a region of the data space with homogeneous data distribution, and the EA was designed to be particularly effective when handling high-dimensional numerical datasets.

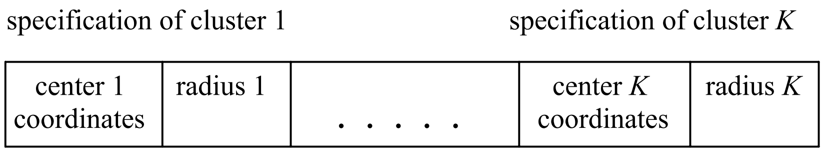


Fig. 2. Structure of cluster description-based individual representation

Centroid/medoid-based representation – In this case each individual represents the coordinates of each cluster's centroid or medoid. A centroid is simply a point in the data space whose coordinates specify the centre of the cluster. Note that there may not be any data instance with the same coordinates as the centroid. By contrast, a medoid is the most "central" representative of the cluster, i.e., it is the data instance which is nearest to the cluster's centroid. The use of medoids tends to be more robust against outliers than the use of centroids (Krzanowski & Marriot 1995) (p. 83). This kind of representation is used, e.g., in (Hall et al. 1999; Estivill-Castro and Murray 1997) and other EAs for clustering reviewed in (Sarafis 2005). This representation is illustrated, at a high level of abstraction, in Figure 3. Each data instance is assigned to the cluster represented by the centroid or medoid that is nearest to that instance, according to a given distance measure. Therefore, the position of the centroids/medoids and the procedure used to assign instances to clusters implicitly determine the precise shape and size of the clusters.

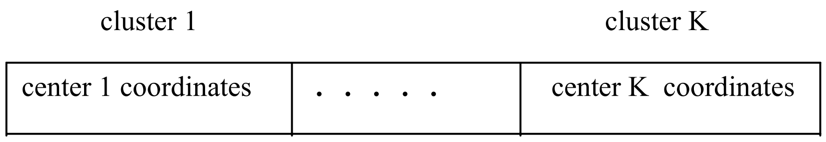


Fig. 3. Structure of centroid/medoid-based individual representation

Instance-based representation – In this case each individual consists of a string of n elements (genes), where n is the number of data instances. Each gene i, i=1,…,n, represents the index (id) of the cluster to which the i-th data instance is assigned. Hence, each gene i can take one out of Kvalues, where K is the number of clusters. For instance, suppose that n = 10 and K= 3. The individual <2 1 2 3 3 2 1 1 2 3> corresponds to a candidate clustering where the second, seventh and eighth instances are assigned to cluster 1, the first, third, sixth and ninth instances are assigned to cluster 2 and the other instances are assigned to cluster 3. This kind of representation is used, for instance, in (Krishma and Murty 1999; Handl & Knowles 2004). A variation of this representation is used in (Korkmaz et al. 2006), where the value of a gene represents not the cluster id of a gene's associated data instance, but rather a link from the gene's instance to another instance which is considered to be in the same cluster. Hence, in this approach, two instances belong to the same cluster if there is a sequence of links from one of them to the other. This variation is more complex than the conventional instance-based representation, and it has been proposed together with repair operators that rectify the contents of an individual when it violates some pre-defined constraints.

Comparing different individual representations for clustering – In both the centroid/medoid-based representation and the instance-based representation, each instance is assigned to exactly one cluster. Hence, the set of clusters determine a partition of the data space into regions that are mutually exclusive and exhaustive. This is not the case in the cluster description-based representation. In the latter, the cluster descriptions may have some overlapping – so that an instance may be located within two or more clusters – and the cluster descriptions may not be exhaustive – so that some instance(s) may not be within any cluster.

Unlike the other two representations, the instance-based representation has the disadvantage that it does not scale very well for large data sets, since each individual's length is directly proportional to the number of instances being clustered. This representation also involves a considerable degree of redundancy, which may lead to problems in the application of conventional genetic operators (Falkenauer 1998). For instance, let $n = 4$ and $K = 2$, and consider the individuals <1 2 1 2> and <2 1 2 1>. These two individuals have different gene values in all the four genes, but they represent the same candidate clustering solution, i.e., assigning the first and third instances to one cluster and assigning the second and fourth instances to another cluster. As a result, a crossover between these two parent individuals can produce two children individuals representing solutions that are very different from the solutions represented by the parents, which is not normally the case in conventional crossover operators used by genetic algorithms. Some methods have been proposed to try to mitigate some redundancy-related problems associated with this kind of representation. For example, (Handl & Knowles 2004) proposed a mutation operator that is reported to work well with this representation, based on the idea that, when a gene has its value mutated – meaning that the gene's corresponding data instance is moved to another cluster – the system selects a number of "nearest neighbors" of that instance and moves all those nearest neighbors to the same cluster to which the mutated instance was moved. Hence, this approach effectively incorporates some knowledge of the clustering task to be solved in the mutation operator.

4.2 Fitness Evaluation for Clustering

In an EA for clustering, the fitness of an individual is a measure of the quality of the clustering represented by the individual. A large number of different measures have been proposed in the literature, but the basic ideas usually involve the following principles. First, the smaller the intra-cluster (within-cluster) distance, the better the fitness. The intra-cluster distance can be defined as the summation of the distance between each data instance and the centroid of its corresponding cluster – a summation computed over all instances of all the clusters. Second, the larger the inter-cluster (between-cluster) distance, the better the fitness. Hence, an algorithm can try to find optimal values for these two criteria, for a given fixed number of clusters. These and other clustering-quality criteria are extensively discussed in the clustering literature – see e.g. (Aldenderfer and Blashfield 1984; Backer 1995; Tan et al. 2006). A discussion of this topic in the context of EAs can be found in (Kim et al. 2000; Handl & Knowles 2004; Korkmaz et al. 2006; Krishma and Murty 1999; Hall et al. 1999).

In any case, it is important to note that, if the algorithm is allowed to vary the number of discovered clusters without any restriction, it would be possible to minimize intra-cluster distance and maximize inter-cluster distance in a trivial way, by assigning each example to its own singleton cluster. This would be clearly undesirable. To avoid this while still allowing the algorithm to vary the number of clusters, a common response is to incorporate in the fitness function a preference for a smaller number of clusters. It might also be desirable or necessary to incorporate in the fitness function a penalty term whose value is proportional to the number of empty clusters (i.e. clusters to which no data instance was assigned) (Hall et al. 1999).

5 Evolutionary Algorithms for Data Preprocessing

5.1 Genetic Algorithms for Attribute Selection

In the attribute selection task the goal is to select, out of the original set of attributes, a subset of attributes that are relevant for the target data mining task (Liu & Motoda 1998; Guyon and Elisseeff 2003). This Subsection assumes the target data mining task is classification – which is the most investigated task in the evolutionary attribute selection literature – unless mentioned otherwise.

The standard individual representation for attribute selection consists simply of a string of N bits, where N is the number of original attributes and the i-th bit, i=1,. . . ,N, can take the value 1 or 0, indicating whether or not, respectively, the i-th attribute is selected. For instance, in a 10-attribute data set, the individual "1 0 1 0 1 0 0 0 0 1" represents a candidate solution where only the 1st, 3rd, 5th and 10th attributes are selected. This individual representation is simple, and traditional crossover and mutation operators can be easily applied. However, it has the disadvantage that it does not scale very well with the number of attributes. In applications with many thousands of attributes (such as text mining and some bioinformatics problems) an individual would have many thousands of genes, which would tend to lead to a slow execution of the GA.

An alternative individual representation, proposed by (Cherkauer & Shavlik 1996), consists of M genes (where M is a user-specified parameter), where each gene can contain either the index (id) of an attribute or a flag – say 0 – denoting no attribute. An attribute is considered selected if and only if it occurs in at least one of the M genes of the individual. For instance, the individual "3 0 8 3 0", where $M = 5$, represents a candidate solution where only the 3rd and the 8th attributes are selected. The fact that the 3rd attribute occurs twice in the previous individual is irrelevant for the purpose of decoding the individual into a selected attribute subset. One advantage of this representation is that it scales up better with respect to a large number of original attributes, since the value of M can be much smaller than the number of original attributes. One disadvantage is that it introduces a new parameter, M, which was not necessary in the case of the standard individual representation.

With respect to the fitness function, GAs for attribute selection can be roughly divided into two approaches – just like other kinds of algorithms for attribute selection – namely the wrapper approach and the filter approach. In essence, in the wrapper approach the GA uses the classification algorithm to compute the fitness of individuals, whereas in the filter approach the GA does not use the classification algorithm. The vast majority of GAs for attribute selection has followed the wrapper approach, and many of those GAs have used a fitness function involving two or more criteria to evaluate the quality of the classifier built from the selected attribute subset. This can be shown in Table 1, adapted from (Freitas 2002a), which lists the evaluation criteria used in the fitness function of a number of GAs following the wrapper approach. The columns of that table have the following meaning: *Acc* = accuracy; *Sens, Spec* = sensitivity, specificity; |Sel Attr| = number of selected attributes; |rule set| = number of discovered rules; *Info. Cont.* = information content of selected attributes; *Attr cost* = attribute costs; *Subj eval* = subjective evaluation of the user; |*Sel ins*| = number of selected instances.

Table 1. Diversity of criteria used in fitness function for attribute selection

Reference	Acc	Sens, Spec	|Sel Attr|	|rule set|	Info cont	Attr cost	Subj eval	|Sel ins|
(Bala et al. 1995)	yes		yes					
(Bala et al. 1996)	yes		yes		yes			
(Chen et al. 1999)	yes		yes					
(Cherkauer & Shavlik 1996)	yes		yes	yes				
(Emmanouilidis et al. 2000)	yes		yes					
(Emmanouilidis et al. 2002)		yes	yes					
(Guerra-Salcedo, Whitley 1998, 1999)	yes							
(Ishibuchi & Nakashima 2000)	yes		yes					yes
(Llora & Garrell 2003)	yes							
(Miller et al. 2003)		yes						
(Moser & Murty 2000)	yes		yes					
(Ni & Liu 2004)	yes							
(Pappa et al. 2002)	yes			yes				
(Rozsypal & Kubat 2003)	yes		yes					yes
(Terano & Ishino 1998)	yes			yes			yes	
(Vafaie & DeJong 1998)	yes							
(Yang & Honavar 1997, 1998)	yes					yes		
(Zhang et al 2003)	yes							

A precise definition of the terms used in the titles of the columns of Table 1 can be found in the corresponding references quoted in that table. The table refers to GAs that perform attribute selection for the classification task. GAs that perform attribute selection for the clustering task can be found, e.g., in (Kim et al. 2000; Jourdan 2003). In addition, in general Table 1 refers to GAs whose individuals directly represent candidate attribute subsets, but GAs can be used for attribute selection in other ways. For instance, in (Jong et al. 2004) a GA is used for attribute ranking. Once the ranking has been done, one can select a certain number of top-ranked attributes, where that number can be specified by the user or computed in a more automated way.

Empirical comparisons between GAs and other kinds of attribute selection methods can be found, for instance, in (Sharpe and Glover 1999; Kudo & Skalansky 2000). In general these empirical comparisons show that GAs, with their associated global search in the solution space, usually (though not always) obtain better results than local search-based attribute selection methods. In particular, (Kudo & Skalansky 2000) compared a GA with 14 non-evolutionary attribute selection methods (some of them variants of each other) across 8 different data sets. The authors concluded that the advantages of the global search associated with GAs over the local search associated with other algorithms is particularly important in data sets with a "large" number of attributes, where "large" was considered over 50 attributes in the context of their data sets.

5.2 Genetic Programming for Attribute Construction

In the attribute construction task the general goal is to construct new attributes out of the original attributes, so that the target data mining task becomes easier with the new attributes. This Subsection assumes the target data mining task is classification – which is the most investigated task in the evolutionary attribute construction literature.

Note that in general the problem of attribute construction is considerably more difficult than the problem of attribute selection. In the latter the problem consists just of deciding whether or not to select each attribute. By contrast, in attribute construction there is a potentially much larger search space, since there is a potentially large number of operations that can be applied to the original attributes in order to construct new attributes. Intuitively, the kind of EA that lends itself most naturally to attribute construction is GP. The reason is that, as mentioned earlier, GP was specifically designed to solve problems where candidate solutions are represented by both attributes and functions (operations) applied to those attributes. In particular, the explicit specification of both a terminal set and a function set is usually missing in other kinds of EAs.

Data Preprocessing vs. Interleaving Approach

In the data preprocessing approach, the attribute construction algorithm evaluates a constructed attribute without using the classification algorithm to be applied later. Examples of this approach are the GP algorithms for attribute construction proposed by (Otero et al. 2003; Hu 1998), whose attribute evaluation function (the fitness function) is the information gain ratio – a measure discussed in detail in (Quinlan 1993). In addition, (Muharram & Smith 2004) did experiments comparing the effectiveness of two different attribute-evaluation criteria in GP for attribute construction – viz. information gain ratio and gini index – and obtained results indicating that, overall, there was no significant difference in the results associated with those two criteria.

By contrast, in the interleaving approach the attribute construction algorithm evaluates the constructed attributes based on the performance of the classification algorithm with those attributes. Examples of this approach are the GP algorithms for attribute construction proposed by (Krawiec 2002; Smith and Bull 2003; Firpi et al. 2005), where the fitness functions are based on the accuracy of the classifier built with the constructed attributes.

Single-Attribute-per-Individual vs. Multiple-Attributes-per-Individual Representation

In several GPs for attribute construction, each individual represents a single constructed attribute. This approach is used for instance by CPGI (Hu 1998) and the GP algorithm proposed by (Otero et al. 2003). By default this approach returns to the user a single constructed attribute – the best evolved individual. However it can be extended to return to the user a set of constructed attributes, say returning a set of the best evolved individuals of a GP run or by running the GP multiple times and returning only the best evolved individual of each run. The main advantage of this approach is simplicity, but it has the disadvantage of ignoring interactions between the constructed attributes.

An alternative approach consists of associating with an individual a set of constructed attributes. The main advantage of this approach is that it takes into account interaction between the constructed attributes. In other words, it tries to construct the *best set* of attributes, rather than the set of *best attributes*. The main disadvantages are that the individuals' genomes become more complex and that it introduces the need for additional parameters such as the number of constructed attributes that should be encoded in one individual (a parameter that is usually specified in an ad-hoc fashion). In any case, the equivalent of this latter parameter would also have to be specified in the above-mentioned "extended version" of the single-attribute-per-individual approach when one wants the GP algorithm to return multiple constructed attributes.

Examples of this multiple-attributes-per-individual approach are the GP algorithms proposed by (Krawiec 2002; Smith & Bull 2003; Firpi et al. 2005). Here we briefly discuss the former two, as examples of this approach. In (Krawiec 2002) each individual encodes a fixed number K of constructed attributes, each of them represented by a tree, so that an individual consists of K trees – where K is a user-specified parameter. The algorithm also includes a method to split the constructed attributes encoded in an individual into two subsets, namely the subset of "evolving" attributes and the subset of "hidden" attributes. The basic idea is that high-quality constructed attributes are considered hidden (or "protected"), so that they cannot be manipulated by the genetic operators such as crossover and mutation. The choice of attributes to be hidden is based on an attribute quality measure. This measure evaluates the quality of each constructed attribute separately, and the best attributes of the individual are considered hidden.

Another example of the multiple-attributes-per-individual approach is the GAP (Genetic Algorithm and Programming) system proposed by (Smith & Bull 2003, 2004). GAP performs both attribute construction and attribute selection. The first stage consists of attribute construction, which is performed by a GP algorithm. As a result of this first stage, the system constructs an extended genotype containing both the constructed attributes represented in the best evolved individual of the GP run and original attributes that have not been used in those constructed attributes. This extended genotype is used as the basic representation for a GA that performs attribute selection, so that the GA searches for the best subset of attributes out of all (both constructed and original) attributes.

Satisfying the Closure Property

GP algorithms for attribute construction have used several different approaches to satisfy the closure property (briefly mentioned in Section 2). This is an important issue, because the chosen approach can have a significant impact on the types (e.g., continuous or nominal) of original attributes processed by the algorithm and on the types of attributes constructed by the algorithm. Let us see some examples.

A simple solution for the closure problem is used in the GAP algorithm (Smith and Bull 2003). Its terminal set contains only the continuous (real-valued) attributes of the data being mined. In addition, its function set consists only of arithmetic operators (+, –, *, %,) – where % denotes protected division, i.e. a division operator that handles zero denominator inputs by returning something different from an error (Banzhaf et al. 1998; Koza 1992) – so that the closure property is immediately satisfied. (Firpi et al. 2005) also uses the approach of having a function set consisting only of mathematical operators, but it uses a considerably larger set of mathematical operators than the set used by (Smith and Bull 2003).

The GP algorithm proposed by (Krawiec 2002) uses a terminal set including all original attributes (both continuous and nominal ones), and a function set consisting of arithmetical operators (+, –, *, %, log), comparison operators ($<$, $>$, $=$), an "IF (conditional expression)", and an "approximate equality operator" which compares its two arguments with tolerance given by the third argument. The algorithm did not enforce data type constraints, which means that expressions encoding the constructed attributes make no distinction between, for instance, continuous and nominal attributes. Values of nominal attributes, such as male and female, are treated as numbers. This helps to solve the closure problem, but at a high price: constructed attributes can contain expressions that make no sense from a semantical point of view. For instance, the algorithm could produce an expression such as "*Gender* + *Age*", because the value of the nominal attribute *Gender* would be interpreted as a number.

The GP proposed by (Otero et al. 2003) uses a terminal set including only the continuous attributes of the data being mined. Its function set consists of arithmetic operators (+, –, *, %,) and comparison operators ($\geq$, $\leq$). In order to satisfy the closure property, the algorithm enforces the data type restriction that the comparison operators can be used only at the root of the GP tree, i.e., they cannot be used as child nodes of other nodes in the tree. The reason is that comparison operators return a Boolean value, which cannot be processed by any operator in the function set (all operators accept only continuous values as input). Note that, although the algorithm can construct attributes only out of the continuous original attributes, the constructed attributes themselves can be either Boolean or continuous. A constructed attribute will be Boolean if its corresponding tree in the GP individual has a comparison operator at the root node; it will be continuous otherwise.

In order to satisfy the closure property, GPCI (Hu 1998) simply transforms all the original attributes into Boolean attributes and uses a function set containing only Boolean functions. For instance, if an attribute A is continuous (real-valued), such as the attribute *Salary*, it is transformed into two Boolean attributes, such as "Is $Salary > t$?" and "Is $Salary \leq t$?", where t is a threshold automatically chosen by the algorithm in order to maximize the ability of the two new attributes in discriminating between instances of different classes. The two new attributes are named "*positive-A*" and "*negative-A*", respectively. Once every original attribute has been transformed into two Boolean attributes, a GP algorithm is applied to the Boolean attributes. In this GP, the terminal set consists of all the pairs of attributes "*positive-A*" and "*negative-A*" for each original attribute A, whereas the function set consists of the Boolean operators {AND, OR}. Since all terminal symbols are Boolean, and all operators accept Boolean values as input and produce Boolean value as output, the closure property is satisfied.

Table 2 summarizes the main characteristics of the five GP algorithms for attribute construction discussed in this Section.

Table 2. Summary of GP Algorithms for Attribute Construction

Reference	Approach	Individual representation	Datatype of input attrib	Datatype of output attrib
(Hu 1998)	Data preprocessing	Single attribute	Any (attributes are booleanised)	Boolean
(Krawiec 2002)	Interleaving	Multiple attributes	Any (nominal attrib. values are interpreted as numbers)	Continuous
(Otero et al. 2003)	Data preprocessing	Single attribute	Continuous	Continuous or Boolean
(Smith & Bull 2003, 2004)	Interleaving	Multiple attributes	Continuous	Continuous
(Firpi et al. 2005)	Interleaving	Multiple attributes	Continuous	Continuous

6 Multi-Objective Optimization with Evolutionary Algorithms

There are many real-world optimization problems that are naturally expressed as the simultaneous optimization of two or more conflicting objectives (Coello Coello 2002; Deb 2001; Coello Coello & Lamont 2004). A generic example is to maximize the quality of a product and minimize its manufacturing cost in a factory. In the context of data mining, a typical example is, in the data preprocessing task of attribute selection, to minimize the error rate of a classifier trained with the selected attributes and to minimize the number of selected attributes.

The conventional approach to cope with such multi-objective optimization problems using evolutionary algorithms is to convert the problem into a single-optimization problem. This is typically done by using a weighted formula in the fitness function, where each objective has an associated weight reflecting its relative importance. For instance, in the above example of two-objective attribute selection, the fitness function could be defined as, say: "2/3 classification_error + 1/3 Number_of_selected_attributes".

However, this conventional approach has several problems. First, it mixes non-commensurable objectives (classification error and number of selected attributes in the previous example) into the same formula. This has at least the disadvantage that the value returned by the fitness function is not meaningful to the user. Second, note that different weights will lead to different selected attributes, since different weights represent different trade-offs between the two conflicting objectives. Unfortunately, the weights are usually defined in an ad-hoc fashion. Hence, when the EA returns the best attribute subset to the user, the user is presented with a solution that represents just one possible trade-off between the objectives. The user misses the opportunity to analyze different trade-offs.

Of course we could address this problem by running the EA multiple times, with different weights for the objectives in each run, and return the multiple solutions to the user. However, this would be very inefficient, and we would still have the problems of deciding which weights should be used in each run, how many runs we should perform (and so how many solutions should be returned to the user), etc.

A more principled approach consists of letting an EA answer these questions automatically, by performing a global search in the solution space and discovering as many good solutions, with as much diversity among them, as possible. This can be done by using a multi-objective EA, a kind of EA which has become quite popular in the EA community in the last few years (Deb 2001; Coello Coello 2002; Coello Coello & Lamont 2004). The basic idea involves the concept of Pareto dominance. A solution s_1 is said to dominate, in the Pareto sense, another solution s_2 if and only if solution s_1 is strictly better than s_2 in at least one of the objectives and solution s_1 is not worse than s_2 in any of the objectives. The concept of Pareto dominance is illustrated in Figure 4. This figure involves two objectives to be minimized, namely classification error and number of selected attributes (No_attrib). In that figure, solution D is dominated by solution B (which has both a smaller error and a smaller number of selected attributes than D), and solution E is dominated by solution C. Hence, solutions A, B and C are non-dominated solutions. They constitute the best "Pareto front" found by the algorithm. All these three solutions would be returned to the user.

The goal of a multi-objective EA is to find a Pareto front which is as close as possible to the true (unknown) Pareto front. This involves not only the minimization of the two objectives, but also finding a diverse set of non-dominated solutions, spread along the Pareto front. This allows the EA to return to the user a diverse set of good trade-offs between the conflicting objectives. With this rich information, the user can hopefully make a more intelligent decision, choosing the best solution to be used in practice.

At this point the reader might argue that this approach has the disadvantage that the final choice of the solution to be used depends on the user, characterizing a subjective approach. The response to this is that the knowledge discovery process is interactive (Brachman & Anand 1996; Fayyad et al. 1996), and the participation of the user in this process is important to obtain useful results. The questions are *when and how* the user should participate (Deb 2001; Freitas 2004). In the above-described multi-objective approach, based on Pareto dominance, the user participates by choosing the best solution out of all the non-dominated solutions. This choice is made *a posteriori*, i.e., after the algorithm has run and has returned a rich source of information about the solution space: the discovered Pareto front. In the conventional approach – using an EA with a weighted formula and returning a single solution to the user – the user has to define the weights *a priori*, i.e., before

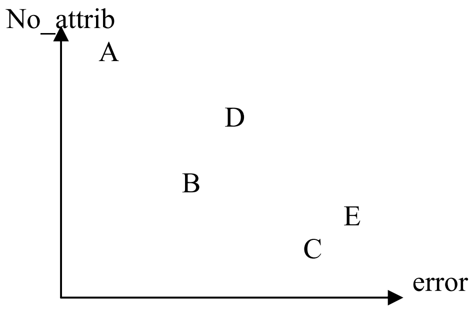


Fig. 4. Example of Pareto dominance

running the algorithm, when the solution space was not explored yet. The multi-objective approach seems to put the user in the loop in a better moment, when valuable information about the solution space is available. The multi-objective approach also avoids the problems of ad-hoc choice of weights, mixing non-commensurable objectives into the same formula, etc.

Table 3 lists the main characteristics of multi-objective EAs for data mining. Most systems included in Table 3 consider only two objectives. The exceptions are the works of (Kim et al. 2000) and (Atkinson-Abutridy et al. 2003), considering 4 and 8 objectives, respectively. Out of the EAs considering only two objectives, the most popular choice of objectives – particularly for EAs addressing the classification task – has been some measure of classification accuracy (or its dual, error) and a measure of the size of the classification model (number of leaf nodes in a decision tree or total number of rule conditions – attribute-value pairs – in all rules). Note that the size of a model is typically used as a proxy for the concept of "simplicity" of that model, even though arguably this proxy leaves a lot to be desired as a measure of a model's simplicity (Pazzani 2000; Freitas 2006). (In practice, however, it seems no better proxy for a model's simplicity is known.) Note also that, when the task being solved is attribute selection for classification, the objective related to size can be the number of selected attributes, as in (Emmanouilidis et al. 2000), or the size of the classification model built from the set of selected attributes, as in (Pappa et al. 2002, 2004). Finally, when solving the clustering task a popular choice of objective has been some measure of intra-cluster distance, related to the total distance between each data instance and the centroid of its cluster, computed for all data instances in all the clusters. The number of clusters is also used as an objective in two out of the three EAs for clustering included in Table 3. A further discussion of multi-objective optimization in the context of data mining in general (not focusing on EAs) is presented in (Freitas 2004; Jin 2006).

Table 3. Main characteristics of multi-objective EAs for data mining

Reference	Data mining task	Objectives being Optimized
(Emmanouilidis et al. 2000)	attribute selection for classification	accuracy, number of selected attributes
(Pappa et al 2002, 2004)	attribute selection for classification	accuracy, number of leafs in decision tree
(Ishibuchi & Namba 2004)	selection of classification rules	error, number of rule conditions (in all rules)
(de la Iglesia 2007)	selection of classification rules	confidence, coverage
(Kim et al. 2004)	classification	error, number of leafs in decision tree
(Atkinson-Abutridy et al. 2003)	text mining	8 criteria for evaluating explanatory knowledge across text documents
(Kim et al. 2000)	attribute selection for clustering	Cluster cohesiveness, separation between clusters, number of clusters, number of selected attributes
(Handl & Knowles 2004)	clustering	Intra-cluster deviation and connectivity
(Korkmaz et al. 2006)	clustering	Intra-cluster variance and number of clusters

7 Conclusions

This chapter started with the remark that EAs are a very generic search paradigm. Indeed, the chapter discussed how EAs can be used to solve several different data mining tasks, namely the discovery of classification rules, clustering, attribute selection and attribute construction. The discussion focused mainly on the issues of individual representation and fitness function for each of these tasks, since these are the two EA-design issues that are more dependent of the task being solved. In any case, recall that the design of an EA also involves the issue of genetic operators. Ideally these three components – individual representation, fitness function and genetic operators – should be designed in a synergistic fashion and tailored to the data mining task being solved.

There are at least two motivations for using EAs in data mining, broadly speaking. First, as mentioned earlier, EAs are robust, adaptive search methods that perform a global search in the solution space. This is in contrast to other data mining paradigms that typically perform a greedy search. In the context of data mining, the global search of EAs is associated with a better ability to cope with attribute interactions. For instance, most “conventional”, non-evolutionary rule induction algorithms are greedy, and therefore quite sensitive to the problem of attribute interaction. EAs can use the same knowledge representation (IF-THEN rules) as conventional rule induction algorithms, but their global search tends to cope better with attribute interaction and to discover interesting relationships that would be missed by a greedy search (Dhar et al. 2000; Papagelis & Kalles 2001; Freitas 2002a).

Second, EAs are a very flexible algorithmic paradigm. In particular, borrowing some terminology from programming languages, EAs have a certain "declarative" – rather than "procedural" – style. The quality of an individual (candidate solution) is evaluated, by a fitness function, in a way independent of how that solution was constructed. This gives the data miner a considerable freedom in the design of the individual representation, the fitness function and the genetic operators. This flexibility can be used to incorporate background knowledge into the EA and/or to hybridize EAs with local search methods that are specifically tailored to the data mining task being solved.

Note that declarativeness is a matter of degree, rather than a binary concept. In practice EAs are not 100% declarative, because as one changes the fitness function one might consider changing the individual representation and the genetic operators accordingly, in order to achieve the above-mentioned synergistic relationship between these three components of the EA. However, EAs still have a degree of declarativeness considerably higher than other data mining paradigms. For instance, as discussed in Subsection 3.3, the fact that EAs evaluate a complete (rather than partial) rule allows the fitness function to consider several different rule-quality criteria, such as comprehensibility, surprisingness and subjective interestingness to the user. In EAs these quality criteria can be directly considered during the search for rules. By contrast, in conventional, greedy rule induction algorithms – where the evaluation function typically evaluates a partial rule – those quality criteria would typically have to be considered in a post-processing phase of the knowledge discovery process, when it might be too late. After all, many rule set post-processing methods just try to select the most interesting rules out of all discovered rules, so that interesting rules that were missed by the rule induction method will remain missing after applying the post-processing method.

Like any other data mining paradigm, EAs also have some disadvantages. One of them is that conventional genetic operators – such as conventional crossover and mutation operators – are "blind" search operators in the sense that they modify individuals (candidate solutions) in a way independent from the individual's fitness (quality). This characteristic of conventional genetic operators increases the generality of EAs, but intuitively tends to reduce their effectiveness in solving a specific kind of problem. Hence, in general it is important to modify or extend EAs to use task specific-operators.

Another disadvantage of EAs is that they are computationally slow, by comparison with greedy search methods. The importance of this drawback depends on many factors, such as the kind of task being performed, the size of the data being mined, the requirements of the user, etc. Note that in some cases a relatively long processing time might be acceptable. In particular, several data mining tasks, such as classification, are typically an off-line task, and the time spent solving that task is usually less than 20% of the total time of the knowledge discovery process. In scenarios like this, even a processing time of hours or days might be acceptable to the user, at least in the sense that it is not the bottleneck of the knowledge discovery process.

In any case, if necessary the processing time of an EA can be significantly reduced by using special techniques. One possibility is to use parallel processing techniques, since EAs can be easily parallelized in an effective way (Cantu-Paz 2000; Freitas & Lavington 1998; Freitas 2002a). Another possibility is to compute the fitness of individuals by using only a subset of training instances – where that subset can be chosen either at random or using adaptive instance-selection techniques (Bhattacharyya 1998; Gathercole & Ross 1997; Sharpe & Glover 1999; Freitas 2002a).

An important research direction is to better exploit the power of Genetic Programming (GP) in data mining. Several GP algorithms for attribute construction were discussed in Subsection 5.2, and there are also several GP algorithms for discovering classification rules (Freitas 2002a; Wong & Leung 2000) or for classification in general (Muni et al. 2004; Song et al. 2005; Folino et al. 2006). However, the power of GP is still underexplored. Recall that the GP paradigm was designed to *automatically* discover computer *programs*, or *algorithms*, which should be *generic "recipes"* for solving a given kind of problem, and not to find the solution to one particular instance of that problem (like in most EAs). For instance, classification is a kind of problem, and most classification-rule induction algorithms are generic enough to be applied to different data sets (each data set can be considered just an instance of the kind of problem defined by the classification task). However, these generic rule induction algorithms have been *manually* designed by a human being. Almost all current GP algorithms for classification-rule induction are competing with conventional (greedy, non-evolutionary) rule induction algorithms, in the sense that both GP and conventional rule induction algorithms are discovering classification rules for a single data set at a time. Hence, the output of a GP for classification-rule induction is a set of rules for a given data set, which can be called a "program" or "algorithm" only in a very loose sense of these words.

A much more ambitious goal, which is more compatible with the general goal of GP, is to use GP to *automatically* discover a rule induction *algorithm*. That is, to perform *algorithm induction*, rather than rule induction. The first version of a GP algorithm addressing this ambitious task has been proposed in (Pappa & Freitas 2006), and an extended version of that work is described in detail in another chapter of this book (Pappa & Freitas 2007).

References

Aldenderfer MS & Blashfield RK (1984) *Cluster Analysis* (Sage University Paper Series on Quantitative Applications in the Social Sciences, No. 44) Sage Publications.

Atkinson-Abutridy J, Mellishm C, and Aitken S (2003) A semantically guided and domain-independent evolutionary model for knowledge discovery from texts. *IEEE Trans. Evolutionary Computation 7(6)*, 546-560.

Bacardit J, Goldberg DE, Butz MV, Llora X, Garrell JM (2004). Speeding-up Pittsburgh learning classifier systems: modeling time and accuracy. *Proc. Parallel Problem Solving From Nature (PPSN-2004), LNCS 3242,* 1021-1031, Springer.

Bacardit J and Krasnogor N (2006) Smart crossover operator with multiple parents for a Pittsburgh learning classifier system. *Proc. Genetic & Evolutionary Computation Conf. (GECCO-2006)*, 1441-1448. Morgan Kaufmann.

Backer E (1995) *Computer-Assisted Reasoning in Cluster Analysis.* Prentice-Hall.

Back T, Fogel DB and Michalewicz (Eds.) (2000) *Evolutionary Computation 1: Basic Algorithms and Operators.* Institute of Physics Publishing.

Bala J, De Jong K, Huang J, Vafaie H and Wechsler H (1995) Hybrid learning using genetic algorithms and decision trees for pattern classification. *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI-95)*, 719-724.

Bala J, De Jong K, Huang J, Vafaie H and Wechsler H (1996) Using learning to facilitate the evolution of features for recognizing visual concepts. *Evolutionary Computation 4(3)*: 297-312.

Banzhaf W (2000) Interactive evolution. In: T. Back, D.B. Fogel and T. Michalewicz (Eds.) *Evolutionary Computation 1*, 228-236. Institute of Physics Pub.

Banzhaf W, Nordin P, Keller RE, and Francone FD (1998) *Genetic Programming ~ an Introduction: On the Automatic Evolution of Computer Programs and Its Applications.* Morgan Kaufmann.

Bhattacharrya S (1998) Direct marketing response models using genetic algorithms. *Proceedings of the 4th Int. Conf. on Knowledge Discovery and Data Mining (KDD-98)*, 144-148. AAAI Press.

Brachman RJ and Anand T. (1996) The process of knowledge discovery in databases: a human-centered approach. In: U.M. Fayyad et al (Eds.) *Advances. in Knowledge Discovery and Data Mining*, 37-58. AAAI/MIT.

Bull L (Ed.) (2004) *Applications of Learning Classifier Systems.* Springer.

Bull L and Kovacs T (Eds.) (2005) *Foundations of Learning Classifier Systems.* Springer.

Cantu-Paz E (2000) *Efficient and Accurate Parallel Genetic Algorithms.* Kluwer.

Caruana R and Niculescu-Mizil A (2004) Data mining in metric space: an empirical analysis of supervised learning performance criteria. *Proc. 2004 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD-04)*, ACM.

Carvalho DR and Freitas AA (2004). A hybrid decision tree/genetic algorithm method for data mining. *Special issue on Soft Computing Data Mining, Information Sciences 163(1-3)*, pp. 13-35. 14 June 2004.

Chen S, Guerra-Salcedo C and Smith SF (1999) Non-standard crossover for a standard representation - commonality-based feature subset selection. *Proc. Genetic and Evolutionary Computation Conf. (GECCO-99)*, 129-134. Morgan Kaufmann.

Cherkauer KJ and Shavlik JW (1996). Growing simpler decision trees to facilitate knowledge discovery. *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD-96)*, 315-318. AAAI Press.

Coello Coello CA, Van Veldhuizen DA and Lamont GB (2002) *Evolutionary Algorithms for Solving Multi-Objective Problems.* Kluwer.

Coello Coello CA and Lamont GB (Ed.) (2004) *Applications of Multi-objective Evolutionary Algorithms.* World Scientific.

Deb K (2001) *Multi-Objective Optimization Using Evolutionary Algorithms.* Wiley.

Deb K and Goldberg DE (1989). An investigation of niche and species formation in genetic function optimization. *Proc. 2nd Int. Conf. Genetic Algorithms (ICGA-89)*, 42-49.

De Jong K (2006) *Evolutionary Computation: a unified approach.* MIT.

De la Iglesia B (2007) Application of multi-objective metaheuristic algorithms in data mining. *Proc. 3rd UK Knowledge Discovery and Data Mining Symposium (UKKDD-2007)*, 39-44, University of Kent, UK, April 2007.

Dhar V, Chou D and Provost F (2000). Discovering interesting patterns for investment decision making with GLOWER – a genetic learner overlaid with entropy reduction. *Data Mining and Knowledge Discovery 4(4)*, 251-280.

Divina F (2005) Assessing the effectiveness of incorporating knowledge in an evolutionary concept learner. *Proc. EuroGP-2005 (European Conf. on Genetic Programming), LNCS 3447,* 13-24, Springer.

Divina F & Marchiori E (2002) Evolutionary Concept Learning. *Proc. Genetic & Evolutionary Computation Conf. (GECCO-2002)*, 343-350. Morgan Kaufmann.

Divina F & Marchiori E (2005) Handling continuous attributes in an evolutionary inductive learner. *IEEE Trans. Evolutionary Computation, 9(1),* 31-43, Feb. 2005.

Eiben AE and Smith JE (2003) *Introduction to Evolutionary Computing.* Springer.

Emmanouilidis C, Hunter A and J. MacIntyre J (2000) A multiobjective evolutionary setting for feature selection and a commonality-based crossover operator. *Proc. 2000 Congress on Evolutionary Computation (CEC-2000)*, 309-316. IEEE.

Emmanouilidis C (2002) Evolutionary multi-objective feature selection and ROC analysis with application to industrial machinery fault diagnosis. In: K. Giannakoglou et al. (Eds.) *Evolutionary Methods for Design, Optimisation and Control.* Barcelona: CIMNE.

Estivill-Castro V and Murray AT (1997) Spatial clustering for data mining with genetic algorithms. *Tech. Report FIT-TR-97-10.* Queensland University of Technology. Australia.

Falkenauer E (1998) *Genetic Algorithms and Grouping Problems.* John-Wiley & Sons.

Fayyad UM, Piatetsky-Shapiro G and Smyth P (1996) From data mining to knowledge discovery: an overview. In: U.M. Fayyad et al (Eds.) *Advances in Knowledge Discovery and Data Mining*, 1-34. AAAI/MIT.

Firpi H, Goodman E, Echauz J (2005) On prediction of epileptic seizures by computing multiple genetic programming artificial features. *Proc. 2005 European Conf. on Genetic Programming (EuroGP-2005), LNCS 3447,* 321-330. Springer.

Folino G, Pizzuti C and Spezzano G (2006) GP ensembles for large-scale data classification. *IEEE Trans. Evolutionary Computation 10(5)*, 604-616, Oct. 2006.

Freitas AA and. Lavington SH (1998) *Mining Very Large Databases with Parallel Processing.* Kluwer.

Freitas AA (2001) Understanding the crucial role of attribute interaction in data mining. *Artificial Intelligence Review 16(3)*, 177-199.

Freitas AA (2002a) *Data Mining and Knowledge Discovery with Evolutionary Algorithms.* Springer.

Freitas AA (2002b) A survey of evolutionary algorithms for data mining and knowledge discovery. In: A. Ghosh and S. Tsutsui. (Eds.) *Advances in Evolutionary Computation*, pp. 819-845. Springer-Verlag.

Freitas AA (2002c). Evolutionary Computation. In: W. Klosgen and J. Zytkow (Eds.) *Handbook of Data Mining and Knowledge Discovery*, pp. 698-706.Oxford Univ. Press.

Freitas AA (2004) A critical review of multi-objective optimization in data mining: a position paper. *ACM SIGKDD Explorations, 6(2),* 77-86, Dec. 2004.

Freitas AA (2005) Evolutionary Algorithms for Data Mining. In: O. Maimon and L. Rokach (Eds.) *The Data Mining and Knowledge Discovery Handbook*, pp. 435-467. Springer.

Freitas AA (2006) Are we really discovering "interesting" knowledge from data? *Expert Update, Vol. 9, No. 1,* 41-47, Autumn 2006.

Furnkranz J and Flach PA (2003). An analysis of rule evaluation metrics. *Proc.20th Int. Conf. Machine Learning (ICML-2003).* Morgan Kaufmann.

Gathercole C and Ross P (1997) Tackling the Boolean even N parity problem with genetic programming and limited-error fitness. *Genetic Programming 1997: Proc. 2nd Conf. (GP-97)*, 119-127. Morgan Kaufmann.

Ghozeil A and Fogel DB (1996) Discovering patterns in spatial data using evolutionary programming. *Genetic Programming 1996: Proceedings of the 1st Annual Conf.*, 521-527. MIT Press.

Giordana A, Saitta L, Zini F (2004) Learning disjunctive concepts by means of genetic algorithms. *Proc. 10th Int. Conf. Machine Learning (ML-94)*, 96-104. Morgan Kaufmann.

Goldberg DE (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley.

Goldberg DE and Richardson J (1987) Genetic algorithms with sharing for multimodal function optimization. *Proc. Int. Conf. Genetic Algorithms (ICGA-87)*, 41-49.

Guerra-Salcedo C and Whitley D (1998) Genetic search for feature subset selection: a comparison between CHC and GENESIS. *Genetic Programming 1998: Proc. 3rd Annual Conf.*, 504-509. Morgan Kaufmann.

Guerra-Salcedo C, Chen S, Whitley D, and Smith S (1999) Fast and accurate feature selection using hybrid genetic strategies. *Proc. Congress on Evolutionary Computation (CEC-99)*, 177-184. IEEE.

Guyon I and Elisseeff A (2003) An introduction to variable and feature selection. *Journal of Machine Learning Research 3*, 1157-1182.

Hall LO, Ozyurt IB, Bezdek JC (1999) Clustering with a genetically optimized approach. *IEEE Trans. on Evolutionary Computation 3(2)*, 103-112.

Hand DJ (1997) *Construction and Assessment of Classification Rules.* Wiley.

Handl J and Knowles J (2004) Evolutionary multiobjective clustering. *Proc. Parallel Problem Solving From Nature (PPSN-2004), LNCS 3242,* 1081-1091, Springer.

Hekanaho J (1995) Symbiosis in multimodal concept learning. *Proc. 1995 Int. Conf. on Machine Learning (ML-95)*, 278-285. Morgan Kaufmann.

Hekanaho J (1996) Testing different sharing methods in concept learning. *TUCS Technical Report No. 71.* Turku Centre for Computer Science, Finland.

Hirsch L, Saeedi M and Hirsch R (2005) Evolving rules for document classification. *Proc. 2005 European Conf. on Genetic Programming (EuroGP-2005)*, *LNCS 3447,* 85-95, Springer.

Hu YJ (1998). A genetic programming approach to constructive induction. *Genetic Programming 1998: Proc. 3rd Annual Conf.*, 146-151. Morgan Kaufmann.

Ishibuchi H and Nakashima T (2000) Multi-objective pattern and feature selection by a genetic algorithm. *Proc. 2000 Genetic and Evolutionary Computation Conf. (GECCO-2000)*, 1069-1076. Morgan Kaufmann.

Ishibuchi H and Namba S (2004) Evolutionary multiobjective knowledge extraction for high-dimensional pattern classification problems. *Proc. Parallel Problem Solving From Nature (PPSN-2004), LNCS 3242,* 1123-1132, Springer.

Jiao L, Liu J and Zhong W (2006) An organizational coevolutionary algorithm for classification. *IEEE Trans. Evolutionary Computation, Vol. 10, No. 1,* 67-80, Feb. 2006.

Jin, Y (Ed.) (2006) *Multi-Objective Machine Learning.* Springer.

Jong K, Marchiori E and Sebag M (2004) Ensemble learning with evolutionary computation: application to feature ranking. *Proc. Parallel Problem Solving from Nature VIII (PPSN-2004), LNCS 3242*, 1133-1142. Springer, 2004.

Jourdan L, Dhaenens-Flipo C and Talbi EG (2003) Discovery of genetic and environmental interactions in disease data using evolutionary computation. In: G.B. Fogel and D.W. Corne (Eds.) *Evolutionary Computation in Bioinformatics*, 297-316. Morgan Kaufmann.

Kim Y, Street WN and Menczer F (2000) Feature selection in unsupervised learning via evolutionary search. *Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD-2000)*, 365-369. ACM.

Kim D (2004). Structural risk minimization on decision trees: using an evolutionary multiobjective algorithm. *Proc. 2004 European Conference on Genetic Programming (EuroGP-2004), LNCS 3003,* 338-348, Springer.

Korkmaz EE, Du J, Alhajj R and Barker (2006) Combining advantages of new chromosome representation scheme and multi-objective genetic algorithms for better clustering. *Intelligent Data Analysis 10* (2006),163-182.

Koza JR (1992) *Genetic Programming: on the programming g of computers by means of natural selection.* MIT Press.

Krawiec K (2002) Genetic programming-based construction of features for machine learning and knowledge discovery tasks. *Genetic Programming and Evolvable Machines 3(4)*, 329-344.

Krsihma K and Murty MN (1999) Genetic k-means algorithm. *IEEE Transactions on Systems, Man and Cyberneics - Part B: Cybernetics,* 29(3), 433-439.

Krzanowski WJ and Marriot FHC (1995) *Kendall's Library of Statistics 2: Multivariate Analysis - Part 2. Chapter 10 - Cluster Analysis*, pp. 61-94.London: Arnold.

Kudo M and Sklansky J (2000) Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition 33(2000)*, 25-41.

Liu JJ and Kwok JTY (2000) An extended genetic rule induction algorithm. *Proc. 2000 Congress on Evolutionary Computation (CEC-2000).* IEEE.

Liu H and Motoda H (1998) *Feature Selection for Knowledge Discovery and Data Mining.* Kluwer.

Liu B, Hsu W and Chen S (1997) Using general impressions to analyze discovered classification rules. *Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD-97)*, 31-36. AAAI Press.

Llora X and Garrell J (2003) Prototype induction and attribute selection via evolutionary algorithms. *Intelligent Data Analysis 7*, 193-208.

Miller MT, Jerebko AK, Malley JD, Summers RM (2003) Feature selection for computer-aided polyp detection using genetic algorithms. *Medical Imaging 2003: Physiology and Function: methods, systems and applications.* Proc. SPIE Vol. 5031.

Moser A and Murty MN (2000) On the scalability of genetic algorithms to very large-scale feature selection. *Proc. Real-World Applications of Evolutionary Computing (EvoWorkshops 2000). LNCS 1803*, 77-86. Springer.

Muharram MA and Smith GD (2004) Evolutionary feature construction using information gain and gene index. *Genetic Programming: Proc. 7th European Conf. (EuroGP-2003), LNCS 3003*, 379-388. Springer.

Muni DP, Pal NR and Das J (2004) A novel approach to design classifiers using genetic programming. *IEEE Trans. Evolutionary Computation 8(2)*, 183-196, April 2004.

Neri F and Giordana A (1995) Search-intensive concept induction. *Evolutionary Computation 3(4)*, 375-416.

Ni B and Liu J (2004) A novel method of searching the microarray data for the best gene subsets by using a genetic algorithms. *Proc. Parallel Problem Solving From Nature (PPSN-2004), LNCS 3242,* 1153-1162, Springer.

Otero FB, Silva MMS, Freitas AA and Nievola JC (2003) Genetic programming for attribute construction in data mining. *Genetic Programming: Proc. EuroGP-2003, LNCS 2610*, 384-393. Springer.

Papagelis A and Kalles D (2001) Breeding decision trees using evolutionary techniques. *Proc. 18th Int. Conf. Machine Learning (ICML-2001),* 393-400. Morgan Kaufmann.

Pappa GL and Freitas AA (2006) Automatically evolving rule induction algorithms. *Machine Learning: ECML 2006 – Proc. of the 17th European Conf. on Machine Learning, LNAI 4212,* 341-352. Springer.

Pappa GL and Freitas AA (2007) Discovering new rule induction algorithms with grammar-based genetic programming. *Maimon O and Rokach L (Eds.) Soft Computing for Knowledge Discovery and Data Mining.* Springer.

Pappa GL, Freitas AA and Kaestner CAA (2002) A multiobjective genetic algorithm for attribute selection. *Proc. 4th Int. Conf. On Recent Advances in Soft Computing (RASC-2002)*, 116-121. Nottingham Trent University, UK.

Pappa GL, Freitas AA and Kaestner CAA (2004) Multi-Objective Algorithms for Attribute Selection in Data Mining. In: Coello Coello CA and Lamont GB (Ed.) *Applications of Multi-objective Evolutionary Algorithms*, 603-626. World Scientific.

Pazzani MJ (2000) Knowledge discovery from data, *IEEE Intelligent Systems,* 10-13, Mar./Apr. 2000.

Quinlan JR. (1993) *C4.5: Programs for Machine Learning.* Morgan Kaufmann.

Romao W, Freitas AA and Pacheco RCS (2002) A Genetic Algorithm for Discovering Interesting Fuzzy Prediction Rules: applications to science and technology data. *Proc. Genetic and Evolutionary Computation Conf. (GECCO-2002)*, pp. 1188-1195. Morgan Kaufmann.

Romao W, Freitas AA, Gimenes IMS (2004) Discovering interesting knowledge from a science and technology database with a genetic algorithm. *Applied Soft Computing 4(2)*, pp. 121-137.

Rozsypal A and Kubat M (2003) Selecting representative examples and attributes by a genetic algorithm. *Intelligent Data Analysis 7*, 290-304.

Sarafis I (2005) *Data mining clustering of high dimensional databases with evolutionary algorithms*. PhD Thesis, School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, UK.

Sharpe PK and Glover RP (1999) Efficient GA based techniques for classification. *Applied Intelligence 11*, 277-284.

Smith RE (2000) Learning classifier systems. In: T. Back, D.B. Fogel and T. Michalewicz (Eds.) *Evolutionary Computation 1: Basic Algorithms and Operators*, 114-123. Institute of Physics Publishing.

Smith MG and Bull L (2003) Feature construction and selection using genetic programming and a genetic algorithm. *Genetic Programming: Proc. EuroGP-2003, LNCS 2610*, 229-237. Springer.

Smith MG and Bull L (2004) Using genetic programming for feature creation with a genetic algorithm feature selector. *Proc. Parallel Problem Solving From Nature (PPSN-2004), LNCS 3242,* 1163-1171, Springer.

Song D, Heywood MI and Zincir-Heywood AN (2005) Training genetic programming on half a million patterns: an example from anomaly detection. *IEEE Trans. Evolutionary Computation 9(3)*, 225-239, June 2005.

Srikanth R, George R, Warsi N, Prabhu D, Petry FE, Buckles B (1995) A variable-length genetic algorithm for clustering and classification. *Pattern Recognition Letters 16(8)*, 789-800.

Tan PN, Steinbach M and Kumar V (2006) *Introduction to Data Mining.* Addison-Wesley.

Terano T and Ishino Y (1998) Interactive genetic algorithm based feature selection and its application to marketing data analysis. In: Liu H and Motoda H (Eds.) *Feature Extraction, Construction and Selection: a data mining perspective*, 393-406. Kluwer.

Terano T and Inada M (2002) Data mining from clinical data using interactive evolutionary computation. In: A. Ghosh and S. Tsutsui (Eds.) *Advances in Evolutionary Computing: theory and applications*, 847-861. Springer.

Vafaie H and De Jong K (1998) Evolutionary Feature Space Transformation. In: H. Liu and H. Motoda (Eds.) *Feature Extraction, Construction and Selection*, 307-323. Kluwer.

Witten IH and Frank E (2005) *Data Mining: practical machine learning tools and techniques* . 2nd Ed. Morgan Kaufmann.

Wong ML and Leung KS (2000) *Data Mining Using Grammar Based Genetic Programming and Applications.* Kluwer.

Yang J and Honavar V (1997) Feature subset selection using a genetic algorithm. *Genetic Programming 1997: Proc. 2nd Annual Conf. (GP-97)*, 380-385. Morgan Kaufmann.

Yang J and Honavar V (1998) Feature subset selection using a genetic algorithm. In: Liu, H. and Motoda, H (Eds.) *Feature Extraction, Construction and Selection*, 117-136. Kluwer.

Zhang P, Verma B, Kumar K (2003) Neural vs. Statistical classifier in conjunction with genetic algorithm feature selection in digital mammography. *Proc. Congress on Evolutionary Computation (CEC-2003)*. IEEE Press.

Zhou C, Xiao W, Tirpak TM and Nelson PC (2003) Evolving accurate and compact classification rules with gene expression programming. *IEEE Trans. on Evolutionary Computation 7(6)*, 519-531.

Genetic Clustering for Data Mining

Murilo Coelho Naldi[1]
André Carlos Ponce de Leon Ferreira de Carvalho[1]
Ricardo José Gabrielli Barreto Campello[1]
Eduardo Raul Hruschka[2]

Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
`murilocn@icmc.usp.br`, `andre@icmc.usp.br`, `campello@icmc.usp.br`,
`eduardo.hruschka@pesquisador.cnpq.br`

Summary. Genetic Algorithms (GAs) have been successfully applied to several complex data analysis problems in a wide range of domains, such as image processing, bioinformatics, and crude oil analysis. The need for organizing data into categories of similar objects has made the task of clustering increasingly important to those domains. In this chapter, the authors present a survey of the use of GAs for clustering applications. A variety of encoding (chromosome representation) approaches, fitness functions, and genetic operators are described, all of them customized to solve problems in such an application context.

1 Introduction

Clustering is one of the main tasks in Machine Learning, being usually employed when none or little information on the dataset is available. Intuitively, clustering is based on an inductive principle where objects within a cluster are more similar to each other then objects belonging to different clusters. This inductive principle is regarded as the objective function of clustering algorithms. The association of this objective function with a dataset creates an optimization problem (Jain *et al.*, 1999), whose goal depends on the validation employed. The partitions obtained by a clustering algorithm depend on the validation function adopted and the values assigned to the algorithm free-parameters. For some algorithms, the order of presentation of the examples can also affect the partitions produced.

The values for the free parameters are usually defined by trial and error, which may be computationally prohibitive. Finding an optimal or near optimal solution to the problem of partitioning n objects into k clusters has been shown to be *NP*-complete (Kaufman and Rousseeuw, 1990). Therefore, more sophisticated search techniques are necessary in order to be able to find a suitable set of values for the free parameters in a reasonable processing time.

Genetic Algorithms (GAs) are population-based search techniques that combine features of selected solutions in order to evolve them towards a global optimum.

GAs have been successfully used in many different tasks, including clustering. This survey will be organized in a framework describing the alternatives followed by different authors for each aspect of this application. For the application of GAs to a clustering problem, it is necessary to determine the representation of the possible solutions, how these solutions will be evaluated (the fitness function), the genetic operators employed to manipulate these solutions and the values of the free parameters (population size and application rate of the genetic operators). These subjects will be discussed in sections 2, 4 and 5, respectively.

1.1 A Brief Look at Genetic Algorithms

GAs are search and optimization methods inspired by the process of evolution of biological organisms. According to Charles Darwin in *The Origin of Species* (Darwin, 2006), organisms evolve by principles of natural selection and survival of the fittest organisms. John Holland's group from the University of Michigan introduced GAs in the middle of 1976 (Holland, 1975). However, its full use only started almost ten years later (Goldberg, 1989).

In a few words, a GA uses a population of individuals to solve a given problem. Each individual of the population corresponds to a possible solution for the problem. A reproduction based mechanism is applied to the current population, generating a new population. The population usually evolves through several generations until a suitable solution is reached.

According to (Goldberg, 1989), GAs differ from traditional methods of search and optimization mainly in four aspects:

- they can work with a code of the set of parameters and not necessarily with the own parameters;
- they work with several possible solutions and not with a single solution point;
- they use cost information or reward functions and not derivative or other auxiliary knowledge;
- they use probabilistic rules of transition instead of deterministic rules.

GAs start generating an initial population formed by a random group of individuals, which can be seen as first guesses to solve the problem. Supposing that a solution of a problem can be represented by a set of parameters, such parameters are coded into an individual by using a data structure called chromosome (in general a vector or a bit string). The chromosome is composed by a string of genes, where each gene is a coded parameter. The codification of parameters is defined by the programmer.

The initial population is evaluated and, for each individual, a score (named fitness) is given, reflecting the quality of the solution associated to it. In function optimization problems, the fitness is usually equal to the (raw or scaled) value of the objective function of the problem.

By mimicking the "natural selection", a GA probabilistically selects the best individuals whereas the worst are discarded. The selected individuals can be modified by genetic operators such as crossover and mutation, generating descendants for the next generation. This process is named reproduction. The evolutionary procedure is repeated until the population converges to a unique solution that is likely to be an optimal solution. Figure 1 presents a general diagram of a GA life cycle.

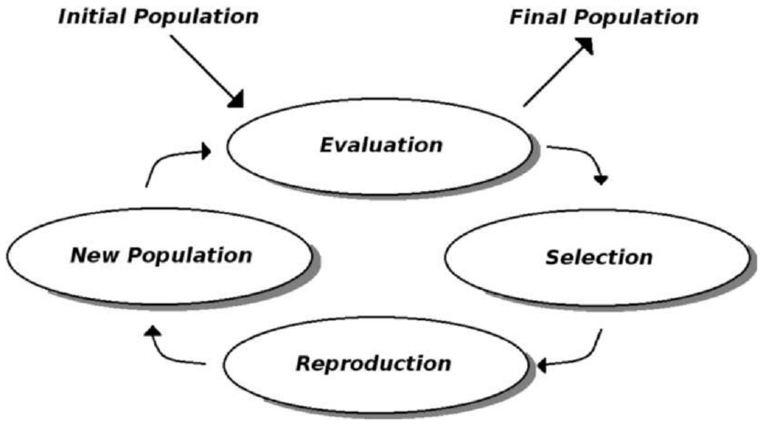


Fig. 1. Genetic Algorithm cycle.

GAs operate on a population of candidates in parallel. Thus, they can simultaneously search different regions of the solutions space. While selection drives the population into the direction of better solutions, crossover and mutation explore new solutions (i.e. new areas of the search space).

Different methods have been proposed to select individuals from a population. The most commonly used method is the roulette wheel sampling. In this method, each individual from the population occupies an area of the roulette wheel proportional to its fitness value. If P individuals are to be selected, the roulette wheel randomly spins P times. For each run, the individual pointed by the roulette wheel is selected. As a result, fitter individuals have higher chances of being selected.

Although largely used, the roulette wheel method does not work with negative fitness values and the expected number of children from a same parent suffers high variance. This problem can be overcome by the Stochastic Universal Sampling (SUS) (Baker, 1987), which ensures a selection of offspring which is closer to what is deserved than roulette wheel selection. The individuals are mapped to contiguous segments, such that each individual's segment is equal in size to its fitness exactly as in roulette-wheel selection. Equally spaced pointers are placed over the segments as many as there are individuals to be selected. Consider P the number of individuals to be selected, then the distance between the pointers are $1/P$ and the position of the first pointer is given by a randomly generated number in the range $[0, 1/P]$, the position of the second pointer is given by a randomly generated number in the range $[1/P, 2/P]$ and so on. SUS samples individuals by spinning the roulette P times for each equally-spaced pointers, instead of spinning randomly for all the segments as occurs in the roulette wheel method.

Other alternative is tournament selection (Mitchell, 1999), in which T (usually, $T = 2$) chromosomes are randomly selected from the population, with the same probability. These chromosomes compete against each other and the chromosome with the highest fitness value is selected.

The crossover operation exchanges parts of a pair of chromosomes, creating new chromosomes called children or offspring (If crossover does not succeed in the probability test, then the children will be identical copies of their parents). Figure 2 illustrates a crossover operation.

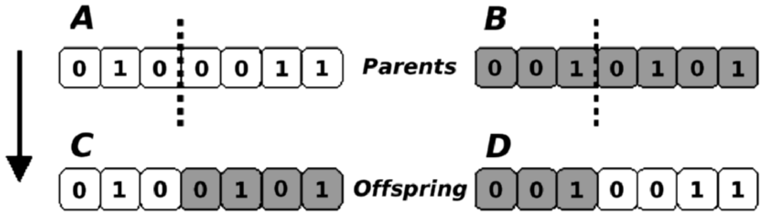


Fig. 2. Crossover operation.

In Figure 2, a cut point is randomly chosen and two chromosomes, A and B, contribute with a subset of their genes for the creation of the offspring C and D. There are several variations of crossover operators, such as the two cut point crossover and the uniform crossover. When the two point crossover is used, the segments between the two randomly chosen cut points are exchanged between the two parents (Goldberg, 1989, Mitchell, 1999), as can be seen in Figure 3.

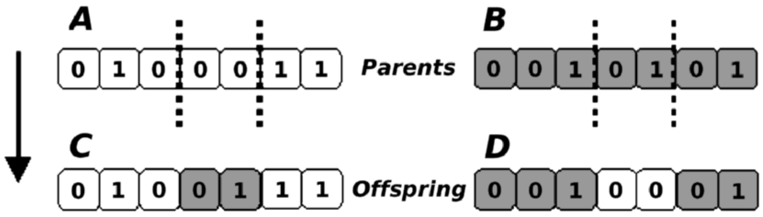


Fig. 3. Two point crossover operation.

For the uniform crossover, a mask is used to define from which parent each offspring inherits each of its genes (Goldberg, 1989, Mitchell, 1999).

The mutation operator aims to increase the variability of the population, allowing the GA to simultaneously search different areas of the solution space. This operator changes at random the value of a chromosome gene, also randomly chosen with a given probability (named mutation rate). Figure 4 shows a mutation operation that changes the value of the forth gene from 0 to 1.

After the creation of the children, the new population is obtained by replacing the original parents by their children. The usual approach involves replacing all parents. This approach is called generational replacement, which combined with elitism gives better results (Goldberg, 1989). An elitist policy means never replacing

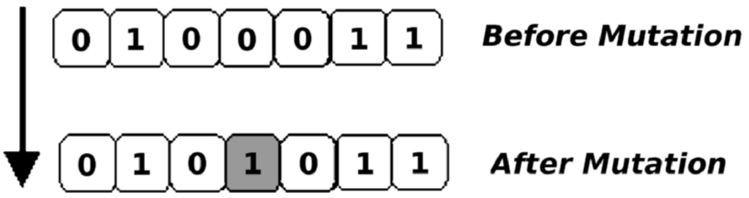


Fig. 4. Mutation operation.

the best parent, or set of parents, by any children worse (e.g. with lower aptitude) than them (Mitchell, 1999).

It's worth noting that the raw fitness (i.e. the fitness obtained directly from the objective function or cost function) may cause problems in most real world problems. Such as, for instance, premature convergence and low resolution of selection in later stages of evolution (when many individuals have high fitness values). Scaling and ranking adjust raw fitness more gently. Scaling adjusts the raw fitness using a linear function $a + b\dot{f}_{raw}$, where a and b are constants. Ranking sorts chromosomes best-to-worst fashion and assign fitness by interpolating the best (rank = 1) individual to the worst (rank = P) according to some function, usually linear or exponential. The ranked fitness of the ith individual using a linear function is given by:

$$min + (max - min)\frac{P - rank(i)}{P - 1} \tag{1}$$

This ranking requires $1 \leq max \leq 2$ and $min + max = 2$.

2 Representation

Clustering algorithms can be divided into two main categories: partitional and hierarchical. Partitional clustering algorithms identify the partition that optimizes a given clustering criterion (Jain *et al.*, 1999). Hierarchical clustering algorithms produce a nested series of partitions based on a criterion for either merging (agglomerative algorithms) or splitting (divisive algorithms) clusters based on a similarity measure. Hierarchical clusters may be represented by a *dendrogram*, showing the nested grouping of objects and similarity levels at which groupings change (Jain *et al.*, 1999). An example of a partition (a) and a *dendrogram* (b) for the objects A, B, C, D, E, F, G, is shown in Figure 5.

When applied to partitional clustering problems, some GAs can search for the number of clusters that best fits the dataset structure. As described in Section 1.1, each possible solution is represented in a GA by a vector (chromosome) of numeric values (genes). Different representations have been proposed for clustering using GAs (Cole, 1998). The most frequently used representations for partitional algorithms are:

- **Group-Number**: It is an encoding scheme in which a chromosome is an integer vector of n positions, where n is the number of dataset objects or objects. Each position corresponds to an object, i.e., the ith position (gene) represents the

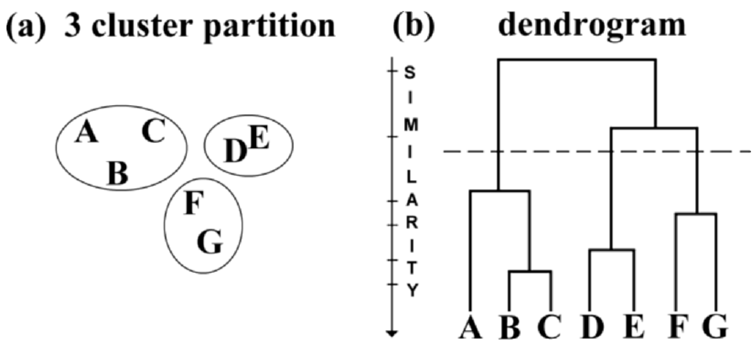


Fig. 5. Main cluster types: (a) partitional and (b) hierarchical.

ith dataset object. Provided that a genotype represents a partition formed by k clusters, each gene has a value from the alphabet 1,2,3,...,k. These values define the cluster labels (Krovi, 1992, Murthy and Chowdhury, 1996, Cowgill *et al.*, 1998). The same encoding scheme is used in (Hruschka and Ebecken, 2003, Hruschka *et al.*, 2004), but the authors additionally propose to store the number of clusters k in the genotype.

- **Binary Matrix**: In this case, the chromosome is represented by a $k \times n$ matrix of binary values, where k is the number of clusters and n is the number of objects in the dataset. If the value of the matrix position $P(C, x)$ is 1, the object x belongs to cluster C. Otherwise, it does not belong to this cluster (Bezdek *et al.*, 1994).
- **Centroids and Medoids**: The chromosomes are made up of real numbers that represent the coordinates of the cluster centers (Scheunders, 1997, Fränti *et al.*, 1997, Maulik and Bandyopadhyay, 2000, Merz and Zell, 2002, Kivijärvi *et al.*, 2003). If a genotype i encodes k clusters in a d dimensional space, its length is $d \times k$. Lucasius et al. (Lucasius *et al.*, 1993) proposed a related representation, which is based on the position of k selected objects (named medoids) from the dataset. Given the set of these medoids, k clusters are formed by assigning the remaining $(n-k)$ objects to the nearest medoid. Thus, each partition is encoded with a string of k different integers from $1, ..., n$. These integers correspond to the objects according to the order they appear in the dataset. The same representation scheme is adopted in (Estivill-Castro, 1997, Hall *et al.*, 1999, Sheng and Liu, 2004)
- **Labels**: Ma et al. (Ma *et al.*, 2006) proposed an evolutionary algorithm for clustering, named EvoCluster, which encodes a partition in such a way that each gene represents one cluster and contains the labels of the objects grouped into it. Thus, a genotype encoding k clusters $(C_1, C_2, ..., C_k)$ of a dataset with n objects is formed by k genes, each of which stores l_i labels $(l_1 + l_2 + ... + l_k = n)$. The they claim that this encoding scheme is an advantageous alternative over other different schemes. In particular, they argue that the group-number encoding, where each object is encoded as a gene and given a label from 1 to k, is not very scalable since the length of each genotype is exactly the number of objects of the training set. Although this assertion is persuasive at a first glance, it is worth noticing that the amount of information that must be stored (and handled) in both encoding schemes previously described is essentially the same, that is,

n object labels (EvoCluster's encoding) or n cluster labels (Group-Number). Therefore, the scalability of EvoCluster in terms of memory requirement does not benefit from its encoding scheme. Actually, the encoding scheme does not seem to be a crucial aspect regarding the practical usefulness of an algorithm when handling large data sets. In the end, the data set itself must be handled somehow (e.g. using efficient data structures for external memory management) and its dimensionality is necessarily larger than that of any encoding scheme.

Figure 6 presents the chromosome for the clusters $\{\{A, C, F\}, \{B, D, E\}\}$, using each of the representations described. The clusters are labeled 1 and 2 and the instances are labeled in the range between A and E. In Figure 6, the chromosome (c) represents two centroids, z_1 and z_2, and their respective hypothetical attribute values from a_1 to a_d.

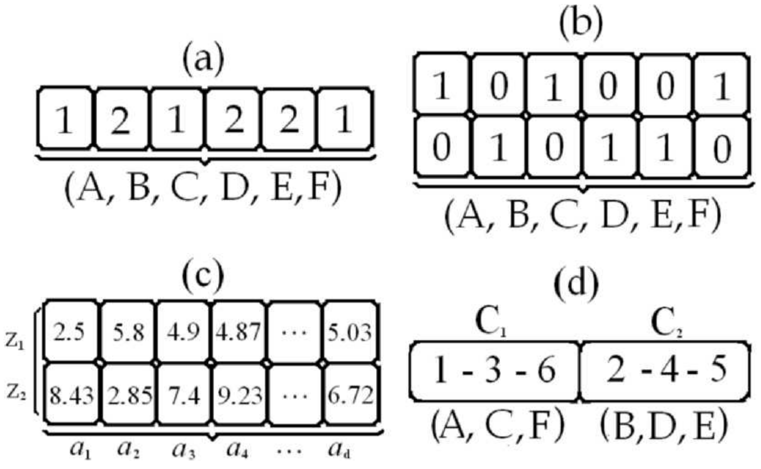


Fig. 6. Examples of (a)Group-Number, (b)Binary Matrix, (c)Centroids and (d) using labels.

In hierarchical clustering, the hierarchy, represented by the *dendrogram*, can be broken at different levels to yield different clustering partitions of the dataset. Thus, a hierarchical cluster can be represented by a set of partitions with its respective representation. Figure 7 shows two possible representations for the *dendrogram* in Figure 5: a set of partitions, represented by Group-Number, and an object oriented approach representation proposed by Greene (Greene, 2003). In the first representation (a), each line is associated with one possible level of partition and each column with a object. The second representation (b) indicates the clusters of the *dendrogram* as nodes in a graph. The cluster associated with each node contains the children clusters and belongs to the cluster represented by its parent node. This relationship is represented by the edges of the graph.

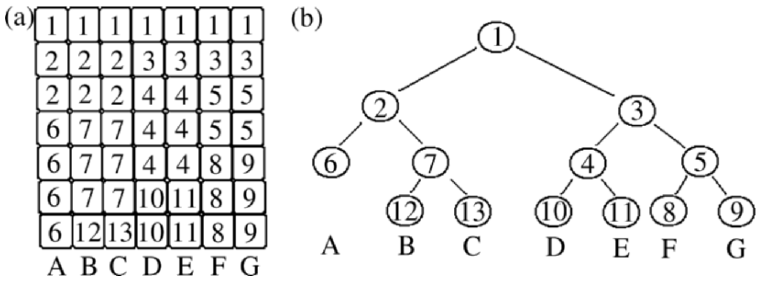


Fig. 7. Hierarchical representations using a set of partitions (a) and object orientation (b).

Another representation, named Cluster Description-Based Representation, was proposed by Freitas (Freitas, 2005). In this representation, each chromosome specifies a set of parameters that precisely specify each cluster. These parameters are related with the shape of the clusters produced.

Most representations show some form of redundancy, as different chromosomes may represent the same partition. Unfeasible solutions can be eliminated or remunerated by pos-processing (Belew and Booker, 1991). Korkmaz et al. (Korkmaz *et al.*, 2006) propose to avoid redundancy by using an encoding scheme in which each gene is a link from an object to another object of the same cluster.

3 Initialization

Many authors build the initial population of the genetic algorithm from random assignments of objects to clusters (Krovi, 1992,Murthy and Chowdhury, 1996,Cowgill *et al.*, 1998,Hruschka and Ebecken, 2003,Hruschka *et al.*, 2004,Ma *et al.*, 2006). Lucasius et al. (Lucasius *et al.*, 1993) suggest to randomly selecting a subset of objects to be the medoids of the initial population when prior knowledge is not available. Similarly, in (Kuncheva and Bezdek, 1997,Estivill-Castro, 1997,Maulik and Bandyopadhyay, 2000,Merz and Zell, 2002,Sheng and Liu, 2004) an initialization scheme is adopted that randomly chooses dataset objects to be initial prototypes of the clusters. The initial centers of the clusters can also be randomly generated (Scheunders, 1997,Fränti *et al.*, 1997,Kivijärvi *et al.*, 2003).

In Tseng and Yang (Tseng and Yang, 2001), the population of strings is randomly generated. The number of $1's$ in the binary strings is uniformly distributed within $[1, k]$, where k is the number of clusters initially generated. Some authors use heuristics to find good initial partitions and avoid invalid clusters (Bezdek *et al.*, 1994).

4 Fitness Function

A fitness function must be defined for the evaluation of the chromosomes. This function is based on the objective function used by traditional clustering algorithms.

Since the objective function has to be rescaled, the fitness function is very often a composition of the objective function and a scaling function (Grefenstette, 2000).

One of the most commonly used fitness function consists on minimizing the sum of squared Euclidean distances of the objects to their respective cluster mean (centroids) (Murthy and Chowdhury, 1996, Maulik and Bandyopadhyay, 2000, Merz and Zell, 2002). This fitness function $f(C_1, C_2, ..., C_k)$ can be formally described by:

$$f(C_1, C_2, ..., C_k) = \sum_{j=1}^{k} \sum_{x \in C_j} ||x - z_j||^2 \tag{2}$$

where $(C_1, C_2, ..., C_k)$ is the set of k clusters encoded by the genotype, x is a dataset object, and z_j is the mean vector (centroid) of cluster C_j. Similarly, the fitness functions used in the genetic algorithms described in (Fränti *et al.*, 1997, Kivijärvi *et al.*, 2003) aim to minimize the distortion in the clusters. The minimization of such distortion is equivalent to minimize $f(C_1, C_2, ..., C_k)$ defined in Equation (2). More precisely, the distortion dst is a measure of the intra-cluster diversity, which can be defined as:

$$dst = \frac{f(C_1, C_2, ..., C_k)}{n \times m} \tag{3}$$

where n and m are the numbers of objects and attributes, respectively. Adopting $f(C_1, C_2, ..., C_k)$ defined in Equation (2) and assuming a dataset formed by n objects, the fitness function employed in (Scheunders, 1997) can be written as:

$$f_m(C_1, C_2, ..., C_k) = \frac{n}{f(C_1, C_2, ..., C_k)} \tag{4}$$

Similar to what is carried out with centroids, the minimization of the distances of the k medoids to all the corresponding objects of the same cluster was proposed also in (Lucasius *et al.*, 1993). Functions presented above are monotonic with the number of cluster and does not optimize it.

Alternative validation criteria have also been used as fitness functions. The validation criteria are in general statistical indexes employed to evaluate the quality of a given partition. Three different approaches can be followed internal, external and relative:

- Internal criteria: measure the quality of a partition using only the original dataset. They measure how well the clusters obtained represent the similarities present in the dataset. A fitness function based on Euclidian distance is an example of internal criterion.
- External criteria: these criteria evaluate the partitions according to a predefined structure, based on what is known about the dataset. This predefined structure can be either a known partition for the dataset or a partition defined a specialist in the data domain.
- Relative criteria: They are employed to compare two or more clustering techniques regarding a particular aspect. They can be used, for example, to compare different clustering algorithms or runs of the same algorithm with different parameter values.

In principle, any relative clustering validity criterion (Jain and Dubes, 1988, Milligan and Cooper, 1985, Halkidi *et al.*, 2001) that is not monotonic with the number of clusters can be potentially used as a fitness function for a genetic algorithm designed to optimize the number of clusters. These criteria have been extensively studied, and some of them have shown good results for several applications. This fact has motivated their use as fitness functions, as it will be seen in this survey. However, it is worth mentioning that the particular features of a given relative validity criterion can make its performance problem dependent (Pal and Bezdek, 1995). The following criteria can optimize the final partition's cluster number.

Variation Ratio Criteria (VRC) (Calinski and Harabasz, 1974) and Silhouette (Rousseeuw, 1987) are two popular relative criteria choices when clustering is combined with GAs (Cowgill *et al.*, 1998, Casillas *et al.*, 2003, Pan *et al.*, 2003, Hruschka and Ebecken, 2003, Hruschka *et al.*, 2004). The values produced by these criteria are independent of the cluster algorithm used and can be employed to estimate the natural number of clusters in a dataset (Milligan and Cooper, 1985).

VRC is based on internal cluster cohesion and external cluster isolation. The internal cohesion is calculated by the within-group sum of square distances (WGSS) and the external isolation by the between-groups sum of square distances (BGSS) (Duda *et al.*, 2001), given by:

$$WGSS = \sum_{i=1}^{n} \sum_{j=i+1}^{n} D_{ij} \tag{5}$$

where n is the total number of objects and i and j are objects with $i \in C$ and $j \in C$, for all clusters C, D_{ij} is the dissimilarity between objects i and j, and

$$BGSS = \sum_{i=1}^{n} \sum_{j=i+1}^{n} D_{ij} \tag{6}$$

where i and j are objects with $i \in C$ and $j \notin C$, for all clusters C. The VRC criterion is given by:

$$VRC = \frac{BGSS}{(k-1)} / \frac{WGSS}{(n-k)} \tag{7}$$

with k being the total number of clusters and n the total number of objects.

Silhouette is based on the distance between objects from the same cluster and their distance to the closest cluster. Consider an object x belonging to a cluster C_a. Let the average dissimilarity of x to all other objects of C_a be denoted by $a(x)$. Next, let the average dissimilarity of x to all objects of a cluster C be represented by $d(x, C)$. After computing $d(x, C)$ for all clusters $C \neq C_a$, the smallest value, $b(x)$, is selected, where $b(x) = \min d(x, C) \forall C \neq C_a$. Thus, the silhouette for object x is given by:

$$s(x) = \begin{cases} 1 - a(x)/b(x), & a(x) < b(x) \\ 0, & a(x) = b(x) \\ b(x)/a(x) - 1, & a(x) > b(x) \end{cases} \tag{8}$$

It is easy to verify that $-1 \leq s(x) \leq 1$. This measure is appropriate when the values of the different attributes exhibit similar inferior and superior limits and the true clusters are compact and disjoint (Rousseeuw, 1987). In addition, if $s(x)$ is equal to zero, then it is not clear whether the instance should have been assigned to its current cluster or to a neighboring one (Everitt *et al.*, 2001). Finally, if the cluster is a singleton, then $s(x)$ is not defined and the most neutral choice is to set

$s(x) = 0$ (Kaufman and Rousseeuw, 1990). The silhouette criterion is given by the average of $s(i)$ over $i = 1, 2, ..., n$.

Two additional validity indexes to guide the genetic search were proposed by Hruschka et al. (Hruschka *et al.*, 2004), one of them is a simplified version of the silhouette. This criterion is based on the computation of distances between objects and cluster centroids, which are the mean vectors of the clusters. More specifically, the term $a(x)$ of Equation (8) becomes the dissimilarity of object x to the centroid of its cluster C_a. Similarly, instead of computing $d(x, C)$ as the average dissimilarity of x to all objects of C, $C \neq C_a$, only the distance between x and the centroid of C must be computed. Alternatively to the original and simplified versions of the silhouette, Hruschka et al. (Hruschka *et al.*, 2004) have shown that the fitness function can be taken as the average of $b(x)/(a(x) + \varepsilon)$ over $i = 1, 2, ..., n$, using the centroid based terms $a(x)$ and $b(x)$ just described. The term ε is necessary to compute $s(x)$ when $a(x)$ is zero, i.e., when all objects of cluster C_a are equal to each other. This modified objective function seems to be more sensitive to slight changes in $a(x)$ and $b(x)$, which in turn may correspond to significant changes in the clustering solution.

S. Bandyopadhyay and U. Maulik (Bandyopadhyay *et al.*, 2001) proposed a validity index $I(k)$ for computing the fitness of a genotype that represents k clusters, that is defined as:

$$I(k) = \left(\frac{1}{k} . \frac{E_1}{E_k} . D_k\right)^p \tag{9}$$

where p is any real number larger than or equal to 1, E_j and D_j are given by the following equations, respectively:

$$E_j = \sum_{j=1}^{k} \sum_{i=1}^{n} \mu_{ji} ||x_i - z_j|| \tag{10}$$

$$D_j = \max_{l,w=1}^{k} ||z_l - z_w|| \tag{11}$$

where n is the total number of objects in the dataset, $[\mu_{ji}]_{k \times n}$ is a partition matrix for the dataset $D = \{x_1, ..., x_n\}$, and z_w is the center of the w^{th} cluster. They report some experiments in which $I(k)$ provides better results than the indexes proposed in (Davies *et al.*, 1979) and (Dunn *et al.*, 1973), which are commonly used as relative validity criteria for clustering. However, in a more recent work (Bandyopadhyay and Maulik, 2002), they decided to use a fitness function based on the Davis-Bouldin (DB) (Davies *et al.*, 1979) index. The DB index for the partitioning of n objects into k clusters is defined as:

$$DB = \frac{1}{k} \sum_{i=1}^{k} R_{C_i} \tag{12}$$

The index for the ith cluster, R_{C_i}, is given by:

$$R_{C_i} = \max_{j \neq i} \{R_{j,i}\} \tag{13}$$

and $R_{j,i}$ measure the within-to-between cluster spread for all pairs of clusters (j, i):

$$R_{j,i} = \frac{e_j + e_i}{\upsilon_{j,i}} \tag{14}$$

where e_j is the within cluster variation for the jth cluster and $\upsilon_{j,i}$ is the distance between the centers of the j^{th} and the i^{th} clusters.

The fitness function adopted in EvoCluster (Ma *et al.*, 2006) has been conceived to deal with noisy and missing data, as well as to recognize interesting feature values for the clustering process. The fitness of each genotype is assessed by means of two main steps. The first step is dedicated to the discovery of statistically significant association of objects in the partition encoded in the genotype. To this end, some objects from different clusters are randomly selected to form a training set for object discovery. Let a be an attribute, a_j the j^{th} value this attribute takes in the dataset, and obs_{ij} the total number of objects in the dataset that belong to cluster C_i and are characterized by the same attribute value a_j. According to Ma et al. (Ma *et al.*, 2006), $exp_{ij} = (obs_{i+}) \cdot (obs_{+j})/n'$ is the expected number of objects under the assumption that being a member of C_i is independent of whether a object has the value a_j, where obs_{i+}, obs_{+j}, and n' are given by:

$$obs_{i+} = \sum_{j=1}^{g} obs_{ij} \tag{15}$$

$$obs_{+j} = \sum_{i=1}^{k} obs_{ij} \tag{16}$$

$$n' = \sum_{i,j} obs_{ij} \leq n \tag{17}$$

where g is the total number of distinct values for a, k is the number of clusters encoded by the genotype and n is the number of objects. The statistical significance of the association can be evaluated by:

$$z_{ij} = \frac{obs_{ij} - exp_{ij}}{\sqrt{exp_{ij}}} \tag{18}$$

where the maximum likelihood estimate of its asymptotic variance υ_{ij} is defined by:

$$\upsilon_{ij} = \left(1 - \frac{obs_{i+}}{n'}\right)\left(1 - \frac{obs_{+j}}{n'}\right) \tag{19}$$

Then $sd_{ij} = z_{ij}/(\upsilon_{ij})^{1/2}$ has an approximately standard normal distribution and the attribute value can be selected based on a statistically significant cluster dependency. In step 2 of the fitness computation, an uncertainty measure, named weight of evidence, W, is calculated for the values a_j associated with the cluster C_i at a given confidence level. The value of W for an object characterized by a_j to belong to C_i and not to other clusters is given by:

$$W\left(cluster = C_i/cluster \neq C_i | a_j\right) = L\left(C_i : a_j\right) - L\left(\neq C_i : a_j\right) \tag{20}$$

where $L(C_i : a_j)$ is given by:

$$L(C_i : a_j) = log \frac{P(C_i|a_j)}{P(C_i)} \tag{21}$$

For a collection of selected attribute values, the weight of evidence from all observed values is defined as:

$$W\left(cluster = C_i/cluster \neq C_i|a_1...a_j...a_{g'}\right) = \sum_{j=1}^{g'} W\left(cluster = C_i/cluster \neq C_i|a_j\right) \tag{22}$$

where g' is the number of selected attribute values. Cluster C_i is inferred if W is maximized. Thus, the predicted value can be compared with the original label of each object encoded in the genotype to determine the reclassification accuracy of the objects not selected in step 1. Finally, the fitness value is calculated based on this accuracy. The authors (Ma *et al.*, 2006) claim that EvoCluster does not require the number of clusters k to be defined in advance. However, this aspect was not fully investigated in the reported experimental evaluation, in which a set of interesting values for k was chosen a priori.

Validation criteria are independent of the representation and the cluster algorithm used.

5 Genetic Operators

Genetic operators are responsible for the modification of the individuals from one population to the next. Such operators may include, for instance, the exchange of parts of the parents, thus allowing the production of new solutions sharing features from both parents. By creating new solutions, genetic operators expand the exploration of the search space, making it possible to reach any of its regions. The main genetic operators are selection crossover and mutation. When GAs are used for clustering, the traditional genetic operators may need to be adapted to fit the chosen clustering representation.

5.1 Selection

As chromosomes are selected based on their relative fitness, the selection type is independent of representation or clustering algorithm. Proportional selection has been used by several authors (Krovi, 1992, Lucasius *et al.*, 1993, Murthy and Chowdhury, 1996, Estivill-Castro, 1997, Cowgill *et al.*, 1998, Maulik and Bandyopadhyay, 2000, Kivijärvi *et al.*, 2003). The simplest approach for proportional selection is the roulette wheel method, described in Section 1.1. Another popular choice is tournament selection, also described in Section 1.1.

Additionally to proportional selection, elitist variants for selecting genotypes are also investigated in (Murthy and Chowdhury, 1996, Fränti *et al.*, 1997, Kivijärvi *et al.*, 2003). A particular kind of elitist strategy is adopted in (Kuncheva and Bezdek, 1997), where the parents and the children are pooled and the best genotypes survive, composing the new population. Similarly, Merz and Zell (Merz and Zell, 2002) derive a new population by selecting the best genotypes out of the pool of parents and children. These selection methods can be viewed as variants of the so-called $(\mu + \lambda)$ selection procedure used in evolution strategies.

5.2 Crossover

Traditional crossover operators act at the chromosome level, ignoring the true shape of the clusters. This may result, for instance, in offspring whose representation is very similar to the parents, but whose corresponding clusters are very different. Therefore, the incorporation of context sensitivity to the crossover operators applied to clustering problems is commonly used, although it is not compulsory (Cowgill *et al.*, 1998, Bandyopadhyay and Maulik, 2002).

The main idea behind context sensitivity is to create crossover operators that can transmit clusters (integrally or partially) from the parents to the offspring, preserving their building blocks.

An example of this type of crossover is the edge-based crossover (Belew and Booker, 1991). In this operator, after the selection of parent two chromosomes, their objects are selected based on the edge of the clusters they belong to. Two objects are connected by the same edge if they are present in the same cluster in both chromosomes. The resulting children are composed of the non-empty intersections of their parent clusters. An example of two parent chromosomes and a possible child chromosome is showed on Figure 8, where cluster $\{C, D\}$ is copied from PC_1, $\{A, E\}$ from PC_2 and $\{B, F\}$ from both parents.

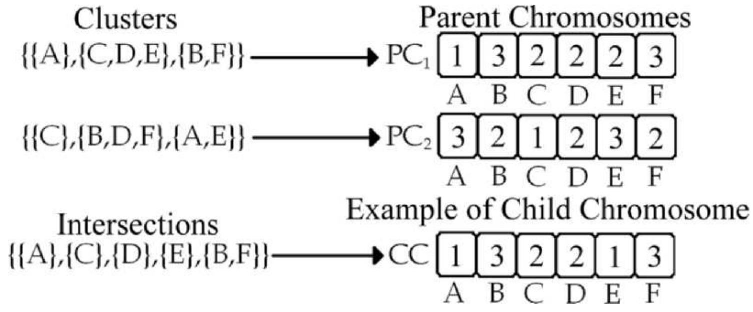


Fig. 8. Two partitions and their Group-Number representations.

Another context-sensitive crossover operator is proposed by Hruschka and Ebecken (Hruschka and Ebecken, 2003) and combines clustering solutions coming from different genotypes. After the selection of two chromosomes (PC_1 and PC_2), $e \in 1, 2, ..., k_1$ clusters are copied from PC_1 to PC_2, assuming that PC_1 represents k_1 clusters. The unchanged clusters of PC_2 are maintained. The modified clusters have their remaining instances allocated to the cluster with the nearest centroid, resulting in the offspring CC_1. The procedure is repeated to generate another offspring, CC_2. However, now, the modified clusters of PC_2 are copied into PC_1. Figure 3 illustrates this crossover, where clusters 2 and 3 are copied from PC_1 to PC_2. The instances of clusters indirectly affected have their value set to zero and are reallocated to the cluster with the nearest centroid later.

The crossover operators used by the EvoCluster algorithm (Ma *et al.*, 2006) can be seen as modified versions of the context-sensitive evolutionary operators of the clustering genetic algorithm (CGA) proposed by Hruschka and Ebecken (Hruschka and Ebecken, 2003). Indeed, in both CGA and EvoCluster, the crossover operators

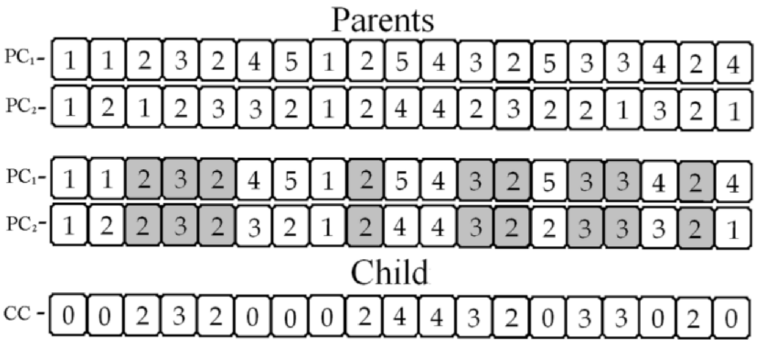

Parents
PC₁- 1 1 2 3 2 4 5 1 2 5 4 3 2 5 3 3 4 2 4
PC₂- 1 2 1 2 3 3 2 1 2 4 4 2 3 2 2 1 3 2 1
PC₁- 1 1 2 3 2 4 5 1 2 5 4 3 2 5 3 3 4 2 4
PC₂- 1 2 2 3 2 3 2 1 2 4 4 3 2 2 3 3 3 2 1
Child
CC - 0 0 2 3 2 0 0 0 2 4 4 3 2 0 3 3 0 2 0

Fig. 9. Example of crossover between the chromosomes PC_1 and PC_2.

were essentially designed to copy (exchange), split, merge, and eliminate groups. Besides some minor details on how these basic operations are performed, there is an important difference regarding the way the operators are applied in each algorithm: the application of EvoCluster's crossover operator can be probabilistically guided by information concerning the quality of the individual clusters in a given partition.

Estivill-Castro et al. (Casillas *et al.*, 2003) use a one-point crossover that is also context sensitive. This operator manipulates the edges of a Minimum Spanning Tree (MST) in which the nodes represent the dataset objects and the edges correspond to proximity indexes between them.

An interesting work, Pasi Fränti et al. (Fränti *et al.*, 1997) use five crossover operators that fundamentally select k centroids from two parents. The random crossover operator randomly chooses $k/2$ centroids from each of the two parents. Duplicate centroids are replaced by averages of repeated draws. In the operator named centroid distance, the clusters are initially sorted according to their distances from the grand mean of the dataset. Next, they are divided into two subsets, namely: central clusters and remote clusters. The central clusters are those closer to the centroid of the dataset, whereas the remote clusters are the remaining ones. An offspring is created by taking the central clusters from parent PC_1 and the remote clusters from parent PC_2. In pairwise crossover, clusters codified in different parents are paired according to the similarities of their centroids. Offspring is then generated by randomly taking one centroid from each pair of clusters. In the largest partition operator, y centroids are selected by a greedy heuristic based on the assumption that larger clusters are more important than smaller ones. Finally, they evaluate the pairwise nearest neighbor crossover operator that considers that the $2k$ centroids from parents PC_1 and PC_2 can be clustered into k clusters that will form the offspring. All the crossover operators previously described by the authors are context-sensitive, except for centroid distance, which can be viewed as a variant of the single point crossover. They argue that the pairwise nearest neighbor operator is the best choice. Another work (Kivijärvi *et al.*, 2003) used the same crossover operators above in addition to a single point crossover operator.

5.3 Mutation

The best known mechanism for producing new variations on the population is mutation. This operator randomly selects genes and modifies their values. In clustering, the mutation usually works by moving objects between clusters (Murthy and Chowdhury, 1996). In Tseng et al. (Tseng and Yang, 2001), bits of the strings representing the individuals are selected according to a given probability, and then changed either from 0 to 1 or from 1 to 0. Conceptually speaking, generated clusters can be either inserted into a given chromosome or eliminated from it.

One of the most used mutation operators consists of randomly selecting a centroid/medoid to be replaced by an object from the dataset - according to a predetermined probability (Lucasius *et al.*, 1993, Estivill-Castro, 1997, Sheng and Liu, 2004, Fränti *et al.*, 1997, Kivijärvi *et al.*, 2003).

Some implementations alter the partition with the insertion/creation and removal/agglomeration of clusters (Hall *et al.*, 1999, Greene, 2003). An example are the two mutation operators used in (Hruschka and Ebecken, 2003, Hruschka *et al.*, 2004). The first operator works only on genotypes that encode more than two clusters. It eliminates a randomly chosen cluster, placing its objects into the nearest remaining clusters (according to their centroids). The second mutation operator splits a randomly selected cluster, which must be formed by at least two objects to be eligible for this operator, into two new ones. The first cluster is formed by the objects closer to the original centroid, whereas the other cluster is formed by those objects closer to the farthest object from the centroid.

Another class of operators is based on the displacement of the clusters centroids on the vectorial space. An operator of this class, proposed by Scheunders (Scheunders, 1997), randomly adds either a negative or a positive constant (-1 or +1) to a randomly chosen component of the centroid of a given cluster. The mutation of clusters centers by the a similar procedure is investigated in (Maulik and Bandyopadhyay, 2000). A number δ in the range $[0, 1]$ is generated with uniform distribution. This number is then used to change the value v of a given gene to $(1 \pm 2\delta)v$, when $v \neq 0$, or to $\pm 2\delta$ when $v = 0$. The signs "+" and "$-$" occur with equal probability.

A number of GAs combine two or more mutation operators to increase the population diversity. An example is the GA described in (Merz and Zell, 2002), which uses two distinct mutation operators. The first operator assigns a randomly chosen dataset object to substitute a randomly chosen centroid. The second operator randomly selects two clusters C_i and C_j. Then, the object belonging to C_i with the maximum distance from its centroid is chosen to replace the centroid of C_j.

Another example is the algorithm EvoCluster (Ma *et al.*, 2006) which has six mutation operators. Similarly to the operators used in (Hruschka and Ebecken, 2003), these operators essentially split, merge, and eliminate groups. However, differently from the previous operators, EvoCluster's mutation operators can be simultaneously applied to multiple clusters of the same partition. Besides, they can be probabilistically guided by information concerning the quality of the individual clusters in a given partition.

6 Some Related Works and Applications

GAs have been successfully used as clustering techniques in many applications, such as image processing (Hall *et al.*, 1999, Kivijärvi *et al.*, 2003), classification of pixels of satellite image (Bandyopadhyay and Maulik, 2002), gene expression analysis (Pan *et al.*, 2003), crude oil analysis (Murthy and Chowdhury, 1996, Maulik and Bandyopadhyay, 2000) and intrusion detection in computer networks (Liu *et al.*, 2004), just to mention a few.

Several works combine the use of GAs with other clustering techniques, e. g. by employing a GA to select characteristics of the database to be clusted (Ohtsuka *et al.*, 2002) or initial clusters for these techniques (Maulik and Bandyopadhyay, 2000). A method that combines a Bayesian feature selection approach with a clustering genetic algorithm to get classification rules is described in (Hruschka and Ebecken, 2003). In (Hruschka and Ebecken, 2006), a clustering genetic algorithm is applied to extract rules from multilayer perceptrons trained in classification problems.

7 Conclusion

This text presents a survey on the use of Genetic Algorithms for clustering. GAs can represent different types of clustering partitions while being able to adopt a wide variety of fitness functions as objective functions based on clustering validity criteria. The combination of several desirable optimization skills makes GAs able to be applied to many clustering problems from a large number of application areas.

Most of the objective functions involved in clustering and measures employed for clustering validation are based on different inductive biases, which can favor datasets with particular characteristics. Thus, it is worth stressing that the selection of a proper fitness function to the problem in hand is important to obtain the desired results.

Current research has focused on more efficient cluster representations and context sensitive genetic operators. The study of fitness and objective functions is also an important research area on GAs applied to clustering problems.

Acknowledgements

The authors would like to thank CNPq, FAPESP and FINEP for the support received during this work.

References

Baker, J.E., (1987), Reducing bias and inefficiency in the selection algorithm. Proceedings of the Second International Conference on Genetic Algorithms and their Application, pp. 14–21.

Bandyopadhyay, S., Maulik, U., (2001), Nonparametric genetic clustering: Comparison of validity indices. Systems, Man and Cybernetics, Part C, IEEE Transactions on : Applications and Reviews. **31**(1): 120–125.

Bandyopadhyay, S., Maulik, U., (2002), An evolutionary technique based on k-means algorithm for optimal clustering in rn. Inf. Sci. Appl. **146**(1-4): 221–237.

Belew, R.K., Booker, L.B., (1991), eds., Solving Partitioning Problems with Genetic Algorithms. In Belew, R.K., Booker, L.B., eds.: ICGA, Morgan Kaufmann.

Bezdek, J.C., Boggavaparu, S., Hall, L.O., (1994), Bensaid, A., Genetic algorithm guided clustering. Procedings of the First IEEE Conference on Evolutionary Computation: 34–40.

Calinski, T., Harabasz, J., (1974), A dendrite method for cluster analysis. Communications in statistics **3**(1): 1–27.

Casillas, A., de Lena, M.T.G., Martnez, R., (2003), Document clustering into an unknown number of clusters using a genetic algorithm. Lecture Notes in Computer Science **2807**: 43–49.

Cole, R.M., (1998), Clustering with Genetic Algorithms. PhD thesis, Department of Computer Science, University of Western Australia.

Cowgill, M.C., Harvey, R.J., Watson, L.T., (1998), A genetic algorithm approach to cluster analysis. Technical report, Virginia Polytechnic Institute & State University, Blacksburg, VA, USA.

Darwin, C., (2006), The Origin of Species: A Variorum Text. University of Pennsylvania Press.

Davies, D., Bouldin, D.W., (1979), A cluster separation measure. IEEE Transactions of Pattern Analysis and Machine Intelligence **1**: 224–227.

Duda, R., Hart, P., Stork, D., (2001), Pattern Classification. John Wiley & Sons.

Dunn, J., (1973), A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. J. Cybern **3**: 32–57.

Estivill-Castro, V., (1997), Spatial clustering for data mining with genetic algorithms. Technical report, Australia.

Everitt, B., Landau, S., Leese, M., (2001), Cluster Analysis, Arnold Publishers. Arnold Publishers.

Fränti, P., Kivijärvi, J., Kaukoranta, T., Nevalainen, O., (1997), Genetic algorithms for large scale clustering problems. The Computer Journal **40**: 547–554.

Freitas, A. (2005), Evolutionary Algorithms for Data Mining. in Oded Maimon, Lior Rokach (Eds.), The Data Mining and Knowledge Discovery Handbook, Springer, pp. 435–467.

Goldberg, D., (1989), Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley.

Greene, W.A., (2003), Unsupervised hierarchical clustering via a genetic algorithm. In: Proceedings of the 2003 Congress on Evolutionary Computation, IEEE Press, pp. 998–1005.

Grefenstette, J., (2000), Proportional selection and sampling algorithms. In: Evolutionary Computation 1. Institute of physics publishing, pp. 172–180.

Halkidi, M., Batistakis, Y., Vazirgiannis, M., (2001), On clustering validation techniques. Intelligent Information Systems Journal **17**(2-3): 107–145.

Hall, L., Ozyurt, B., Bezdek, J., (1999), Clustering with a genetically optimized approach. IEEE Transations on Evolutionary Computation. 3: 103–112.

Holland, J., (1975), Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor.

Hruschka, E.R., Campello, R.J.G.B., de Castro, L.N., (2004), Improving the efficiency of a clustering genetic algorithm. In: Advances in Artificial Intelligence

- IBERAMIA 2004: 9th Ibero-American Conference on AI, Puebla, Mexico, November 22-25. Proceedings. Volume 3315., Springer-Verlag GmbH, Lecture Notes in Computer Science, pp.861-868.

Hruschka, E.R., Ebecken, N.F.F., (2003), A genetic algorithm for cluster analysis. Intelligent Data Analysis **7**(1): 15–25.

Hruschka, E.R., Ebecken, N.F.F., (2003), A feature selection bayesian approach for extracting classification rules with a clustering genetic algorithm. Applied Artificial Intelligence **17**(5-6): 489–506.

Hruschka, E.R., Ebecken, N.F.F., (2006), Extracting rules from multilayer perceptrons in classification problems: A clustering-based approach. Neurocomputing **70**: 384–397.

Jain, A.K., Murty, M.N., Flynn, P.J., (1999), Data clustering: a review. ACM Computing Surveys **31**(3): 264–323.

Jain, A., Dubes, R., (1988), Algorithms for Clustering Data. Prentice Hall.

Kaufman, L., Rousseeuw, P., (1990), Finding groups in data: An introduction to cluster analysis. Wiley Series in Probability and Mathematical Statistics.

Kivijärvi, J., Fränti, P., Nevalainen, O., (2003), Self-adaptive genetic algorithm for clustering. Journal of Heuristics **9**(2): 113–129.

Korkmaz, E.E., Du, J., Alhajj, R., Barker, K., (2006), Combining advantages of new chromosome representation scheme and multi-objective genetic algorithms for better clustering. Intell. Data Anal. **10**(2): 163–182.

Krovi, R., (1992), Genetic algorithms for clustering: a preliminary investigation. System Sciences, 1992. Proceedings of the Twenty-Fifth Hawaii International Conference on **4**: 540–544.

Kuncheva, L., Bezdek, J.C., (1997), Selection of cluster prototypes from data by a genetic algorithm. Procedings of the 5th European Congress on Intelligent Techniques and Soft Computing, pp. 1683–1688.

Liu, Y., Chen, K., Liao, X., Zhang, W., (2004), A genetic clustering method for intrusion detection. Pattern Recognition **37**(5): 927–942.

Lucasius, C.B., Dane, A.D., Kateman, G., (1993), On k-medoid clustering of large data sets with the aid of a genetic algorithm: background, feasibility and comparison. Analytica Chimica Acta, pp. 647–669.

Ma, P.C.H., Chan, K.C.C., Yao, X., Chiu, D.K.Y., (2006), An evolutionary clustering algorithm for gene expression microarray data analysis. IEEE Trans. Evolutionary Computations **10**(3): 296–314.

Maulik, U., Bandyopadhyay, S., (2000), Genetic algorithm-based clustering technique. Pattern Recognition **33**: 1455 – 1465.

Merz, P., Zell, A., (2002), Clustering gene expression profiles with memetic algorithms. In: PPSN VII: Proceedings of the 7th International Conference on Parallel Problem Solving from Nature, London, UK, Springer-Verlag, pp. 811–820.

Milligan, G.W., Cooper, M.C., (1985), An examination of procedures for determining the number of clusters in a data set. Psychometrika **50**: 159–179.

Mitchell, M., (1999), An introduction to Genetic Algorithms. MIT Press.

Murthy, C.A., Chowdhury, N., (1996), In search of optimal clusters using genetic algorithms. Pattern Recogn. Lett. **17**(8): 825 – 832.

Ohtsuka, A., Kamiura, N., Isokawa, T., Matsui, N., (2002), On detection of confused blood samples using self organizing maps and genetic algorithm. In: Neural Information Processing, 2002. ICONIP '02. Proceedings of the 9th International Conference on. Volume 5., Department of Computer Science and Illinois Genetic Algorithms Laboratory, 2233 – 2238.

Pal, N., Bezdek, J., (1995), On cluster validity for the fuzzy c-means model. IEEE Transactions of Fuzzy Systems **3**(3):370–379.

Pan, H., Zhu, J., Han, D., (2003), Genetic algorithms applied to multi-class clustering for gene expression data. Genomics, Proteomics and Bioinformatics **1**(4): 279–287.

Rousseeuw, P.J., (1987), Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics **20**:53–65.

Scheunders, P., (1997), A genetic c-means clustering algorithm applied to color image quantization. Pattern Recognition **30**(6): 859–866.

Sheng, W., Liu, X., (2004), A hybrid algorithm for k-medoid clustering of large data sets. In: Proceedings of the 2004 IEEE Congress on Evolutionary Computation, Portland, Oregon, IEEE Press, pp. 77–82.

Tseng, L., Yang, S.B., (2001), A genetic approach to the automatic clustering problem. Pattern Recognition **34**:415–424.

Discovering New Rule Induction Algorithms with Grammar-based Genetic Programming

Gisele L. Pappa and Alex A. Freitas

Computing Laboratory
University of Kent
Canterbury, Kent, CT2 7NF, UK
`glp6, A.A.Freitas@kent.ac.uk`
`http://www.cs.kent.ac.uk/∼aaf`

Summary. Rule induction is a data mining technique used to extract classification rules of the form IF (conditions) THEN (predicted class) from data. The majority of the rule induction algorithms found in the literature follow the sequential covering strategy, which essentially induces one rule at a time until (almost) all the training data is covered by the induced rule set. This strategy describes a basic algorithm composed by several key elements, which can be modified and/or extended to generate new and better rule induction algorithms. With this in mind, this work proposes the use of a grammar-based genetic programming (GGP) algorithm to automatically discover new sequential covering algorithms. The proposed system is evaluated using 20 data sets, and the automatically-discovered rule induction algorithms are compared with four well-known human-designed rule induction algorithms. Results showed that the GGP system is a promising approach to effectively discover new sequential covering algorithms.

1 Introduction

In the classification task of data mining, one way of representing the knowledge discovered by the data mining algorithm consists of a set of classification rules, where each rule has the form: IF (conditions) THEN (predicted class). This knowledge representation has the advantage of being intuitively comprehensible to the user (Iglesia *et al.*, 1996).

There are a number of rule induction algorithms that have been proposed to discover such classification rules (Fürnkranz, 1999, Mitchell, 1997). A particularly popular strategy consists of the sequential covering approach, where in essence the algorithm discovers one rule at a time until (almost) all examples are covered by the discovered rules (i.e., match the conditions of at least one rule). Sequential covering rule induction algorithms are typically greedy, performing a local search in the rule space.

An alternative approach to discover classification rules consists of using an evolutionary algorithm (EA), which performs a more global search in the rule space.

Indeed, there are also a number of EAs for discovering a set of classification rules from a given data set (Freitas, 2002).

In this chapter, however, we propose a very different and pioneering way of using an EA in the context of classification rule discovery. We propose a grammar-based genetic programming (GGP) system that *automatically* discovers new sequential covering rule induction *algorithms* (rather than *rules*). The discovered rule induction algorithms are generic and robust enough to be applicable to virtually any classification data set in any application domain, in the same way that a manually-designed rule induction algorithm is generic and robust enough to be applicable to virtually any classification data set.

The proposed method allows the automatic discovery of new rule induction algorithms potentially quite different from conventional, manually-designed rule induction algorithms. Hence, the automatically-discovered rule induction algorithms can avoid some of the human preconceptions and biases embedded in manually-designed rule induction algorithms, possibly leading to more effective algorithms in challenging application domains.

The first version of the proposed GGP system and its corresponding computational results have been previously published in (Pappa and Freitas, 2006). Nonetheless, one limitation of that version is that the evolved rule induction algorithms could cope only with nominal (categorical) attributes, and not with continuous (real-valued) attributes. In this chapter we describe a new, extended version of the system, which can cope with both nominal and continuous attributes. This new characteristic introduced into the system makes it now suitable for a larger variety of classification data sets, and this fact is reflected in the greater number of data sets used to evaluate the system, namely 20 data sets, whereas our first experiments reported in (Pappa and Freitas, 2006) used only 11 data sets. In addition, this chapter shows and discusses in detail one of the rule induction algorithms automatically generated by the GGP system, a result that was not reported in (Pappa and Freitas, 2006).

The remainder of this chapter is organized as follows. Section 2 briefly discusses rule induction algorithms. Section 3 gives a brief overview of GGP. Section 4 introduces the proposed GGP. Section 5 reports the results of several computational experiments. Finally, Section 6 presents the conclusions and describes future research directions.

2 Sequential Covering Rule Induction Algorithms

The sequential covering strategy (also known as separate and conquer) is certainly the most explored and most used strategy to induce rules from data. It was first employed by the algorithms of the AQ family (Michalski, 1969) in the late sixties, and over the years was applied again and again as the basic algorithm in rule induction systems.

The separate and conquer strategy works as follows. It learns a rule from a training set, removes from it the examples covered by the rule, and recursively learns another rule which covers the remaining examples. A rule is said to cover an example e when all the conditions in the antecedent of the rule are satisfied by the example e. For instance, the rule "IF (salary $>$ £ 100,000) THEN rich" covers all the examples in the training set in which the value of salary is greater than £100,000, regardless of the current value of the class attribute of an example. The learning process goes on until a pre-defined criterion is satisfied. This criterion usually requires that all or almost all examples in the training set are covered by a rule.

Algorithms following the sequential covering approach usually differ from each other in four main elements: the representation of the candidate rules, the search mechanisms used to explore the space of candidate rules, the way the candidate rules are evaluated and the pruning method, although the last one can be absent (Fürnkranz, 1999, Witten and Frank, 2005).

Considering the first of these four rule induction algorithms elements, the rule representation has a significant influence in the learning process, since some concepts can be easily expressed in one representation but hardly expressed in others. The most common rule representations used by rule induction algorithms are propositional or first order logic.

The next two elements found in rule induction algorithms determine how the algorithm will explore the space of candidate rules. The first of them, i.e., the search mechanism, is composed by two components: a search strategy and a search method. The search strategy determines the region of the search space where the search starts and its direction, while the search method specifies which specializations/generalizations should be considered.

Broadly speaking, there are three kinds of search strategies: bottom-up, top-down and bi-directional. A bottom-up strategy starts the search with a very specific rule, and iteratively generalizes it. A top-down strategy, in contrast, starts the search with the most general rule and iteratively specializes it. The most general rule is the one that covers all examples in the training set (because it has an empty antecedent, which is always satisfied for any example). At last, a bi-directional search is allowed to generalize or specialize the candidate rules.

Any of these search strategies is complemented with a search method. The search method is a very important part of a rule induction algorithm since it determines which specializations/generalizations will be considered at each specialization/generalizations step. Too many specializations/generalizations are not allowed due to computational time, but too few may disregard good conditions and reduce the chances of finding a good rule. Among the available search methods are the greedy search and the beam search.

The search method is guided by a rule evaluation heuristic, which is the second component found in rule induction algorithms which has influence in the way the rule space is explored. The regions of the search space being explored by a rule induction algorithm can drastically change according to the heuristic chosen to assess a rule while it is being built. Among the heuristics used to estimate the quality of a rule are the information content, information gain, Laplace correction, m-estimate, ls-content, among others (Fürnkranz, 1999).

The first algorithms developed using the sequential covering approach were composed by the three components described so far: a rule representation, a search strategy and a evaluation heuristic. However, these first algorithms searched the data for complete and consistent rule sets. It means they were looking for rules that covered all the examples in the training set (complete) and that covered no negative examples (consistent). These are not common characteristics of any real-world data sets. Hence, pruning methods were introduced to sequential covering algorithms to avoid over-fitting and to handle noisy data.

Pruning methods are divided in two categories: pre- and post-pruning. Pre-pruning methods stop the refinement of the rules before they become too specific or over-fit the data, while post-pruning methods first produce a complete and consistent rule or rule set, and later try to simplify it. When comparing pre- and post-pruning techniques, each of them has its advantages and pitfalls. Though pre-pruning techniques are faster, post-pruning techniques usually produce simpler and more accurate models (at the expense of inefficiency, since some rules are learned and then simply discarded from the model).

In an attempt to solve the problems caused by pre- and post-pruning techniques, some methods combine or integrate them to get the best of both worlds. Some of these systems, for instance, prune each rule right after it is created, instead of waiting for the complete model to be generated (Cohen, 1995).

This section briefly described the main elements which compose a sequential covering rule induction algorithm. Knowledge about these elements and the variety of ways they can be implemented was the base to build the grammar used by the GGP system described in Section 4. For a more complete survey of sequential covering rule induction algorithms and its components the user is referred to (Fürnkranz, 1999, Mitchell, 1997, Witten and Frank, 2005).

3 Grammar-based Genetic Programming

Genetic Programming (GP) (Koza, 1992) is an area of evolutionary computation which aims to automatically evolve computer programs. It works by following Darwin's principle of selection and survival of the fittest, and can be easily adapted to solve a variety of problems. GP's success is backed up by a list of 36 human-competitive solutions, where two created patentable new inventions (gp.org).

Grammar-based Genetic Programming (GGP) is a variation of the classical GP method and, as its name indicates, the main difference among GGP and GP is that the former uses a grammar to create the population of candidate solutions for the targeted problem. The main advantage of using a grammar together with a GP system is that it can include previous knowledge about how the target problem is solved, and so be used to guide the GP search. Moreover, GGP solves a well-known problem in the GP literature, called closure[1].

Grammars (Aho *et al.*, 1986) are simple mechanisms capable of representing very complex structures. Their formal definition was first given by Chomsky in 1950. According to Chomsky, a grammar can be represented by a four-tuple $\{N, T, P, S\}$, where N is a set of non-terminals, T is a set of terminals, P is a set of production rules, and S (a member of N) is the start symbol. The production rules define the language which the grammar represents by combining the grammar symbols.

In this work we are specially interested in context-free grammars (CFG). CFGs are the class of grammars most commonly used with genetic programming, and they are usually described using the Backus Naur Form (BNF) (Naur, 1963).

[1] A traditional GP system creates individuals by combining a set of functions and terminals. The closure property states that every function in the GP function set has to be able to process any value it receives as input. For further details see (Banzhaf *et al.*, 1998).

According to the BNF notation, production rules have the form <expr> ::= <expr><op><expr>, and symbols wrapped in "<>" represent the non-terminals of the grammar. Three special symbols might be used for writing the production rules in BNF: "|","[ ]" and "( )". "|" represents a choice, like in <var> ::=$x|y$, where <*var*> generates the symbol x or y. "[ ]" wraps an optional symbol which may or may not be generated when applying the rule. "( )"is used to group a set of choices together, like in $x ::= k(y|z)$, where x generates k followed by y or z. The application of a production rule from $p \in P$ to some non-terminal $n \in N$ is called a derivation step, and it is represented by the symbol $\Rightarrow$.

Once a grammar that includes background knowledge about the target problem has been defined by the user, a GGP system follows the pseudo-code defined in Algorithm 1.

Algorithm 1: Basic pseudo-code for a GGP system

```
Define a representation for the individuals
Define parameters such as population size, number of generations, crossover,
mutation and reproduction rates
Generate the first GGP population based on the production rules of the
grammar
while maximum number of generations not reached do
    Evaluate the individuals using a pre-defined fitness function
    Select the best individuals, according to the fitness function, to breed
    Perform crossover, mutation and reproduction operations with the
    selected individuals, always producing offspring which are also valid
    according to the defined grammar
Return individual with best fitness
```

Note that each individual represents a candidate solution to the target problem. Hence, since we use a GGP to automatically evolve rule induction algorithms, each individual is a rule induction algorithm, and the grammar gathers knowledge about how rule induction algorithms were previously developed.

4 Discovering New Rule Induction Algorithms

This section describes an extended version of the GGP method proposed in (Pappa and Freitas, 2006). As explained before, the main difference between the method described here and the one described in (Pappa and Freitas, 2006) is that, while the latter could only cope with nominal attributes, the former can cope with both nominal and numerical attributes.

In summary, the proposed GGP works as follows. It creates the first population of individuals based on the production rules of a grammar, which is used to guide the search for new rule induction algorithms. In this population, each GGP individual represents a complete sequential-covering rule induction algorithm, such as CN2 (Clark and Boswell, 1991). As the GGP system is based on the principle of survival of the fittest, a fitness function is associated with each individual in the population, and used to select a subset of them (through a tournament selection of size 2) to breed and undergo crossover, mutation and reproduction operations. The individuals generated by these operations (which also have to be valid according to the grammar being used) are inserted into a new population, representing a new generation of evolved individuals. The evolution process is carried out until a maximum number of generations is reached.

Note that the main modifications introduced to the system in order to cope with numerical attributes are related to the terminals of the grammar and the way they are implemented. Hence, in this section, we first present the grammar introduced in (Pappa and Freitas, 2006), emphasizing its components which cannot be found in traditional rule induction algorithms, and then we present the modifications necessary to make it cope with numerical attributes. Following the grammar description, we show an example of an individual which can be evolved by the system, and then describe the individuals' evaluation process. Finally, we explain how the evolutionary operators were implemented.

4.1 The grammar

In a GGP system, the grammar is the element which determines the search space of the candidate solutions for a target problem. Hence, in the GGP system proposed here, the grammar contains previous knowledge about how humans design rule induction algorithms, plus some new concepts which were borrowed from other data mining paradigms or created by the authors (and that to the best of the authors' knowledge were never used in the design of sequential-covering rule induction algorithms).

The proposed grammar is presented in Table 1. It uses the BNF terminology introduced earlier, and its *Start* symbol is represented by the non-terminal with the same name. Recall that non-terminals are wrapped into <> symbols, and each of them originates a production rule. Grammar symbols not presented between <> are terminals. In the context of rule induction algorithms, the set of non-terminals and terminals are divided into two subsets. The first subset includes general programming elements, like *if* statements and *for/while* loops, while the second subset includes components directly related to rule induction algorithms, such as *RefineRule* or *PruneRule*.

The non-terminals in the grammar represent high-level operations, like a while loop (*whileLoop*) or the procedure performed to refine a rule (*RefineRule*). The terminals, in turn, represent a very specific operation, like *Add1*, which adds one condition-at-a-time to a candidate rule during the rule refinement process (*RefineRule*). Terminals are always associated with a building block. A building block represents an "atomic operation" (from the grammar's viewpoint) which does not need any more refinements. Building blocks will be very useful during the phase of rule induction code generation, as each of them is associated with a chunk of Java code.

As observed in Table 1, the grammar contains 26 non-terminals (NT), where each NT can generate one or more production rules. Recall that in the BNF notation, used to describe the grammar in Table 1, the symbol "|" separates different production rules, and the symbol "[ ]" wraps an optional symbol (which may or may not be generated when applying the rule). For

Table 1. The grammar used by the GGP (adapted from (Pappa and Freitas, 2006))

```
1-  <Start> ::= (<CreateRuleSet>|<CreateRuleList>) [<PostProcess>].
2-  <CreateRuleSet> ::= forEachClass <whileLoop> endFor
                        <RuleSetTest>.
3-  <CreateRuleList> ::= <whileLoop> <RuleListTest>.
4-  <whileLoop>::= while <condWhile> <CreateOneRule> endWhile.
5-  <condWhile>::= uncoveredNotEmpty |uncoveredGreater
                  (10| 20| 90%| 95%| 97%| 99%) trainEx.
6-  <RuleSetTest> ::= lsContent |confidenceLaplace.
7-  <RuleListTest>::= appendRule | prependRule.
8-  <CreateOneRule>::= <InitializeRule> <innerWhile> [<PrePruneRule>]
                       [<RuleStoppingCriterion>].
9-  <InitializeRule> ::= emptyRule| randomExample| typicalExample |
                         <MakeFirstRule>.
10- <MakeFirstRule> ::= NumCond1| NumCond2| NumCond3| NumCond4.
11- <innerWhile> ::= while (candNotEmpty| negNotCovered)
                     <FindRule> endWhile.
12- <FindRule> ::= (<RefineRule>|<innerIf>) <EvaluateRule>
                  [<StoppingCriterion>] <SelectCandidateRules>.
13- <innerIf> ::= if <condIf> then <RefineRule> else <RefineRule>.
14- <condIf> ::=  <condIfExamples> | <condIfRule>.
15- <condIfRule> ::= ruleSizeSmaller (2| 3| 5| 7).
16- <condIfExamples> ::= numCovExp ( >| <)(90%| 95%| 99%).
17- <RefineRule> ::= <AddCond>| <RemoveCond>.
18- <AddCond> ::= Add1| Add2.
19- <RemoveCond>::= Remove1| Remove2.
20- <EvaluateRule>::= confidence | Laplace| infoContent| infoGain.
21- <StoppingCriterion> ::= MinAccuracy (0.6| 0.7| 0.8)|
                            SignificanceTest (0.1| 0.05| 0.025| 0.01).
22- <SelectCandidateRules> ::= 1CR| 2CR| 3CR| 4CR| 5CR| 8CR| 10CR.
23- <PrePruneRule> ::= (1Cond| LastCond| FinalSeqCond) <EvaluateRule>.
24- <RuleStoppingCriterion> ::= accuracyStop (0.5| 0.6| 0.7).
25- <PostProcess> ::= RemoveRule EvaluateModel| <RemoveCondRule>.
26- <RemoveCondRule> ::= (1Cond| 2Cond| FinalSeq) <EvaluateRule>.
```

instance, the NT *Start* generates four production rules: *CreateRuleList*, *CreateRuleSet*, *CreateRuleList PostProcess* and *CreateRuleSet PostProcess*. In total, the grammar has 83 production rules, which were carefully generated after a comprehensive study of the main elements of the pseudo-codes of basic rule induction algorithms, which follow the basic process described in Section 2.

In this section, we focus on the new components of the grammar, which are usually not found in traditional rule induction algorithms. The major "new" components inserted to the grammar are:

- The terminal *typicalExample*, which creates a new rule using the concept of typicality, borrowed from the instance-based learning literature (Zhang, 1992). An example is said to be typical if it is very similar to the other examples belonging to the same class it belongs to, and not similar to the other examples belonging to other classes. In other words, a typical example has high intra-class similarity and low inter-class similarity.
- The non-terminal *MakeFirstRule*, which allows the first rule to be initialized with one, two, three or four attribute-value pairs, selected probabilistically from the training data in proportion to their frequency. Attribute-value pairs are selected subject to the restriction that they involve different attributes (to prevent inconsistent rules such as "sex = male AND sex = female").
- The non-terminal *innerIf*, which allows rules to be refined in different ways (e.g. adding or removing one or two conditions-at-a-time to/from the rule) according to the number of conditions they have, or the number of examples the rule list/set covers.
- Although some methods do use rule look-ahead, i.e., they do insert more than one condition-at-a-time to a set of candidate rules, we did not find in the literature any rule induction algorithm which removes two conditions-at-a-time from a rule. This is implemented by the terminal *Remove2*.

Note that the list above shows a set of single components which are new "building blocks" of rule induction algorithms. These components increase the diversity of the candidate rule induction algorithms considerably, but it is the combination of the "standard" and new components which will potentially contribute to the creation of a new rule induction algorithm different from conventional algorithms.

As it will be discussed in Section 4.3, the individuals generated by following the production rules of the grammar are converted into executable rule induction algorithms by using a GGP/Java interface, which reads out an individual and puts together chunks of code associated with the terminals and non-terminals of the grammar contained in that individual.

Hence, in order to modify the grammar and make it cope with data sets containing numerical attributes, the main modifications are introduced in some chunks of Java code associated with the terminals of the grammar. The terminals whose implementation went through major extensions were the ones responsible for refining rules by adding/removing conditions to/from it. They were extended in a way that they can produce algorithms that represent rule conditions of the form "<attribute, operator, value>", where operator is "=" in the case of nominal attributes, and operator is "$\geq$" or "$\leq$" in the case of numerical attributes.

The approach followed by these terminals to generate rule conditions with numerical attributes is similar to the one implemented by the Ripper and C4.5 algorithms, where the values of a numerical attribute are sorted, and all threshold values considered. The best threshold value is chosen according to

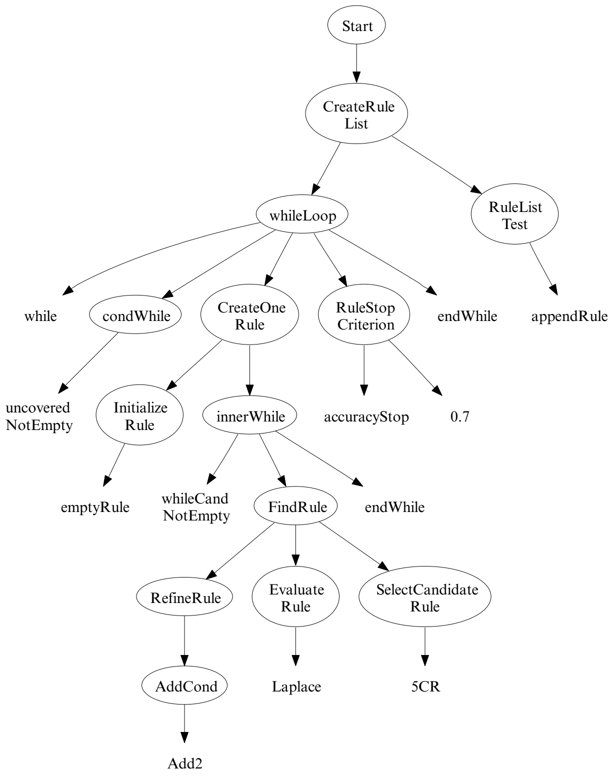


Fig. 1. Example of a GGP Individual (a complete rule induction algorithm)

the information gain associated with that attribute-value pair - see (Witten and Frank, 2005) or (Quinlan, 1993) for details.

By applying the production rules defined by the grammar, the GGP system can generate up to approximately 5 billion different rule induction algorithms (Pappa, 2007). Each of these rule induction algorithms can be represented by an individual in the GGP population. The next section explains how the individuals extracted from the grammar are represented in the proposed GGP system.

4.2 Individual Representation

In a GGP system, each individual represents a candidate solution for the problem being tackled. In this work, each individual represents a complete rule induction algorithm following the sequential covering approach, which can be applied to generate rules for any classification data set.

Figure 1 shows an example of an individual generated by the grammar presented in the previous section. The root of the tree is the non-terminal *Start*. The tree is then derived by the application of production rules for each non-terminal. For example, *Start* (NT 1) generates the non-terminal *CreateRuleList* (NT 3), which in turn produces the non-terminals *whileLoop* and *RuleListTest*. This process is repeated until all the leaf nodes of the tree are terminals.

In order to extract from the tree the pseudo-code of the corresponding rule induction algorithm, we read all the terminals (leaf-nodes) in the tree from left to right. The tree in Figure 1, for example, represents the pseudo-code described in Alg. 2 (shown at the end of Section 5), expressed at a high level of abstraction.

4.3 Individual's Evaluation

An evolutionary algorithm works by selecting the fittest individuals of a population to reproduce and generate new offspring. Individuals are selected based on how good their corresponding candidate solutions are to solve the target problem. In our case, we need to evaluate how good a rule induction algorithm is.

In the rule induction algorithm literature, comparing different classification algorithms is not a straightforward process. There is a variety of metrics which can be used to estimate how good a classifier is, including classification accuracy, ROC analysis (Fawcett, 2003) and sensitivity/specificity. There are studies comparing these different metrics, and showing advantages and disadvantages in using each of them (Flach, 2003, Caruana and Niculescu-Mizil, 2004). Nevertheless, as pointed out by Caruana and Niculescu-Mizil (Caruana and Niculescu-Mizil, 2004), in supervised learning there is one ideal classification model, and "we do not have performance metrics that will reliably assign best performance to the probabilistic true model given finite validation data".

Classification accuracy is still the most common metric used to compare classifiers, although some authors tried to show the pitfalls of using classification accuracy when evaluating induction algorithms (Provost *et al.*, 1998) – specially because it assumes equal misclassification costs and known class distributions – and others tried to introduce ROC analysis as a more robust standard measure. Based on these facts and on the idea of using a simpler measure when first evaluating the individuals produced by the GGP, we chose to use a measure based on accuracy to compose the fitness of the GGP system.

In this framework, a rule induction algorithm RI_A is said to outperform a rule induction algorithm RI_B if RI_A has better classification accuracy in a set of classification problems. Thus, in order to evaluate the rule induction algorithms being evolved, we selected a set of classification problems, and created a meta-training set. The meta-training set consists of a set of data sets, each of them divided as usual into (non-overlapping) training and validation sets.

As illustrated in Figure 2, each individual in the GGP population is decoded into a rule induction algorithm using a GGP/Java interface. The Java code is then compiled, and the resulting rule induction algorithm run in all the

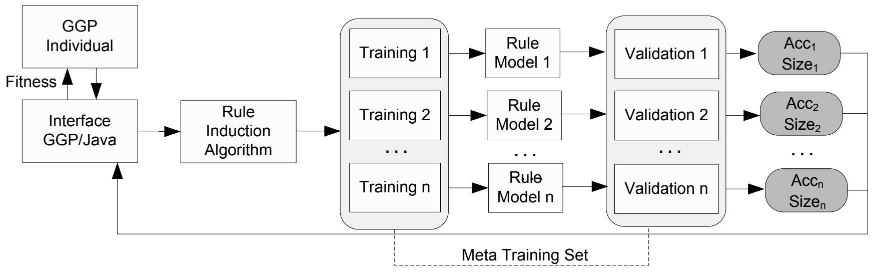


Fig. 2. Fitness evaluation process of a GGP Individual

data sets belonging to the meta-training set. It is a conventional run where, for each data set, a set or list of rules is built using the set of training examples and evaluated using the set of validation examples. It is important to observe that, during preliminary experiments with the GGP, we noticed that it was suffering from a common problem found when solving predictive data mining tasks: over-fitting. As the same training sets were being accessed by the GGP over and over, the produced rule induction algorithms were over-fitting these data sets. We solved this problem with a simple and effective solution borrowed from the literature on GP for data mining (Bhattacharyya, 1998): at each generation, the data used in the training and validation sets of the data sets in the meta-training set are merged and randomly redistributed. This means that, at each generation, the GGP individuals are evaluated in a different set of validation data, helping to avoid over-fitting.

After the rule induction algorithm is run in all the data sets in the meta-training set, the accuracy in the validation set and the rule lists/sets produced for all data sets are returned. These two measures can be used to calculate a fitness function. In this work, we used as the fitness function the average values of the function described in Eq.(1) over all the data sets in the meta-training set.

$$fit_i = \begin{cases} \frac{Acc_i - DefAcc_i}{1 - DefAcc_i}, \text{ if Acc}_i > \text{DefAcc}_i \\ \frac{Acc_i - DefAcc_i}{DefAcc_i}, \text{ otherwise} \end{cases} \tag{1}$$

According to the definition of fit_i, where i denotes the id of a given data set, if the accuracy obtained by the classifier is better than the default accuracy, the improvement over the default accuracy is normalized, by dividing the absolute value of the improvement by the maximum possible improvement. In the case of a drop in the accuracy with respect to the default accuracy, this difference is normalized by dividing the negative value of the difference by the maximum possible drop (the value of *DefAcc$_i$*). The default accuracy for a given data set is simply the accuracy obtained when using the most frequent class in the training set to classify new examples in the validation (or test) set.

The fitness values obtained by the process just described are used for selecting the best individuals in the population, and passing them onto the new generations. At the end of the evolution process, the individual with the best fitness value is returned as the GGP's final solution.

However, in order to verify if the newly created rule induction algorithm is really effective, we have to test it in a new set of data sets, which where unseen during the GGP's evolution. This new set of data sets was named meta-test set, and it is the accuracy obtained by the discovered rule induction algorithms in these data sets which has to be taken into account when evaluating the GGP system.

4.4 Evolutionary Operators

After individuals are generated and evaluated, they are selected to undergo reproduction, crossover and mutation operations, according to used defined probabilities. The reproduction operator simply copies the selected individual to the new population, without any modifications. The crossover operator, in contrast, involves two selected individuals, and swaps a selected subtree between them. The mutation operator also selects a subtree of one selected individual, and replace it by a new, randomly generated tree.

However, in GGP systems, the new individuals produced by the crossover and mutation operators have to be consistent with the grammar. For instance, when performing crossover the system cannot select a subtree with root *EvaluateRule* to be exchanged with a subtree with root *SelectCandidateRules*, because this would create an invalid individual according to the grammar.

Therefore, crossover operations have to exchange subtrees whose roots contain the same non-terminal. Mutation can be applied to a subtree rooted at a non-terminal or applied to a terminal. In the former case, the subtree undergoing mutation is replaced by a new subtree, produced by keeping the same label in the root of the subtree and then generating the rest of the subtree by a new sequence of applications of production rules, so producing a new derivation subtree. When mutating terminals, the terminal undergoing mutation is replaced by another "compatible" symbol, i.e., a terminal or non-terminal which represents a valid application of the production rule whose antecedent is that terminal's parent in the derivation tree. The probability of mutating a non-terminal is 90%, while the probability of mutating a terminal is 10%.

Figure 3 shows an example of a crossover operation. Note that just part of the individuals are shown, for the sake of simplicity. The process works as follows. Parent 1 has a node probabilistically selected for crossover. In the example illustrated, the chosen node is *RefineRule*. The node *RefineRule* is then searched in the derivation tree of parent 2. As parent 2 has a node named *RefineRule*, their subtrees are swapped, generating child 1 and child

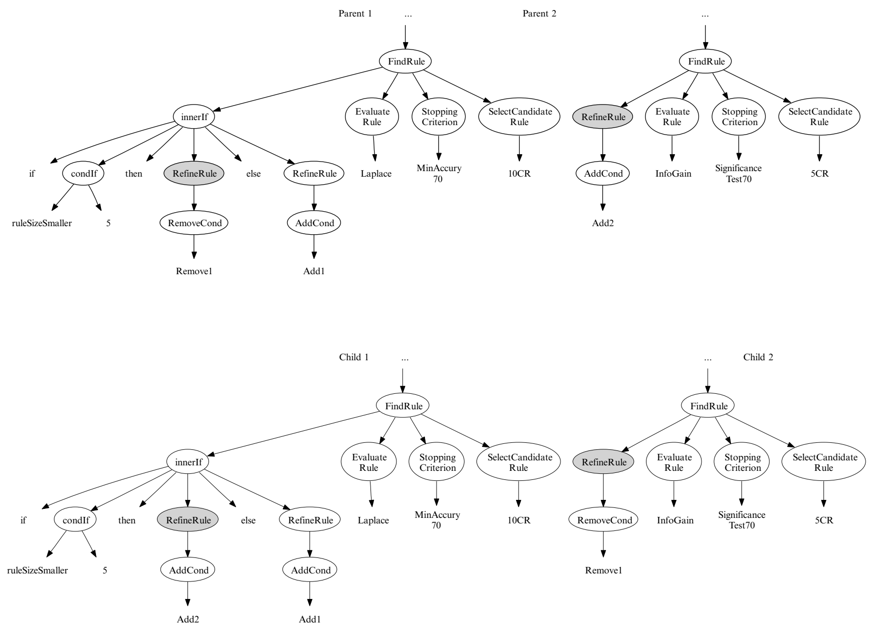


Fig. 3. Example of Crossover in the proposed GGP

2. If *RefineRule* is not present in the tree of parent 2, a new non-terminal is selected from the tree of parent 1. The GGP performs at most 10 attempts to select a node which can be found in both parents. If after 10 attempts it does not happen, both individuals undergo mutation operations.

5 Computational Results and Discussion

In order to test the effectiveness of the proposed GGP system to discover new rule induction algorithms, we have to define two different sets of parameters: (1) the parameters for the GGP system and (2) the data sets used during the training phase of the system.

Table 2 shows the 20 data sets used in the experiments. The figures in the column *Examples* indicate the number of examples present in the training and validation data sets – numbers before and after the "/", respectively, followed by the number of nominal attributes, numerical attributes and classes. The last column shows the default accuracy. It is important to note that during the evolution of the rule induction algorithm by the GGP, for each data set in the meta-training set, each candidate rule induction algorithm (i.e., each GGP individual) is trained with 70% of the examples, and then validated in the

Table 2. Data sets used by the GGP

Data set	Examples	Attributes		Classes	Def. Acc.
		Nomin.	Numer.		(%)
monks-2	169/432	6	-	2	67
monks-3	122/432	6	-	2	52
bal-scale-discr	416/209	4	-	3	46
lymph	98/50	18	-	4	54
zoo	71/28	16	-	7	43
glass	145/69	-	9	7	35.2
pima	513/255	-	8	2	65
hepatitis	104/51	14	6	2	78
vehicle	566/280	-	18	4	26
vowel	660/330	3	10	11	9
crx	461/229	9	6	2	67.7
segment	1540/770	-	19	7	14.3
sonar	139/69	-	60	2	53
heart-c	202/101	7	6	2	54.5
ionosphere	234/117	-	34	2	64
monks-1	124/432	6	-	2	50
mushroom	5416/2708	23	-	2	52
wisconsin	456/227	9	-	2	65
promoters	70/36	58	-	2	50
splice	2553/637	63	-	3	52

remaining 30%. In contrast, in the meta-test set, the evolved rule induction algorithms are evaluated using a well-known 5-fold cross validation procedure (Witten and Frank, 2005).

As our priority was to investigate the influence the GGP parameters have in the quality of the rule induction algorithms produced, we first defined the data sets which will be used in the GGP meta-training and meta-test sets. However, it is not clear how many data sets should be used in each of these meta-sets of data, or what would be the best criteria to distribute them into these two meta-sets. Intuitively, the larger the number of data sets in the meta-training set, the more robust the evolved rule induction algorithm should be. On the other hand, the smaller the number of data sets in the meta-test set, the less information we have about the ability of the evolved rule induction algorithm to obtain a high predictive accuracy for data sets unseen during the evolution of the algorithm.

As a reasonable compromise, the data sets in Table 2 were divided into 2 groups of 10 data sets each. The top 10 sets listed in Table 2 were inserted into the meta-training set, while the bottom 10 data sets formed the meta-test set. We selected the data sets which compose the meta-training set based on the execution time of rule induction algorithms, so that we included in the meta-training set the data sets leading to faster runs of the rule induction algorithms.

After creating the meta-training and meta-test sets, we turned to the GGP parameters: population size, number of generations and crossover, mutation and reproduction rates. In all the experiments reported in this section the population size is set to 100 and the number of generations to 30. These two figures were chosen when evaluating the GGP evolution in preliminary experiments, but are not optimized. Regarding crossover, mutation and reproduction rates, GPs usually use a high rate of crossover and low rates of mutation and reproduction. However, the balance between these three numbers is an open question, and may be very problem dependent (Banzhaf *et al.*, 1998).

In previous experiments, we set the value for the reproduction rate parameter to 0.05, and run the GGP with crossover/mutation rates of 0.5/0.45, 0.6/0.35, 0.7/0.25, 0.8/0.15 and 0.9/0.05, respectively. The results obtained by the GGP when run with these different parameters configurations showed that the system was robust to these variations, producing very similar results with all the configurations. In the experiments reported in this section, the crossover rate was set to 0.7 and the mutation rate to 0.25.

The results obtained by the GGP-derived rule induction algorithms (GGP-RIs) were compared with four well-known rule induction algorithms: the ordered (Clark and Niblett, 1989) and unordered (Clark and Boswell, 1991) versions of CN2, Ripper (Cohen, 1995) and C4.5Rules (Quinlan, 1993). Out of these four algorithms, C4.5Rules is the only one which does not follow the sequential covering approach, which is the approach followed by the GGP-RIs. However, as C4.5Rules has been used as a benchmark algorithm for classification problems for many years, we also included it in our set of baseline comparison algorithms.

It is also important to observe that the current version of the grammar does not include all the components present in Ripper, but does include all the components present in both versions of CN2. In other words, the space of candidate rule induction algorithms searched by the GGP includes CN2, but it does not include C4.5Rules nor the complete version of Ripper.

Table 3 shows the average accuracy obtained by the rule induction algorithms produced by the GGP in 5 different runs, followed by the results of runs of Ordered-CN2, Unordered-CN2, Ripper and C4.5Rules (using default parameter values in all these algorithms). Note that the results reported in Table 3 are the ones obtained in the data sets belonging to the meta-test set (containing data sets unseen during the GGP evolution), and were obtained using a 5-fold cross-validation procedure for each data set. The results obtained by the GGP in the data sets belonging to the meta-training set are not reported here because, as these data sets were seen by the GGP many time during evolution, it is not fair to compare them with any other algorithms. The numbers after the symbol "±" are standard deviations. Results were compared using a statistical t-test with significance level 0.01. Cells in dark gray represent significant wins of the GGP-RIs against the corresponding baseline algorithm, while light gray cells represent significant GGP-RIs' losses.

Table 3. Comparing predictive accuracy rates (%) for the data sets in the meta-test set

Data Set	GGP-RIs	OrdCN2	UnordCN2	Ripper	C45Rules
crx	77.46±3.8	80.16 ± 1.27	80.6 ± 0.93	84.37 ± 1.21	84.82 ± 1.53
segment	95.06±0.26	95.38 ± 0.28	85.26 ± 0.87	95.44 ± 0.32	88.16 ± 7.72
sonar	72.34±1.91	70.42 ± 2.66	72.42 ± 1.4	72.88 ± 4.83	72.4 ± 2.68
heart-c	76.72±1.5	77.9 ± 1.96	77.54 ± 2.85	77.53 ± 1.1	74.2 ± 5.43
ionosphere	87.04±2.2	87.6 ± 2.76	90.52 ± 2.03	89.61 ± 1.75	89.06 ± 2.71
monks-1	99.93±0.07	100 ± 0	100 ± 0	93.84 ± 2.93	100 ± 0
mushroom	99.98±0.02	100 ± 0	100 ± 0	99.96 ± 0.04	98.8 ± 0.06
wisconsin	95.58±0.74	94.58 ± 0.68	94.16 ± 0.93	93.99 ± 0.63	95.9 ± 0.56
promoters	78.98±2.93	81.9 ± 4.65	74.72 ± 4.86	78.18 ± 3.62	83.74 ± 3.46
splice	88.68±0.31	90.32 ± 0.74	74.82 ± 2.94	93.88 ± 0.41	89.66 ± 0.78

In total, Table 3 contains 40 comparative results between GGP-RIs and baseline algorithms – 10 data sets × 4 baseline classification algorithms. Out of theses 40 cases, the accuracy of GGP-RIs was statistically better than the accuracy of a baseline algorithm in three cases, whilst the opposite was true in one case. In the other 36 cases there was no significant difference.

The GGP-RIs' predictive accuracies are statistically better than the C4.5Rules' accuracy in *mushroom* and Unordered-CN2's accuracy in *segment* and *splice*. Naturally, these three cases involve algorithms with the worst accuracy for the respective data set. It is also in a comparison among Ripper and the GGP-RIs in *splice* where the GGP-RIs obtain a significantly worse accuracy than Ripper.

Hence, these experiments lead us to conclude that the GGP-RIs can easily outperform classifiers which are not competitive with the other baseline algorithms. For example, in *splice* the predictive accuracy of Unordered-CN2 is 74.82 ± 2.94, while the other algorithms obtain accuracies close to 90%. In this case, the GGP-RIs can easily find a better accuracy than the one found by Unordered-CN2.

On the other hand, we can say that the GGP was not able to find a rule induction algorithm good enough to outperform the predictive accuracies of Ripper in *splice* because it did not have all the components necessary to do that in its grammar. However, note that the accuracy obtained by Ripper in *splice* is also statistically better than the ones obtained by C4.5Rules and Ordered-CN2 when applying a t-test with 0.01 significance level.

Finally, recall that the search space of the GGP includes both Unordered and Ordered CN2. Hence, it seems fair to expect that the GGP-RIs would never obtain a predictive accuracy significantly worse than either version of CN2. Indeed, this was the case in the experiments reported in Table 3, where the GGP-RIs significantly outperformed Unordered-CN2 in two cases (dark

Algorithm 2: Example of a Decision List Algorithm created by the GGP

```
RuleList = $\emptyset$
repeat
    bestRule = an empty rule
    candidateRules = $\emptyset$
    candidateRules = candidateRules $\cup$ bestRule
    while candidateRules $\neq \emptyset$ do
        newCandidateRules = $\emptyset$
        for each candidateRule CR do
            Add 2 conditions-at-a-time to CR
            Evaluate CR using the Laplace estimation
            newCandidateRules = newCandidateRules $\cup$ CR
        candidateRules = 5 best rules selected from newCandidateRules
        bestRule' = best rule in candidateRules
        if Laplace(bestRule')> Laplace(bestRule) then bestRule = bestRule'
    if accuracy(bestRule) $< 0.7$ then break
    else RuleList = RuleList $\cup$ bestRule
until all examples in the training set are covered
```

gray cells in that table), and there was no case where either version of CN2 significantly outperformed the GGP-RIs.

So far we have shown that the evolved GGP-RIs are competitive to traditional human-designed rule induction algorithms. But how similar the former are to the latter? Out of the 5 GGP-RIs produced by the experiments described in this section (corresponding to 5 runs of the GGP with a different random seed in each run), 3 shared one relatively uncommon characteristic: they added two conditions instead of one condition at-a-time to an empty rule, as shown in Alg. 2. Alg. 2 starts to produce rules with an empty condition, adds two condition-at-a-time to it, evaluates the rule with the new conditions using the Laplace estimation and selects the best 5 produced rules to go on into the refinement process. The algorithm keeps inserting new conditions to the best selected rules until all the examples in the training set are covered, or while the rules found have accuracy superior to 70%.

In other words, Alg. 2 is a variation of CN2 where two conditions are added to a rule at-a-time. The other difference with respect to CN2 lies on the condition used to stop inserting rules to the model (predictive accuracy superior to 70%). But why most of the algorithms produced by the GGP are similar to CN2?

The UCI data sets (Newman *et al.*, 1998) are very popular in the machine learning community, and they have been used to benchmark classification algorithms for a long time. To a certain extent, most of the manually-designed rule induction algorithms were first designed or later modified targeting these data sets. The fact that the evolved rule induction algorithms are similar to CN2 is evidence that CN2 is actually one of the best algorithms in terms of average predictive accuracy in a set of data sets available in the UCI repository. At the same time, as the rule induction algorithms produced by the GGP showed, there are many other variations of the basic sequential covering pseudo-code which obtain accuracies competitive to the ones produced by CN2, Ripper or C4.5Rules. In general, the evolved algorithms did not obtain significantly better accuracies than the baseline classification algorithms, but the former obtained slightly better results than the latter, overall. This can be observed in Table 3, which contains three significant wins (dark gray cells) and just one significant loss (light gray cell) for the evolved algorithms.

6 Conclusions and Future Directions

This work presented a grammar-based genetic programming system which automatically discovers new sequential covering rule induction algorithms. Computational results showed that the system can effectively evolve rule induction algorithms which are competitive in terms of accuracy with well-known human designed rule induction algorithms.

This work opens a whole new area of research, and there are many other directions which could be taken. Improvements to the current work include changing the fitness of the system to use the ROC framework, and studying the impacts this change would have in the created rule induction algorithms.

A more interesting direction, which at the moment is part of our ongoing work, is to automatically create rule induction algorithms tailored to a specific application domain. In other words, we can replace the meta-training and meta-test sets of the current GGP system by a single data set, corresponding to a target application domain, and produce customized rule induction algorithms. This would be a great contribution to the area of meta-learning, in particular, which is putting many efforts into finding which algorithms are the best to mine specific data sets.

Acknowledgments

The first author is financially supported by CAPES, the Brazilian Research Council, process number 165002-5.

References

Aho, A.V., Sethi, R., Ullman, J.D, (1986), Compilers: Principles, Techniques and Tools. 1st edn. Addison-Wesley.

Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D, (1998), Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications. Morgan Kaufmann.

Bhattacharyya, S, (1998), Direct marketing response models using genetic algorithms. In: Proc. of 4th Int. Conf. on Knowledge Discovery and Data Mining (KDD-98). 144–148.

Caruana, R., Niculescu-Mizil, A, (2004), Data mining in metric space: an empirical analysis of supervised learning performance criteria. In: Proc. of the 10th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining (KDD-04), ACM Press 69–78.

Clark, P., Boswell, R., (1991), Rule induction with CN2: some recent improvements. In Kodratoff, Y., ed, EWSL-91: Proc. of the European Working Session on Learning on Machine Learning, New York, NY, USA, Springer-Verlag 151–163.

Clark, P., Niblett, T, (1989), The CN2 induction algorithm. Machine Learning **3** 261–283.

Cohen, W.W., (1995), Fast effective rule induction. In Prieditis, A., Russell, S., eds, Proc. of the 12th Int. Conf. on Machine Learning (ICML-95), Tahoe City, CA, Morgan Kaufmann 115–123.

Fawcett, T, (2003), Roc graphs: Notes and practical considerations for data mining researchers. Technical Report HPL-2003-4, HP Labs.

Flach, P, (2003), The geometry of roc space: understanding machine learning metrics through roc isometrics. In: Proc. 20th International Conference on Machine Learning (ICML-03), AAAI Press 194–201.

Freitas, A.A, (2002), Data Mining and Knowledge Discovery with Evolutionary Algorithms. Springer-Verlag.

Fürnkranz, J, (1999), Separate-and-conquer rule learning. Artificial Intelligence Review **13**(1) 3–54.

de la Iglesia, B., Debuse, J.C.W., Rayward-Smith, V.J, (1996) Discovering knowledge in commercial databases using modern heuristic techniques. In: Proc. of the 2nd ACM SIGKDD Int. Conf. on Knowledge discovery and data mining (KDD-96), 44–49.

Genetic Programming, http://www.genetic-programming.org/ (2006)

Koza, J.R, (1992), Genetic Programming: On the Programming of Computers by the means of natural selection. The MIT Press, Massachusetts.

Michalski, R.S, (1969), On the quasi-minimal solution of the general covering problem. In: Proc. of the 5th Int. Symposium on Information Processing, Bled, Yugoslavia 125–128.

Mitchell, T, (1997), Machine Learning. Mc Graw Hill.

Naur, P, (1963), Revised report on the algorithmic language algol-60. Communications ACM **6**(1) 1–17.

Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J., (1998), UCI Repository of machine learning databases. University of California, Irvine, http://www.ics.uci.edu/∼mlearn/MLRepository.html

Pappa, G.L., Freitas, A.A. (2006), Automatically evolving rule induction algorithms. In Fürnkranz, J., Scheffer, T., Spiliopoulou, M., eds, Proc. of the 17th European Conf. on Machine Learning (ECML-06). Volume 4212 of Lecture Notes in Computer Science., Springer Berlin/Heidelberg 341–352.

Pappa, G.L, (2007), Automatically Evolving Rule Induction Algorithms with Grammar-based Genetic Programming. PhD thesis, Computing Laboratory, University of Kent, Cannterbury, UK.

Provost, F., Fawcett, T., Kohavi, R, (1998), The case against accuracy estimation for comparing induction algorithms. In: Proc. of the 15th Int. Conf. on Machine

Learning (ICML-98), San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. 445–453.

Quinlan, J.R, (1993), C4.5: programs for machine learning. Morgan Kaufmann.

Witten, I.H., Frank, E, (2005), Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. 2^{nd} edn. Morgan Kaufmann.

Zhang, J, (1992), Selecting typical instances in instance-based learning. In: Proc. of the 9^{th} Int. Workshop on Machine learning (ML-92), San Francisco, CA, USA, Morgan Kaufmann 470–479.

Evolutionary Design of Code-matrices for Multiclass Problems

Ana Carolina Lorena[1] and André C. P. L. F. de Carvalho[1]

[1] Centro de Matemática, Computação e Cognição
Universidade Federal do ABC
Rua Catequese, 242,09210-170,
Santo André, SP,
Brazil
ana.lorena@ufabc.edu.br

[2] Ciências de Computação
Instituto de Ciências Matemática e de Computação
Universidade de São Paulo - Campus de São Carlos
Caixa Postal 668 13560-970 São Carlos, SP, Brazil
andre@icmc.usp.br

Summary. Several real problems involve the classification of data into categories or classes. Given a dataset containing data whose classes are known, Machine Learning algorithms can be employed for the induction of a classifier able to predict the class of new data from the same domain, performing the desired discrimination. Several machine learning techniques are originally conceived for the solution of problems with only two classes. In multiclass applications, an alternative frequently employed is to divide the original problem into binary subtasks, whose results are then combined. The decomposition can be generally represented by a code-matrix, where each row corresponds to a codeword assigned for one class and the columns represent the binary classifiers employed. This chapter presents a survey on techniques for multiclass problems code-matrix design. It also shows how evolutionary techniques can be employed to solve this problem.

1 Introduction

Many problems involve the classification of data into categories or classes. Given a training dataset, Machine Learning (ML) (Mitchell, 1997) algorithms can be employed for the induction of a classifier, which should be able to predict the class of new data from the same domain.

A classification problem with only two classes is known as a binary classification problem. An example of a binary classification problem is the medical diagnostic of a particular disease. In this example, the induced classifier uses clinical information from a patient to determine if he/she has a particular disease. The classes represent the presence or absence of the disease. Many real problems, however, involve the discrimination of more than two categories or classes. Examples of such problems are the classification of handwritten digits (Knerr *et al.*, 1992), the distinction of multiple types of cancer (Statnikov *et al.*, 2005) and text categorization (Berger, 1999, Ghani, 2000).

A multiclass classification problem is intrinsically more complex than a binary problem, since the generated classifier must be able to separate the data into a larger number of categories, which also increases the chances of committing classification errors. Let us consider, for example, a balanced classification problem, with similar number of data per class, with equiprobable classes and a random classifier. If the problem is binary, the chance to obtain a correct classification is of 50%. For four classes, this chance reduces to 25%.

Several popular ML techniques are originally designed for the solution of binary classification problems. Among them, one can mention the Support Vector Machines (SVMs) (Cristianini and Shawe-Taylor, 2000) and the Adaboost algorithm (Freund and Schapire, 1997).

Two approaches have been adopted in the literature to deal with multiclass problems using binary classification techniques: adaptation of the internal operations of the classifier training algorithm and decomposition of the multiclass problem into a set of two-class classification problems.

The extension of a binary learning algorithm to a multiclass version may be either impractical or, frequently, not easy to perform (Passerini *et al.*, 2004). For SVMs, in particular, Hsu and Lin (2002) observed that the reformulation of this technique into multiclass versions leads to high cost training algorithms. Therefore, it is common to use the alternative of decomposing the multiclass problem into binary subproblems, a strategy named decomposition.

The decomposition performed can be generally represented by a code-matrix $\mathbf{M}$ (Allwein *et al.*, 2000). This matrix has k rows, representing code-words assigned to each of the k classes in the multiclass problem and the columns correspond to the desired outputs of the binary classifiers induced in the decomposition.

There are several alternatives to decompose the multiclass problem into binary subtasks (Allwein *et al.*, 2000). This chapter surveys recent developments in how to decompose multiclass problems into binary subproblems. This task can be formulated as the problem of designing code-matrices. This chapter also shows how Genetic Algorithms (GAs) (Mitchell, 1999), a search technique based on principles of genetics and natural evolution, can be used to solve the code-matrix design problem.

Initially, Section 2 introduces the code-matrix framework to solve multiclass problems. Section 3 discusses the code-matrix design problem. Section 4 describes how GAs can be used to solve the code-matrix design problem. Section 5 presents a general discussion on experimental results achieved by the described techniques and Section 6 concludes this chapter.

2 Code-matrix Decomposition of Multiclass Problems

Classification using ML techniques consists of inducing a function $f(\mathbf{x})$ from a dataset composed of pairs $(\mathbf{x}_i, y_i)$, where $y_i \in \{1, \ldots, k\}$. Some learning techniques, like the SVMs (Cristianini and Shawe-Taylor, 2000), are originally restricted to classification problems where $k = 2$, i. e., to binary classification problems.

The most common approach for the generalization of binary classification techniques to solve multiclass problems is to decompose the original problem into several binary subproblems. The learning algorithm induces a classifier for each one of these subproblems. The outputs of these classifiers are then combined to obtain the multiclass prediction.

There are several motivations for the use of decomposition strategies in multiclass solutions. Mayoraz and Moreira (1996), Masulli and Valentini (2000) and Frnkranz (2002) state that the use of a decomposition approach may reduce the complexity involved in the classes separation. Herewith, it can also benefit ML techniques whose algorithms are easily extensible to the solution of multiclass problems. Knerr *et al.*. (1992), for example, observed that the classes in a digit recognition problem could be linearly separated when considered in pairs. Therefore, they opted to combine linear classifiers for all pairs of classes, an alternative considered simpler than the use of an unique classifier able to separate all classes simultaneously.

Pimenta (2005b) also points that the decomposition approach opens up new possibilities for the use of parallel processing, since the binary subproblems are independent and can be solved in different processors.

Section 2.1 reviews the main decomposition techniques from the literature. Next, Section 2.2 describes the code-matrix framework, generally used to represent the decomposition strategies.

2.1 Common Decomposition Strategies

The most straightforward decomposition strategy is the one-against-all (1AA). Given a problem with k classes, k binary classifiers are generated by using this strategy. Each classifier is responsible to distinguish a class i from the remaining classes. The final prediction is usually given by the classifier with the highest output value.

Another standard methodology, named one-against-one (1A1), consists of building $k(k-1)/2$ predictors, each differentiating a pair of classes i and j, where $i \neq j$ (Knerr *et al.*, 2000, Hastie and Tibshirani, 1998). To combine the outputs produced by these classifiers, a majority voting scheme can be applied (Kreβel, 1999). Each 1A1 classifier gives one vote to its preferred class. The final result is the class with most of the votes.

Dietterich and Bariki (1995) suggested to see the ML solution of a multiclass problem as a communication task, where the correct class of a new example must be transmitted by a channel. This channel is constituted by the example attributes, the training data and the learning algorithm. Due to errors that can be present in the attributes, in the training data and/or failures in the classifier learning process, the class information can be disrupted. To provide the system the ability to recover from these transmission errors, the class is codified by an error correcting code and each of its bits is transmitted separately, that is, through separate executions of the learning algorithm.

Herewith, a distributed output code is used to represent the k classes associated to the multiclass problem. A codeword of length l is assigned to each class. Commonly, the size of the codewords has more bits than needed in order to represent each class uniquely. The additional bits can be used to correct eventual classification errors. For this reason, this method is named error-correcting output coding (ECOC).

The generated codes are stored on a matrix $\mathbf{M} \in \{-1, +1\}^{k\text{x}l}$. The rows of this matrix represent the codewords of each class and the columns correspond to the desired outputs of the l binary classifiers $(f_1(\mathbf{x}), ..., f_l(\mathbf{x}))$ induced.

A new pattern $\mathbf{x}$ can be classified by evaluating the predictions of the l classifiers, which generate a vector $\mathbf{f}(\mathbf{x})$ of length l. This vector is then compared to the rows of $\mathbf{M}$. The example is assigned to the class with the closest row according to the Hamming distance. This process is also named decoding.

Dietterich and Bariki (1995) proposed codewords to be designed in order to maximize their error correcting capability and presented four techniques for the construction of good error correcting codes. The choice of each technique is determined by the number of classes in the problem. These techniques are briefly described in Section 3.

2.2 Code-Matrix Framework

Allwein *et al.*(2000) presented a framework that unified the previous strategies and can be generally used to represent decomposition techniques. Throughout this framework, the decomposition strategies are reduced to code-matrix based methods. For such, a value from the set $\{-1, 0, +1\}$ is assigned to each element of the code-matrix $\mathbf{M}$. Figure 1 presents an example of code-matrix for a problem with four classes that uses four classifiers in the decomposition of the multiclass problem. It also shows, bellow this matrix, the binary partitions of classes imposed by each of the binary classifiers in the matrix columns.

Each element of the matrix assumes values in the set $\{-1, 0, +1\}$. An element m_{ij} with $+1$ value indicates that the class correspondent to row i assumes a positive label in classifier f_j induction. The -1 value designates a negative label and the 0 value indicates that the data from class i do not participate on classifier f_j induction. Binary classifiers are then trained to learn the labels represented in the columns of $\mathbf{M}$.

In the 1AA case, $\mathbf{M}$ has dimension $k\text{x}k$, with diagonal elements equal to $+1$. The remaining elements are equal to -1. In the 1A1 decomposition, $\mathbf{M}$ has dimension $k\text{x}k(k-1)/2$ and each column corresponds to a binary classifier

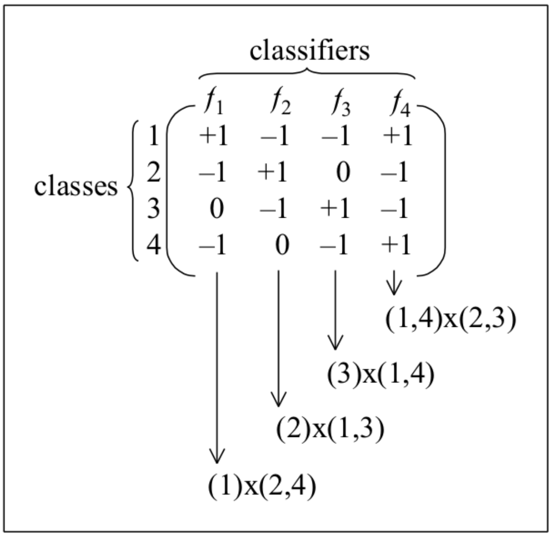


Fig. 1. Example of code-matrix for a problem with four classes

for a pair of classes (i, j). In each column representing a pair (i, j), the value of the elements corresponding to lines i and j are defined as $+1$ and -1, respectively. All other elements receive the value 0, indicating that patterns from the other classes do not participate on the induction of this particular binary classifier.

The prediction of a new pattern's class involves a decoding step, like in the ECOC strategy. Several decoding strategies have been proposed in the literature (Passerini *et al.*, 2004, Allwein *et al.*, 2000, Windeatt and Ghaderi, 2003, Escalera *et al.*, 2006, Klautau *et al.*, 2003). This chapter is concerned with the problem of decomposing the multiclass problem. For this reason, readers interested in the decoding step should look at (Passerini *et al.*, 2004, Allwein *et al.*, 2000, Windeatt and Ghaderi, 2003, Escalera *et al.*, 2006, Klautau *et al.*, 2003) and related publications.

In experimental studies, Allwein *et al.*(2000) have not verified any clear winner among different coding strategies, including 1AA, 1A1, dense random codes and sparse random codes. The authors pointed out the necessity of formulating methods to design problem specific output codes.

Next section describes the code-matrix design problem and reviews some of the main developments in this area.

3 Code-matrix Design Problem

Several alternatives can be employed in order to decompose a multiclass problem into multiple binary subproblems. The most compact decomposition of a problem with k classes can be performed with the use of $l = \lceil \log_2 (k) \rceil$ binary classifiers (Mayoraz and Moreira, 1996). One example of compact matrix for a problem with four classes is presented in Figure 2a.

$$\begin{pmatrix} +1 & +1 \\ +1 & -1 \\ -1 & +1 \\ -1 & -1 \end{pmatrix} \qquad \begin{pmatrix} +1 & +1 & +1 & +1 & +1 & +1 \\ -1 & -1 & -1 & -1 & +1 & +1 \\ -1 & -1 & +1 & +1 & -1 & -1 \\ -1 & +1 & -1 & +1 & -1 & +1 \end{pmatrix}$$

(a) (b)

$$\begin{pmatrix} +1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 \\ -1 & -1 & +1 & -1 \\ -1 & -1 & -1 & +1 \end{pmatrix} \qquad \begin{pmatrix} +1 & +1 & +1 & 0 & 0 & 0 \\ -1 & 0 & 0 & +1 & +1 & 0 \\ 0 & -1 & 0 & -1 & 0 & +1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{pmatrix}$$

(c) (d)

Fig. 2. Different code-matrices for a problem with four classes

The total number of different binary predictors for a problem with k classes is $0.5\left(3^k + 1\right) - 2^k$, considering that $f = -f$, that is, that the inversion of the positive and negative labels produces the same classifier (Mayoraz and Moreira, 1996). Among those, $2^{k-1} - 1$ include all classes simultaneously, i. e., have only the labels $+1$ and -1, without the 0 element. One example of code-matrix constituted of such classifiers for a problem with four classes is illustrated in Figure 2b.

Among the main decomposition strategies reported in the literature one can mention: 1AA, 1A1 (Knerr *et al.*, 2000, Hastie and Tibshirani, 1998) and ECOC (Dietterich and Bariki, 1995). The 1AA and 1A1 code-matrices were already described on Section 2. Figures 2c and 2d represent the 1AA and 1A1 matrices, respectively, for a problem with four classes.

The 1AA decomposition has some disadvantages when the number of examples in one class is much smaller than number of data in other classes. This unbalance may harm the induction of a classifier with good predictive performance in the considered class. In the 1A1 case, the answer of a predictor for a pair of classes (i, j) does not provide useful information when the example does not belong to classes i or j (Alpaydin and Mayoraz, 1999).

Section 3.1 reviews some strategies used in order to obtain ECOC matrices, i. e., code-matrices with error correcting ability. Section 3.2 describes techniques able to adapt code-matrices to each multiclass problem under consideration. Section 3.3 presents other strategies employed in code-matrices obtainment. Although several authors refer to all types of code-matrices for multiclass problems as ECOCs, in this chapter, we will consider as ECOCs only code-matrices developed to have an error correcting capability.

Unless it is explicitly stated, the described works use binary code-matrices, that is, code-matrices with only $+1$ and -1 elements.

3.1 ECOC Design

Dietterich and Bariki (1995) enforce two characteristics necessary to ensure error correcting capability when designing ECOC matrices:

- Row separation;
- Column separation.

Where the separation is measured through Hamming distance, which is equal to the differences between different bit strings[3]. Constant columns (with only positive or negative elements) should also be avoided, since they do not represent a binary decision problem.

Let d_m designate the minimum Hamming distance between any pair of rows of $\mathbf{M}$. The final ECOC multiclass classifier is able to correct at least $\left\lfloor \frac{d_m - 1}{2} \right\rfloor$ errors of the binary classifiers outputs. Since according to the Hamming distance each incorrect prediction implies in deviating one unity from the correct class codeword, committing $\left\lfloor \frac{d_m - 1}{2} \right\rfloor$ errors, the closest codeword will still be that of the correct class (Dietterich and Bariki, 1995). This is the reason why a high row separation is demanded. According to this principle, the 1AA coding is unable to recover from any error, since its d_m is equal to 2. The row separation requirement is also demanded in the design of Error-Correcting Codes (ECC) in telecommunications (Alba and Chicano, 2004).

Besides, to obtain good error correcting codes for the multiclass problem solution, the errors of the binary classifiers induced must be uncorrelated. For such, a column separation is also demanded, that is, the Hamming distance among each pair of columns of $\mathbf{M}$ must be high. If in the learning algorithm the inversion of the positive and negative labels produces the same classifier ($f = -f$), the Hamming distance between each column and the complement of the others must also be maximized.

Based on these observations, Dietterich and Bariki (1995) proposed four techniques to design code-matrices with good error-correcting capability. The choice of each one of them is determined by the number of classes in the multiclass problem. No justificative was given to how the numbers of classes were stipulated for each method.

[3] Recalling that in the ECOC framework, the code-matrix is constituted only of the elements $+1$ and -1 and the Hamming distance is used in the decoding step.

- For $k \leqslant 7$, they recommend the use of an exhaustive code, which consists on the combination of the $2^{k-1}-1$ binary classifiers with labels $+1$ and -1, as illustrated in Figure 1b for a problem with four classes. The codeword of the first class is composed of only $+1$ values. For each other class i, where $i > 1$, it is composed of alternate runs of 2^{k-i} negative (-1) and positive $(+1)$ labels. The d_m distance in the matrix obtained through the exhaustive method is 2^{k-2}.
- If $8 \leqslant k \leqslant 11$, a method that selects columns from the exhaustive code is applied.
- For $k > 11$, there are two options: a method based on the *hill-climbing* algorithm and the generation of BCH codes (Boser and Ray-Chaudhuri, 1960), from the theory of designing good error correcting codes used in communication coding. One problem with BCH codes is that they do not ensure a good column separation.

In a recent work, Pimenta and Gama (2005) proposed an algorithm for the design of ECOCs that presented competitive predictive performance against traditional decompositions, using Decision Trees (DTs) (Quinlan, J.R., 1986) and SVMs as base classifiers. They proposed a function for the evaluation of the ECOCs quality according to their error-correcting properties. An iterative persecution algorithm (PA) was then used to construct the ECOCs. This algorithm adds or removes columns from an initial ECOC, in order to maximize its quality.

In his Msc. Dissertation, Pimenta (2005b) also employed two algorithms originally designed to obtain ECC in telecommunications in the obtainment of ECOC matrices for multiclass problems. The first is the Repulsion Algorithm (RA) (Alba and Chicano, 2004), based on the Physic behavior of equally charged particles on a sphere. Under this situation, the particles will move over the sphere until an equilibrium is reached. In the RA, each codeword is considered a charged particle, positioned in one corner of a hypercube. The movements allowed are to move from one corner to another, which corresponds to invert one bit in the binary codeword. The RA tries to maximize an evaluation function that gets higher as d_m increases. Since the row separation is not required in the design of an ECC, Pimenta adapted the evaluation function in order to penalize matrices with identical or complementary columns. Pimenta also tested an hybrid version of the RA. In this case, GAs are used to design the code-matrices, aiming to maximize the evaluation function. The RA is used in the mutation step of the GA. This hybrid algorithm is described in Section 4.

Experimentally, PA performed better on finding valid ECOCs, where the validity was measured by the criteria of avoiding equal, complementary and constant columns, while RA was the worst method. Among the valid ECOCs generated, in general PA still performed better, obtaining ECOCs with good quality according to the evaluation function proposed by Pimenta and Gama

(2005). Nevertheless, GARA (GA with RA) also designed ECOCs of good quality.

Pimenta and Gama (2005) also suggested a method to determine the number of columns in the ECOC (i. e., the number of classifiers employed in the decomposition), examining an evaluation function based on the number of errors that can be corrected by ECOCs of different sizes.

Zhang *et al.*(2003) proposed the use of Hadamard matrices from the ECC theory in the multiclass decomposition. They point out that these matrices can be considered optimal ECOCs, within the pool of k class codes that combine $k-1$ base learners, where the optimality is measured according to the row and column separations criteria. Nevertheless, the Hadamard matrices are designed with numbers of rows of power two. For others numbers of classes, some rows have to be deleted. Experimentally, these ECOCs performed better than random and 1AA matrices, employing SVMs in the binary classifiers induction.

There are some studies that claim that randomly designed ECOCs show good multiclass predictive performance (Berger, 1999, Windeatt and Ghaderi, 2003, Tapia *et al.*, 2003). Allwein *et al.*(2000), for example, evaluated the use of two randomly designed matrices: dense and sparse. In the dense matrix obtainment, 10,000 random matrices, with $\lceil 10 * \log_2(k) \rceil$ columns and elements assuming -1 or $+1$ values with the same probability, are generated. The matrix with higher d_m and without identical or complementary columns is chosen, following the directions of Dietterich and Bariki (1995) . In the sparse matrix, which uses the ternary alphabet, the number of columns in the code-matrix is $\lceil 15 \log_2 k \rceil$, and the elements are chosen as 0 with 0.5 probability and $+1$ or -1 with probability 0.25 each. Again, 10,000 random matrices are generated and the one with higher d_m is chosen.

Berger (1999) gives statistical and combinatorial arguments of why random matrices can perform well. Among these arguments, are theorems that state that random matrices are likely to show good row and column separations, specially as their number of columns increases. Nevertheless, it is assumed that the errors of the individual predictors are uncorrelated, which do not hold for real applications.

Windeatt and Ghaderi (2002) also express the desirability of equidistant codes. Equidistant codes are those for which the Hamming distance between rows is approximately constant. It was shown that if $\mathbf{M}$ is an equidistant code-matrix, the number of $+1$'s in different rows are the same and the number of common $+1$'s between any pair of rows is equal. They used this heuristic to select a subset of rows from BCH codes, producing equidistant code-matrices. Experimentally, they verified that equidistant codes were superior to 1AA and random codes for shorter codes (with less columns), using Multilayer Perceptron (MLP) Neural Networks (NNs) (Haykin, 1999) as base classifiers. As the length of the codes increases, the coding strategy seems to be less significant, favoring a random design.

3.2 Adapting Code-matrices to the Multiclass Problems

A common criticism to the 1AA, 1A1 and ECOC strategies is that all of them perform the multiclass problem decomposition a priori, without taking into account the properties and characteristics of each application (Allwein *et al.*, 2000, Mayoraz and Moreira, 1996, Alpaydin and Mayoraz, 1999, Mayoraz and Alpaydim, 1998, Dekel and Singer, 2003, Rätsch *et al.*, 2003, Pujol *et al.*, 2006). Besides, Allwein *et al.*(2000) point out that, although the ECOC codes have good error correcting property, several of the binary subproblems created may be difficult to learn.

Crammer and Singer (2000) presented one of the most known attempts to design code-matrices adapted to each multiclass problem considered. They argued that finding a discrete code-matrix can be considered a NP-hard problem and relaxed it allowing that $\mathbf{M}$ had continuous elements. As a result of their work, a version of SVMs for the direct solution of multiclass problems was obtained. Although accuracy results of this technique are comparable to those of the 1AA and 1A1 decomposition strategies used with SVMs (Hsu and Lin, 2002), the complexity of the training algorithm is higher, implying in a high computational cost.

Alpaydin and Mayoraz (1999) proposed to combine linear binary classifiers in order to obtain a non-linear multiclass classifier. In this process, a MLP NN is obtained, in which the first weight layer represents the parameters of the linear classifiers, the internal nodes correspond to the linear classifiers and the final weight layer is equivalent to the code-matrix. This NN has the architecture and second layer weights initialized according to an usual code-matrix. As a result, the code-matrix and classifiers parameters are optimized jointly in the NN training. The proposed method showed higher accuracy than those of 1AA, 1A1 and ECOC decompositions employing linear binary classifiers.

In (Dekel and Singer, 2003), an algorithm named Bunching was introduced, which, during the learning process, adapts code-matrices to the multiclass problem. In this algorithm, the training data and their labels are mapped to a common space. In this space, it is possible to define a function that measures the divergence between the data and their labels. Two matrices are used in the mapping process, one for the data and other for the labels, which is the code-matrix. These two matrices are iteratively adapted by the algorithm in order to obtain a minimum error for the training data. This error is measured by the divergence between the training data and their labels in the common space. The code-matrices are probabilistic. Given an initial code-matrix, the Bunching algorithm modifies it according to the previous procedure. Given a new example, it is mapped to the common space and the predicted class is the one closer to the example in this space. Empirically, the authors verified good results in the adaptation of code-matrices of the 1AA type and random ones, using logistic regression classifiers (Collins *et al.*, 2002).

In (Rätsch *et al.*, 2003), an optimization problem is formulated, in which the codes and weights to the binary functions used in the decomposition are determined jointly. A maximal margin approach is followed, in which the difference between the prediction vector $\mathbf{f}(\mathbf{x})$ to the code of the correct class and the distance to the closer code from another class is maximized. Preliminary experimental results indicated an advantage of this method in relation to the direct use of DTs.

A heuristic method for designing ternary code-matrices based on a hierarchical partition of the classes according to a discriminative criterion was presented in (Pujol *et al.*, 2006). The criterion used was the mutual information between the feature data and its class label. Initiating with all classes, they are recursively partitioned into two subsets in order to maximize the mutual information measure, until each subset contains one class. These partitions define the binary predictors to be employed in the code-matrix. For a problem with k classes, $k-1$ binary classifiers are generated in this process. Experimental results demonstrated the potential of the approach using DTs and Boosted Decision Stumps (BDS) (Freund and Schapire, 1997) as base classifiers. The algorithm showed competitive results against 1AA, 1A1 and random code-matrices.

In (Lorena and Carvalho, 2006), GAs were used to determine ternary code-matrices according to the performance obtained by them in the multiclass problem solution. Another goal of the implemented GA was to minimize the number of columns in the matrices, producing simpler decompositions. This work will be better described in Section 4.

3.3 Other Strategies

This section presents code-matrix design works that could not be fully characterized into one of the classes described in the previous sections, either because they employ alternative criteria in the code-matrix design or because a combination of the error-correcting and adaptiveness criteria is used.

In (Mayoraz and Moreira, 1996), an iterative algorithm to code-matrix design was presented, which takes into account three criteria. The first two are the same required by Dietterich and Bariki (1995) in the construction of ECOCs. The third criterion is that each inserted column must be pertinent, according to the positions of the classes in the input space. A binary partition of classes is considered pertinent if it is easy to learn. The largest contribution of this procedure was the use of classifiers simpler than those from ECOC.

Tapia *et al.*(2001) employed concepts from telecommunications coding theory to propose a particular class of ECOCs named Recursive ECOCs (RECOC). The recursive codes are constructed from component subcodes of small length, which may be weak when working on their own, but strong when working together. This results in an ensemble of ECOCs, where each component subcode defines a local multiclass learner. Another interesting feature of RECOCs, pointed by the authors, is that they allow a regulated degree of randomness in their design. Tapia *et al.*(2003) state that, according the telecommunications theory, a random code is the ideal way to protect information against noise. The decoding of RECOC is adaptive and uses information from the training error of the component subcodes in a belief propagation algorithm, allowing some degree of adaptiveness. Experimentally, the RECOCs achieved good results on a set of benchmark datasets using DTs and Boosted decision stumps (BDS) (Freund and Schapire, 1997).

Following the channel coding theory, Prez-Cruz and Arts-Rodriguez (2002) proposed to use a puncturing mechanism to prune the dependence among the binary classifiers in a code-matrix. This procedure eliminates classifiers that degrade the performance of a previously designed code-matrix, deleting columns from it. As consequence, it permits to obtain less complex multiclass schemes. A ternary coding was employed, that is, the code-matrices could have positive, negative and null elements. Experimentally, they achieved a good performance when puncturing 1A1 and BCH ECOC codes.

The design of code-matrices to maximize diversity measures from the literature on classifier ensembles is proposed in (Kuncheva, 2005). The code-matrices are constructed through the use of an evolutionary algorithm (Mitchell, 1999), which evolves matrices based on their diversity measure performance. However, the proposed framework was not evaluated experimentally on benchmark or real datasets.

In (Shen and Tan, 2005), GAs were employed to find ternary code-matrices in order to optimize an evaluation function based on the margins of separation among codes of different classes and the diversity among columns defined by Kuncheva (2005). These works will be better described in Section 4. Experimental results on multiclass classification of microarray cancer data showed encouraging results compared to other multiclass decomposition strategies, like 1AA and 1A1.

4 Evolutionary Design of Code-Matrices

The code-matrix design problem can be formulated as a search and optimization problem. As stated in the previous section, there are $0.5\left(3^k + 1\right) - 2^k$ different binary predictors for a multiclass problem with k classes. A combinatorial number of associations of these classifiers is possible, determining different decompositions of the multiclass problem. Based on this observation, some works used Evolutionary Algorithms (EAs) (Eiben and Smith, 2003), which are search techniques based on principles of evolution and genetics, to solve the code-matrix design problem.

This section presents a review of these works. It starts with a brief introduction to the main concepts of EAs (Section 4.1). After, as in the previous section, the design of ECOCs is presented (Section 4.2), followed by a description of the design of code-matrices adapted to the solution of each multiclass problem (Section 4.3) and of other strategies employed in the matrices obtainment (Section 4.4).

4.1 Evolutionary Algorithms

According to the natural evolution theory (Darwin, 1859), organisms better adapted to their environment have higher chances to transmit their characteristics to the next generation. Thus, the environment exerts a selection among the individuals of a population, which privileges adapted individuals.

In 1900, the genetics theory was integrated to Darwin's work, introducing concepts that complemented it. One of the most important concepts is the hereditability, which defines how characteristics of an individual are transmitted to its descendants (Eiben and Smith, 2003). Two other aspects are necessary for the occurrence of the natural selection: the reproduction and the presence of variations among the characteristics of the individuals in a population, that is, the presence of genetic variability. If the genetic variability is present in a population, the natural selection may act in order to privilege individuals with characteristics that make then more adapted (or fit) to the environment. Besides, new characteristics can be introduced.

EAs employ these concepts throughout the operation of a set of possible solutions to a problem. This set is named population. The population is iteratively adapted through the application of genetic operators in order to produce solutions each time more apt to solve the problem. Throughout this process, a search procedure is performed, in which the optimal solution with maximal fitness represents the objective to be found or approximated. For this reason, EAs are regarded as search and optimization algorithms and have been applied to several problems, including applications in the areas of control, planning, combinatorial optimization and ML (Beasley, 2000).

The Genetic Algorithms (GAs) (Mitchell, 1999) can be considered one of the main research areas of EAs. They were proposed by John Holland (1975), with the initial aim of studying the adaptation mechanisms that occur in nature and incorporating them into computational systems.

Given an initial population of possible solutions to a problem, referenced as individuals, a GA seeks the global solution by an iterative process. At each iteration, also named generation, a new population is produced, which contains evolutions of individuals selected from the previous generation.

The individuals are encoded by a structure named chromosome. In the basic GA, the chromosomes are represented as bit strings. Each bit, also referenced as a gene, represents the presence (value 1) or absence (value 0) of a particular characteristic in the individual (Bäck, 2000). Nevertheless, there are several other types of encoding and the representation is normally determined according to the characteristics of the problem to be solved. The initial population is normally composed of either random solutions or solutions derived from some heuristic related to the problem.

Next, it is necessary to define how to evaluate the individuals, quantifying the fitness of each one of them to solve the problem. This evaluation is performed by a fitness function, which decodes the information in the individuals' chromosome and obtains a measure of its quality. As the encoding, this function is problem dependent.

From the evaluated population, a selection mechanism will select individuals for the next generation, which will produce offspring to a new population. The selection must privilege the fittest individuals, in accordance to the natural selection principles.

Following the concepts of hereditability, in the reproduction of the selected individuals, their characteristics are combined in order to produce descendants. This combination is performed by a genetic operator named cross-over. The cross-over is a binary operator and is applied to two individuals. These individuals, named parents, have their genes exchanged in order to produce two new individuals, the offspring. Simulating the stochastic nature of evolution, the cross-over is usually applied according to a crossover rate p_c, often in the interval $0.6 \leq p_c \leq 0.9$ (Zitzler *et al.*, 2004). For such, a random number is generated. If it is lower than p_c, the cross-over operator is applied. Otherwise, the parents are directly passed to the next generation.

After the cross-over combination, a variability is introduced to the new solutions by the application of an unary operator, named mutation. The mutation operator alters values of genes of individuals. It is also applied according to a rate p_m, which is usually small, to prevent a high alteration of the population, which would harm the GA convergence.

Other selection operator usually applied to a population is the elitism. In the elitism, a proportion p_e of the fittest individuals of the current population are directly copied into the new population. This prevents the loss of good solutions in the GA processing.

The procedures of generating a population, evaluating its individuals, selection and application of the genetic operators are iterated, and form the base of the GAs.

To stop the execution of a GA, different criteria can be used. The GA may be stopped when a maximum number of generations is reached, when the mean fitness of the population or the best individual does not change for a given number of generations, when the fitness of the individuals in the population become too alike or when a solution with the best known fitness value is found.

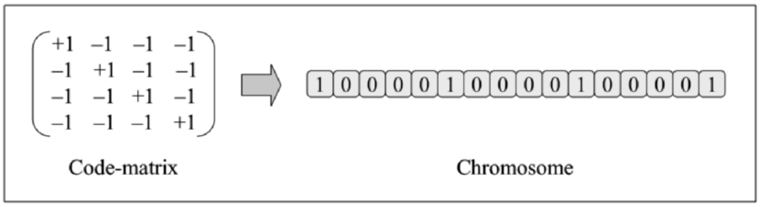
The use of a population approach allied to the genetic operators enhance the chance of finding the optimal solution in the search space when GAs are compared to traditional search techniques, as the hill-climbing algorithm (Michalewicz and Fogel, 2004). The GAs are also able to deal with solution spaces composed of complex and different parts, in which the impact of each part in the problem solution may be difficult to model by traditional search techniques (Mitchell, 1997). They can also take advantage of the use of parallel computation.

Nevertheless, the exploration of populations of solutions also renders the GAs a higher computational cost. GAs have also a set of parameters to be set (p_e, p_c and p_m, for example), whose definition affects their performance in the problem solution.

4.2 Evolutionary ECOC Design

There are several works in communication theory employing GAs to obtain Error Correcting Codes (for example, (Alba and Chicano, 2004, Alba *et al.*, 2002, Simn *et al.*, 2006, Wallis and Houghten, 2002, Dontas and Jong, 1990)). The ECC problem can be summarized as finding a code composed of k codewords with l bits each that corrects a given maximum number of errors, which is known to be a NP-hard optimization problem (Alba and Chicano, 2004). There are conflicting objectives in the ECC design: finding minimum length codewords (which imply in fast transmission) and maximize d_m (for a higher error-correcting capacity), which suggests to include more redundancy in the codewords (more bits and, thus, the use of larger codewords). However, the column separation and avoidance of constant columns are not required under this theory, making difficult to use them directly to find ECOCs for multiclass problems.

Pimenta (2005) adapted the GARA algorithm (Genetic Algorithm with Repulsion Algorithm) (Alba and Chicano, 2004) from the telecommunications theory to the generation of ECOC matrices. The chromosomes in this GA are kxl binary strings, formed by the concatenation of the codewords in the code-matrix **M**, as illustrated in Figure 3. A binary alphabet is used is this work.



$$\begin{pmatrix} +1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 \\ -1 & -1 & +1 & -1 \\ -1 & -1 & -1 & +1 \end{pmatrix}$$
1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1
Code-matrix
Chromosome

Fig. 3. Example of chromosome in GARA (Alba and Chicano, 2004)

Defining d_{ij} as the Hamming distance between codewords i and j in the code-matrix **M** and d_m as in Section 3.1, the fitness function used that evaluates the individuals is presented in Equation 1.

$$f_e(\mathbf{M}) = \frac{1}{\sum_{i=1}^{k}\sum_{j=1}^{k}\frac{1}{d_{ij}^2}} + \left(\frac{d_m}{12} + \frac{d_m^2}{4} + \frac{d_m^3}{6}\right) \tag{1}$$

The first part in the sum of Equation 1 measures how well the codewords in $\mathbf{M}$ are separated in a space of l dimensions. Nevertheless, it may result in a higher value for an ECOC with a lower d_m than that of other ECOC, which is against the desired. The second term is then added to the sum to correct these cases. Overall, f_e is higher for higher d_m matrices and must be maximized by the GA.

To take into account the column separation criterion, Pimenta (2005) added a penalization term to the final fitness evaluation function, which is illustrated by Equation 2.

$$pen\left(\mathbf{M}\right) = 2 * f_e\left(\mathbf{M}\right) * p, \text{ where } \begin{cases} p = 1 \text{ if } \mathbf{M} \text{ has equal columns} \\ p = 1 \text{ if } \mathbf{M} \text{ has complementary columns} \\ p = 0 \text{ otherwise} \end{cases} \tag{2}$$

The final fitness function, which should be maximized by the GA, is then given by Equation 3.

$$f_{it}\left(\mathbf{M}\right) = f_e\left(\mathbf{M}\right) - pen\left(\mathbf{M}\right) \tag{3}$$

The binary tournament selection chooses individuals for reproduction. The cross-over operator used was the single-point cross-over. Details about these operators may be found in (Mitchell, 1999). For mutation, an iteration of the RA was employed, as a local-search procedure. The offspring produced are inserted into a new population if they are better than its current worst individuals. The GA stops when a maximum number of generations is reached or an optimal matrix, according to the fitness function, is found.

As already reported in Section 3.1, the GARA algorithm was compared to the RA and PA algorithms in the obtainment of ECOC code-matrices. In general, PA was better in finding valid ECOCs of good quality, although GARA also obtained good ECOCs.

4.3 Evolutionary Adaptation of Code-matrices to the Problems

As discussed in Section 3.2, many decomposition strategies design the code-matrix a priori. Herewith, they do not take into account the properties and characteristics of each multiclass application.

To overcome this deficiency, a proposal involving the use of GAs to design code-matrices adapted to each multiclass problem was developed (Lorena and Carvalho, 2006). In this proposal, the GA is responsible to determine the combination of binary classifiers in a code-matrix $\mathbf{M}^*$. Herewith, the rows of $\mathbf{M}^*$, which correspond to the codewords attributed to each class, are automatically defined. The GA also determines the number of binary classifiers to be employed in the multiclass solution, that is, the number of columns contained in $\mathbf{M}^*$.

The evaluation of the matrices is based on their predictive performance in the multiclass problem solution. The GA searches for matrices that minimize the error obtained in the multiclass solution. It also aims to minimize the number of columns contained in the matrices, controlling the number of binary classifiers. This criterion represents the search for simpler solutions, and is in accordance to the Occam's razor (Mitchell, 1997), which states that, among several correct hypotheses, the simplest should be chosen. The presence of identical and complementary columns in the matrices must also be avoided, since they represent the use of identical binary classifiers in a decomposition. Columns with equal or complementary elements are denoted as equivalent in the posterior considerations.

The GA must then deal with three objectives: minimize the matrix error and its number of columns and avoid the presence of equivalent columns in the matrix. The aim is to search a code-matrix without equivalent columns with a good performance in the multiclass task and a reduced number of binary classifiers. Two variants of multi-objective GAs were employed by the authors to solve the described problem: a lexicographic, also described in (Lorena and Carvalho, 2006), and another based on the SPEA2 (Strength Pareto Evolutionary Algorithm 2) algorithm (Zitzler *et al.*, 2002).

The chromosomes were directly represented as code-matrices, with a ternary encoding. Each individual corresponds to a possible code-matrix $\mathbf{M}$ with size kxl and elements in the set $\{-1, 0, +1\}$, as described in Section 2.2. This matrix representation is more intuitive to represent solutions to a multiclass problem. Wallet *et al.*(1996) argue that, if the problem has an inherent bidimensional structure, the GA may obtain better results with the use of matrix-codified individuals. The authors also point out that the use of this representation allows the definition of cross-over and mutation operators adequate to this problem.

To determine the number of binary classifiers in the code-matrix, the individuals in a same population had varied numbers of columns l. Herewith, this value is also determined by the GA. According to this strategy, two possible individuals for a problem with four classes are illustrated in Figure 4.

In the codified algorithms, the user limits the maximum allowed number of classifiers for the code-matrices. The generated matrices should also have at least $\lceil \log_2 k \rceil$ binary classifiers, which corresponds to the minimum necessary to divide k classes (Mayoraz and Moreira, 1996).

The initial population was implemented with the definition of random matrices with varying sizes. A consistency test was applied to these individuals, to ensure that each column of the matrices had positive and negative labels, constituting a valid binary partition. The codes of the strategies 1AA, 1A1 and ECOC (exhaustive, dense or sparse random codes) can also be provided to the initial population, adding to the search an additional information.

To evaluate each individual, the GA considers the predictive power of the set of binary classifiers, represented in its code-matrix, for the multiclass problem. For each individual, a validation error is calculated. Unknown classifications, which occur if more than one row of the code-matrix have minimum

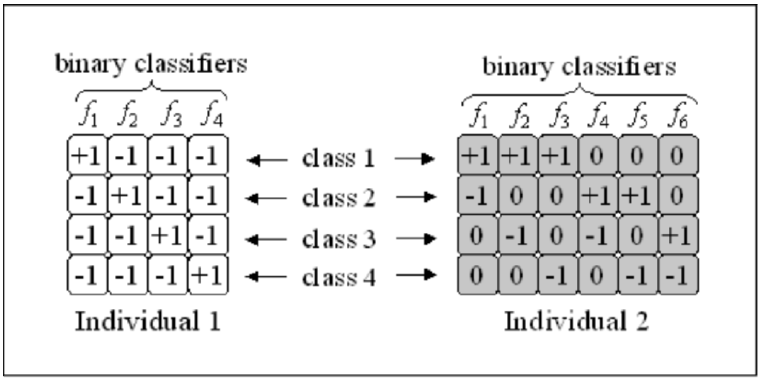


Fig. 4. Illustration of two possible individuals for a problem with four classes

distance to the prediction string, are also considered errors. This error measure should be minimized by the GA.

In the GA operation, a code-matrix may show equivalent columns. To inhibit this occurrence, each multi-objective variant employs a distinct strategy. While the lexicographic version penalizes this characteristic, the SPEA2 algorithm considers the proportion of equivalent columns in the individuals as a third objective to be minimized.

The avoiding of equivalent columns in the lexicographic version was considered a restriction of the problem. Solutions that do not violate this restriction must then be privileged. For such, the fitness of an individual is now calculated by Equation 4 (Deb, 2000). This function must be minimized, so individuals with lower values for Equation 4 are considered to be better.

$$f_{it}\left(i\right) = \begin{cases} e\left(i\right), & \text{if } i \in F \\ \max\limits_{j \in F}\left(e\left(j\right)\right) + p_{ec}\left(i\right), & \text{if } i \in \bar{F} \end{cases} \tag{4}$$

In this equation, F denotes the set of feasible solutions, that is, solutions that do not violate the restriction and do not have equivalent columns. $\bar{F}$, on the other hand, represents non-feasible solutions, $p_{ec}(i)$ represents the proportion of equivalent columns in individual i and $e(i)$ is the validation error rate. Thus, non-feasible solutions have fitness values worst (with higher value of Equation 4) than those of the feasible ones and are compared only in terms of the intensity that they violate the restriction.

As second objective in both GAs is the minimization of the number of binary classifiers in the matrices. The lexicographic version favors the error minimization, placing the reduction of the number of binary classifiers in a second order of importance. In SPEA2, this value was considered as a second objective to be minimized using the Pareto domination relations.

To accomplish the objective ordering in the lexicographic version, first the individuals fitness are calculated using Equation 4. The traditional elitism

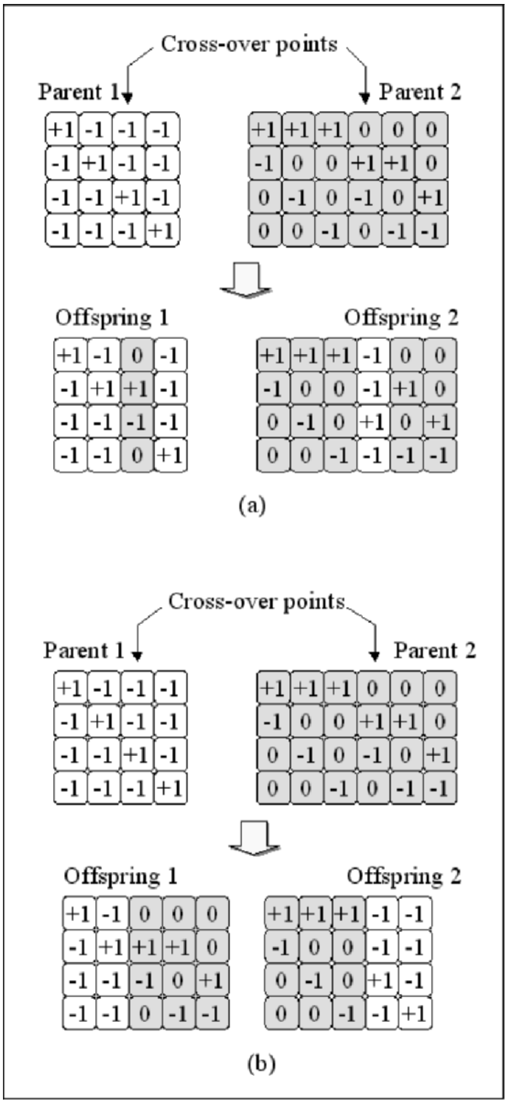


Fig. 5. Cross-over operators for code-matrix design

and selection steps are them adapted. Each time a tie occurs in these steps, the individual with the lowest number of classifiers is chosen.

The GAs stop when a maximum number of generations is reached. A binary tournament described in (Mitchell, 1999) is used in the selection step. The cross-over and mutation genetic operators were designed considering the individuals representation and the characteristics of the formulated code-matrix search problem.

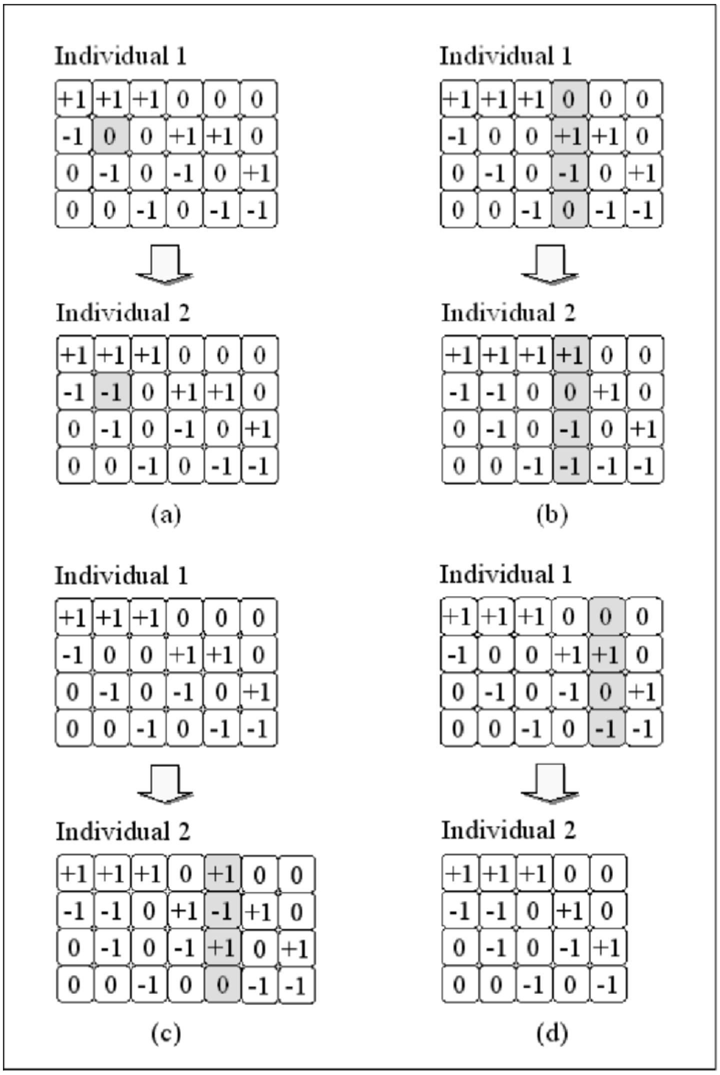


Fig. 6. Mutation operators to code-matrix design

For cross-over, two operators were defined:

- Exchange of columns between two individuals. This operation corresponds to an exchange of binary classifiers, motivated by the fact that a binary predictor can be more efficient in an alternative multiclass combination. This operator is illustrated in Figure 5a.
- Exchange of groups of columns between individuals. In this case, given two individuals, their descendants are produced by permuting all parents columns from randomly chosen points. This operator is illustrated in Fig-

ure 5b. The application of this operator allows the generation of individuals of new sizes, permitting the exploration of code-matrices of varying sizes. If one of the generated offspring has a number of columns outside the minimum and maximum established limits, it is discarded and the corresponding parent is copied into the new generation.

As mutation, four types of operators were defined:

- Change the value of a randomly chosen element of the matrix. This corresponds to the usual mutation operator and is illustrated in Figure 6a.
- New values can also be assigned to all elements in a column, as demonstrated in Figure 6b.
- Given an individual, generate a new column (binary classifier) with random elements. Figure 6c illustrates this modification. This operator can be applied to an individual only if its number of columns is inferior to the maximum value defined.
- Given an individual, remove one of its columns, as illustrated in Figure 6d. This operator can be applied only to individuals whose number of columns is higher than the minimum delimited.

The application of the first three mutation operators may generate columns without negative or positive labels. A consistency check phase must correct theses situations, defining new positive/negative labels.

As there is more than one type of cross-over and mutation operator, which one of them must be applied at each cross-over or mutation step? To opt for one of them, a criterion used in (Martí *et al.*, 2005) was employed. Each possible operator is probabilistically selected according to its performance in the design of good code-matrices in previous generations. Using this scheme, operators that produce better solutions in previous generations have a higher chance to be applied and their importance is adapted by the GA.

At each execution of SPEA2, a set of solutions is obtained. To choose a particular solution, the distance to a reference point is considered. This point presents a null error rate, the minimum number of binary classifiers necessary to distinguish the classes and a null number of equivalent columns. The solution whose evaluations are closer to this point is chosen.

Both GAs were evaluated on a set of benchmark datasets and real multiclass problems from the Bioinformatics domain. They were employed to search for code-matrices with accuracy rates statistically similar or superior to those obtained by the 1AA decomposition (most used in practice (Rifkin and Klautau, 2004)) when using SVMs as base classifiers and with the use of less binary classifiers. The lexicographic GA was able to solve this problem, obtaining code-matrices with good accuracy results and using less binary classifiers. The SPEA2 GA was not successful in this problem. Although the obtained matrices had a low number of binary classifiers, they were not able, in general, to maintain accuracy rates comparable to those of 1AA and also showed equivalent columns.

4.4 Other Evolutionary Strategies

The decomposition framework can be regarded as an ensemble of binary classifiers for the multiclass problem solution. Based on this, Kuncheva (2005) proposed the use of diversity measures from the ensemble literature for generating code-matrices for the multiclass problems. The author argued that measuring diversity through the Hamming distance among columns is insufficient to build accurate ensembles.

The diversity measure used compromises the error-correcting capability in order to have a more diverse ensemble, which shows on average, a better performance. The disagreement measure is used to quantify diversity. It is given by Equation 5 for two codewords i and j, where N^{mn} represents the number of bits for which the codeword i has value m and the codeword j has value n ($m, n \in \{-1, +1\}$ or $\in \{0, 1\}$) and l is the number of columns of $\mathbf{M}$.

$$R_{ij} = \frac{N^{-1+1} + N^{+1-1}}{l} = \frac{\sum_{s=1}^{l} |M(i,s) - M(j,s)|}{l} \tag{5}$$

R_{ij} assumes values between 0 and 1. Larger values are desirable, meaning a larger diversity. The diversity between two rows is then measured by R_{ij}.

For columns, the fact that complementary columns represent the same binary subproblem must be taken into account. The diversity between columns is then given by Equation 6.

$$\begin{aligned} C_{ij} &= \min\left\{\frac{N^{-1+1} + N^{+1-1}}{k}, \frac{N^{-1-1} + N^{+1+1}}{k}\right\} \\ &= \min\left\{\frac{\sum_{s=1}^{k} |M(s,i) - M(s,j)|}{k}, \frac{\sum_{s=1}^{k} |M(s,i) + M(s,j)|}{k}\right\} \end{aligned} \tag{6}$$

For all rows, the total diversity measure is given by the mean of the diversities between all pairs of rows, represented in Equation 7.

$$D_r = \frac{2}{k(k-1)} \sum_{i<j} R_{ij}, \ \ i, j = 1, \ldots, k \tag{7}$$

For columns, the mean is given by Equation 8.

$$D_c = \frac{2}{l(l-1)} \sum_{i<j} C_{ij}, \ \ i, j = 1, \ldots, l \tag{8}$$

To obtain an unique function, Kuncheva (2005) used the average of D_r and D_c, represented in Equation 9.

$$D = \frac{1}{2}(D_r + D_c) \tag{9}$$

It is also possible to obtain a function that measures the row and column separations based on the Hamming distance, as presented in Equation 10, where H_r and H_c are the minimum distances between rows and columns, respectively, and are given by Equations 11 and 12.

$$H = \frac{H_r + H_c}{2} \tag{10}$$

$$H_r = \min_{1 \leqslant i,j \leqslant k} \{R_{ij}\} \tag{11}$$

$$H_c = \min_{1 \leqslant i,j \leqslant l} \{C_{ij}\} \tag{12}$$

EAs were then used to design code-matrices in order to maximize the diversity measure given by Equation 9. As in GARA, the chromosome is represented by a string formed by the concatenation of the codewords in the code-matrix (Figure 3). Only the mutation operator is employed, which is implemented by a bit-flip procedure. To derive a new population, the best ones are chosen from the set formed by the parents and offspring. The EA is stopped after a maximum number of generations.

The evaluation of this EA consisted of verifying whether D or H would be effectively optimized by the EA. No experiments were performed on benchmark or real multiclass datasets.

Using some ideas from the work of Kuncheva (2005), Shen and Tan (2005) also used GAs to search for code-matrices. They adapted the H and D functions to ternary code-matrices. For such, all summations in R_{ij} and C_{ij} were divided by two. Throughout this process, whenever an element is null and another is positive or negative, a value of 0.5 is summed to the computed distance. The chromosomes are again code-matrices with rows concatenated into a bit-string of length kxl.

They define the margin of separation of one class i in relation to the others by Equation 13, where d_{ij} designates the Hamming distance, adapted to the case where null elements are present in the code-matrix (when an element is null and the other is $+1$ or -1, the 0.5 value is added to the distance).

$$\eta_i = \min\{d_{ij}, 1 \leqslant j \leqslant k \,\text{and}\, j \neq i\} \tag{13}$$

This margin measure is based on the rows separation criterion. To maximize all margins simultaneously, the mean of all margins is maximized. To ensure columns separation, D_c is also maximized. This is accomplished by the fitness function presented in Equation 14, which is referred as a multiple margins criterion.

$$m_g = \frac{1}{2k} \sum_{i=1}^{k} \frac{\eta_i}{l} + \frac{1}{2} D_c \tag{14}$$

The GA maximizes the m_g value. When calculating an individual's fitness, Shen and Tan previously remove equivalent or constant columns. Nevertheless, the original size l of the codewords is maintained in the fitness calculations. Throughout this process, code-matrices with these types of columns are penalized.

The GA stops if it cannot improve the fitness values for a given period of time or after a defined consecutive number of generations. The single-point cross-over, uniform mutation and roulette selection were applied (descriptions of these genetic operators may be found in (Mitchell, 1999)).

The proposed GA was evaluated experimentally on two multiclass cancer diagnosis datasets. The GA code-matrices, using linear SVMs as base classifiers, usually outperformed other code-matrices, as 1AA and 1A1, as well as direct multiclass algorithms, like k-nearest neighbor (kNN) (Mitchell, 1997) and DTs. A number of $l = \lceil 10 * \log_2(k) \rceil$ classifiers were used in the matrices. As fitness functions in the GA, they tested both H, D and m_g. The best results were verified using the m_g measure.

5 Discussion of Experimental Results

When describing the code-matrix design approaches in the previous sections, a brief discussion on experiments performed by the authors of each work was presented. The main aspects of these experiments are summarized in Table 1. This table presents, for each of the cited papers, the type of code-matrix design strategy ("CM" column), the number and types of datasets used in their experimental evaluation ("Data" column), the number of classes in the datasets investigated ("♯Classes" column), the base classification techniques used in the binary classifiers induction ("Base cl." column) and the main conclusions obtained from the experimental results. Unless explicitly mentioned, all results were compared based on accuracy or error performance.

The code-matrix design strategy can be of three types, according to the structure adopted in the description of the code-matrix design problem. The types are:

- EC: ECOC design (Sections 3.1 and 4.2);
- AD: adaptation of code-matrices for the multiclass problems (Sections 3.2 and 4.3);
- OS: other strategies (Sections 3.3 and 4.4);

The types of datasets used can also be three, according to their nature:

- A: artificially designed datasets;
- N: natural datasets, which come from benchmarks as the UCI repository (Blake and Merz, 1998);
- R: real datasets, which come from real-world applications;

Regarding the base classifiers, there are works using:

- DT: Decision Trees (Quinlan, J.R., 1986);

- MLP: Multilayer Perceptron Neural Network (Haykin, 1999);
- SVM: Support Vector Machines (Cristianini and Shawe-Taylor, 2000);
- AB: Adaboost (Freund and Schapire, 1997);
- NB: Nave-Bayes (Mitchell, 1997);
- Lin: linear;
- LR: Logistic Regression (Collins *et al.*, 2002);
- BDS: Boosted Decision Stumps (Freund and Schapire, 1997);
- Ran: Random;

Table 1: Summary of experimental evaluation of code-matrix design works

Paper	**CM type**	**Data**	**♯Classes**	**Base cl.**	**Main conclusions**
(Dietterich and Bariki, 1995)	EC	7N/1R	6, 11, 12, 19, 24, 26, 60	DT, MLP	ECOC improves DT and MLP results
(Pimenta and Gama, 2005, Pimenta, 2005)	EC	6N	4 to 6, 10	DT, SVM	Comparable or better than 1AA and 1A1
(Zhang *et al.*, 2003)	EC	6N	6, 8, 10, 11, 26	SVM	Generally better than 1AA and random coding
(Allwein *et al.*, 2000)	EC	13N	6 to 8, 10, 11, 19, 20, 24, 26	SVM, AB	No clear winner among 1AA, 1A1 and random coding
(Berger, 1999)	EC	4R	7, 20, 36, 41	NB	Random codes were effective on text classification
(Windeatt and Ghaderi, 2003)	EC	1A/5N	4 to 7	MLP	Equidistant codes are superior for shorter codes
(Crammer and Singer, 2002, Hsu and Lin, 2002)	AD	1A/10N	3, 4, 6, 7, 11, 26	SVM	Comparable accuracy to 1AA and 1A1 at higher cost
(Alpaydin and Mayoraz, 1999)	AD	8N	3, 7, 10, 11, 26	Lin	Comparable or better than 1AA and 1A1
(Dekel and Singer, 2003)	AD	6N/1R	6, 7, 11, 19, 26	LR	Generally outperforms 1AA and specially random codes
(Rätsch *et al.*, 2003)	AD	2N	3, 6	DT	Could improve DT results
(Pujol *et al.*, 2006)	AD	9N/1R	3, 6, 8, 9, 10, 28	DT, BDS	Comparable or better than 1AA, 1A1 and random codes

Continued on Next Page…

Table 1 – Continued

Paper	**CM type**	**Data**	**♯Classes**	**Base cl.**	**Main conclusions**
(Lorena and Carvalho, 2006, Lorena, 2006)	AD	8N/4R	3, 4 to 10 and 11	SVM	Comparable to 1AA, 1A1 and ECOC with less classifiers
(Mayoraz and Moreira, 1996)	OS	4N	6, 19, 24, 26	DT	Comparable to ECOC with less binary classifiers
(Tapia *et al.*, 2003, Tapia *et al.*, 2001)	OS	6N	4, 6, 7, 22, 24	DT, BDS	RECOCs are suitable to multiclass problems
(Pérez-Cruz and Artés-Rodríguez, 2002)	OS	1N	11	SVM	Good performance puncturing 1A1 and BCH ECOC codes
(Kuncheva, 2005)	OS	1A	50	Rand	Evaluation of GA optimization of H and D measures
(Shen and Tan, 2005)	OS	2R	9, 14	SVM	Comparable or better than 1AA, 1A1, kNN and DTs

A comparison of the experimental results regarding the different strategies introduced in distinct works would bring valuable knowledge. However, it is usually difficult to perform such analysis based on the results presented on the papers. In general, different datasets are used by each author. Even when the same datasets are used, different data partitions are employed to obtain the mean accuracy/error rates reported or different learning techniques are used in the base classifiers induction, making a significant direct comparison impossible.

6 Conclusion

The solution of a multiclass classification problem can be performed through its decomposition into binary subproblems, whose results are later combined. The decomposition can be generally represented by a code-matrix $\mathbf{M}$, whose rows represent codewords assigned to each class and columns represent the binary classifiers desired outputs. How to decompose the multiclass problem can then be reduced to a code-matrix design problem. This chapter surveyed some of the main developments in the design of code-matrices for multiclass problems, with special attention to those using evolutionary computation.

Two general classes of strategies can be used to obtain the codes. The first one considers the error-correcting capability of the codes. The second adapts the codes to each multiclass application. There are, however, works that use a combination of these two strategies or alternative criteria in generation of the code-matrix. Among all reviewed works, some use an evolutionary approach in order to evolve the code-matrices.

From the studies reported, it can be clearly verified that the decomposition of multiclass problems into binary subproblems is an active research area. A good deal of work can be still performed, like comparing different code-matrix design strategies and adapting the GAs in order to use alternative fitness functions.

Acknowledgements

The authors would like to thank the financial support from the Brazilian research agencies CNPq and FAPESP.

References

Alba, E., Cotta, C., Chicano, F., Nebro, A.J., (2002), Parallel evolutionary algorithms in telecommunications: two case studies. In: Proceedings of Congresso Argentino de Ciências de la Computación.

Alba, E., Chicano, J.F., (2004), Solving the error correcting code problem with parallel hybrid heuristics. In: Proceedings of 2004 ACM Symposium on Applied Computing. Volume 2. 985–989.

Allwein, E.L., Shapire, R.E., Singer, Y., (2000), Reducing multiclass to binary: a unifying approach for magin classifiers. In: Proceedings of the 17th International Conference on Machine Learning, Morgan Kaufmann 9–16.

Alpaydin, E., Mayoraz, E., (1999), Learning error-correcting output codes from data. In: Proceedings of the 9th International Conference on Neural Networks. 743–748.

Beasley, D. (2000), (Bäck *et al.*, 2000) 4–18

Berger, A., (1999), Error-correcting output coding for text classification.

Blake, C.L., Merz, C.J., (1998), UCI repository of machine learning databases. Available at: `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Boser, R.C., Ray-Chaudhuri, D.K., (1960), On a class of error-correcting binary group codes. Information and Control **3** 68–79.

Bäck, T., Fogel, D.B., Michalewicz, T., (2000), Evolutionary Computation 1: Basic Algorithms and Operators. Institute of Physics Publishing.

Bäck, T. (2000), (Bäck *et al.*, 2000) 132–135

Collins, M., Shapire, R.E., Singer, Y., (2002), Logistic regression, adaboost and bregman distances. Machine Learning **47**(2/3) 253–285.

Crammer, K., Singer, Y., (2002), On the learnability and design of output codes for multiclass problems. Machine Learning **47**(2-3) 201–233.

Cristianini, N., Shawe-Taylor, J., (2000), An introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press.

Darwin, C., (1859), On the origin of species by means of natural selection. John Murray, London.

Deb, K., (2000), An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering **186** 311–338.

Dekel, O., Singer, Y., (2003), Multiclass learning by probabilistic embeddings. In: Advances in Neural Information Processing Systems. Volume 15., MIT Press 945–952.

Dietterich, T.G., Bariki, G., (1995), Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research **2** 263–286.

Dontas, K., Jong, K.D., (1990), Discovery of maximal distance codes using genetic algorithms. In: Proceedings of the 2nd International IEEE Conference on Tools for Artificial Intelligence, IEEE Computer Society Press 905–811.

Eiben, A.E., Smith, J.E., (2003), Introduction to Evolutionary Computing. Springer.

Escalera, S., Pujol, O., Radeva, R., (2006), Decoding of ternary error correcting output codes. In: Proceedings of the 11th Iberoamerican Congress on Pattern Recognition. Volume 4225 of Lecture Notes in Computer Science., Springer-Verlag 753–763.

Freund, Y., Schapire, R.E., (1997), A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences **1**(55) 119–139.

Fürnkranz, J., (2002), Round robin classification. Journal of Machine Learning Research **2** 721–747.

Ghani, R., (2000), Using error correcting output codes for text classification. In: Proceedings of the 17th International Conference on Machine Learning, Morgan

Kaufmann 303–310.

Hastie, T., Tibshirani, R., (1998), Classification by pairwise coupling. The Annals of Statistics **2** 451–471.

Haykin, S., (1999), Neural Networks - A Compreensive Foundation. 2nd edn. Prentice-Hall, New Jersey.

Holland, J.H., (1975), Adaptation in Natural and Artificial Systems. University of Michigan Press.

Hsu, C.W., Lin, C.J., (2002), A comparison of methods for multi-class support vector machines. IEEE Transactions on Neural Networks **13**(2) 415–425.

Klautau, A., Jevtić, N., Orlistky, A., (2003), On nearest-neighbor error-correcting output codes with application to all-pairs multiclass support vector machines. Journal of Machine Learning Research **4** 1–15.

Knerr, S., Personnaz, L., Dreyfus, G., (1992), Handwritten digit recognition by neural networks with single-layer training. IEEE Transactions on Neural Networks **3**(6) 962–968.

Knerr, S., Personnaz, L., Dreyfus, G., (1990), In: Single-layer learning revisited: a stepwise procedure for building and training a neural network. Springer-Verlag, pp. 41–50

Kreβel, U., (1999), Pairwise classification and support vector machines. In Schölkopf, B., Burges, C.J.C., Smola, A.J., eds.: Advances in Kernel Methods - Support Vector Learning, MIT Press 185–208.

Kuncheva, L.I., (2005), Using diversity measures for generating error-correcting output codes in classifier ensembles. Pattern Recognition Letters **26** 83–90.

Lorena, A.C., Carvalho, A.C.P.L.F., (2006), Evolutionary design of multiclass support vector machines. Journal of Intelligent and Fuzzy Systems . Accepted, to be published..

Lorena, A.C., (2006), Investigação de estratégias para a geração de máquinas de vetores de suporte multiclasses [in portuguese], Ph.D. thesis, Departamento de Ciências de Computação, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, Brazil, `http://www.teses.usp.br/teses/disponiveis/55/55134/tde-26052006-111406`.

Martí, R., Laguna, M., Campos, V., (2005), Scatter search vs. genetic algorithms: An experimental evaluation with permutation problems. In Rego, C., Alidaee, B., eds.: Metaheuristic Optimization Via Adaptive Memory and Evolution: Tabu Search and Scatter Search. Kluwer Academic Publishers 263–282.

Masulli, F., Valentini, G., (2000), Effectiveness of error correcting output codes in multiclass learning problems. In: Proceedings of the 1st International Workshop on Multiple Classifier Systems. Volume 1857 of Lecture Notes in Computer Science., Springer-Verlag 107–116.

Mayoraz, E., Alpaydim, E., (1998), Support vector machines for multi-class classification. Research Report IDIAP-RR-98-06, Dalle Molle Institute for Perceptual Artificial Intelligence, Martigny, Switzerland.

Mayoraz, E., Moreira, M., (1996), On the decomposition of polychotomies into dichotomies. Research Report 96-08, IDIAP, Dalle Molle Institute for Perceptive Artificial Intelligence, Martigny, Valais, Switzerland.

Michalewicz, Z., Fogel, D.B., (2004), How to solve it: modern heuristics. Springer.

Mitchell, T., (1997), Machine Learning. McGraw Hill.

Mitchell, M., (1999), An introduction to Genetic Algorithms. MIT Press.

Passerini, A., Pontil, M., Frasconi, P., (2004), New results on error correcting output codes of kernel machines. IEEE Transactions on Neural Networks **15** 45–54.

Pimenta, E., Gama, J., (2005), A study on error correcting output codes. In: Proceedings of the 2005 Portuguese Conference on Artificial Intelligence, IEEE Computer Society Press 218–223.

Pimenta, E.M.C., (2005), Abordagens para decomposição de problemas multiclasse: os códigos de correcção de erros de saída (in portuguese). Master's thesis, Departamento de Ciências de Computadores, Faculdade de Ciências da Universidade do Porto, Portugal.

Pujol, O., Tadeva, P., Vitrià, J., (2006), Discriminant ECOC: a heuristic method for application dependetn design of error correcting output codes. IEEE Transactions on Pattern Analysis and Machine Intelligence **28**(6) 1007–1012.

Pérez-Cruz, F., Artés-Rodríguez, A., (2002), Puncturing multi-class support vector machines. In: Proceedings of the 12th International Conference on Neural Networks (ICANN). Volume 2415 of Lecture Notes in Computer Science., Springer-Verlag 751–756.

Quinlan, J.R., (1986), Induction of decision trees. Machine Learning **1**(1) 81–106.

Rifkin, R., Klautau, A., (2004), In defense of one-vs-all classification. Journal of Machine Learning Research **5** 1533–7928.

Rätsch, G., Smola, A.J., Mika, S., (2003), Adapting codes and embeddings for polychotomies. In: Advances in Neural Information Processing Systems. Volume 15., MIT Press 513–520.

Shen, L., Tan, E.C., (2005), Seeking better output-codes with genetic algorithm for multiclass cancer classification. Submitted to Bioinformatics.

Simn, M.D.J., Pulido, J.A.G., Rodrguez, M.A.V., (2006), Prez, J.M.S., Criado, J.M.G., A genetic algorithm to design error correcting codes. In: Proceedings of the 13th IEEE Mediterranean Eletrotechnical Conference 2006, IEEE Computer Society Press 807–810.

Statnikov, A., Aliferis, C.F., Tsamardinos, I., (2005), Hardin, D., Levy, S., A comprehensive evaluation of multicategory methods for microarray gene expression cancer diagnosis. Bioinformatics **21**(5) 631–643.

Tapia, E., González, J.C., García-Villalba, J., Villena, J., (2001), Recursive adaptive ECOC models. In: Proceedings of the 10th Portuguese Conference on Artificial Intelligence. Volume 2258 of Lecture Notes in Artificial Intelligence., Springer-Verlag 96–103.

Tapia, E., González, J.C., García-Villalba, J., (2003), Good error correcting output codes for adaptive multiclass learning. In: Proceedings of the 4th International Workshop on Multiple Classifier Systems 2003. Volume 2709 of Lecture Notes in Computer Science., Springer-Verlag 156–165.

Wallet, B.C., Marchette, D.J., Solka, J.L., (1996), A matrix representation for genetic algorithms. In: Automatic object recognition VI, Proceedings of the International Society for Optical Engineering. 206–214.

Wallis, J.L., Houghten, S.K., (2002), A comparative study of search techniques applied to the minimum distance problem of BCH codes. Technical Report CS-02-08, Department of Computer Science, Brock University.

Windeatt, T., Ghaderi, R., (2003), Coding and decoding strategies for multi-class learning problems. Information Fusion **4**(1) 11–21.

Zhang, A., Wu, Z.L., Li, C.H., Fang, K.T., (2003), On hadamard-type output coding in multiclass learning. In: Proceedings of IDEAL. Volume 2690 of Lecture

Notes in Computer Science., Springer-Verlag 397–404.

Zitzler, E., Laumanns, M., Thiele, L., (2002), SPEA2: Improving the strength pareto evolutionary algorithm. In: Evolutionary Methods for Design, Optimisation, and Control, CIMNE, Barcelona, Spain. 95–100.

Zitzler, E., Laumanns, M., Bleuler, S., (2004), A tutorial on evolutionary multiobjective optimization. In Gandibleux, X., Sevaux, M., Srensen, K., T'kindt, V., eds.: Metaheuristics for Multiobjective Optimisation. Volume 535 of Lecture Notes in Economics and Mathematical Systems., Springer-Verlag 3–37.

The Role of Fuzzy Sets in Data Mining

Lior Rokach

Department of Information System Engineering, Ben-Gurion University, Israel
`liorrk@bgu.ac.il`

Summary. In this chapter we discuss how fuzzy logic extends the envelop of the main data mining tasks: clustering, classification, regression and association rules. We begin by presenting a formulation of the data mining using fuzzy logic attributes. Then, for each task, we provide a survey of the main algorithms and a detailed description (i.e. pseudo-code) of the most popular algorithms. However this chapter will not profoundly discuss neuro-fuzzy techniques, assuming that there will be a dedicated chapter for this issue.

1 Introduction

There are two main types of uncertainty in supervised learning: statistical and cognitive. Statistical uncertainty deals with the random behavior of nature and all existing data mining techniques can handle the uncertainty that arises (or is assumed to arise) in the natural world from statistical variations or randomness. While these techniques may be appropriate for measuring the likelihood of a hypothesis, they says nothing about the meaning of the hypothesis.

Cognitive uncertainty, on the other hand, deals with human cognition. Cognitive uncertainty can be further divided into two sub-types: vagueness and ambiguity.

Ambiguity arises in situations with two or more alternatives such that the choice between them is left unspecified. Vagueness arises when there is a difficulty in making a precise distinction in the world.

Fuzzy set theory, first introduced by Zadeh in 1965, deals with cognitive uncertainty and seeks to overcome many of the problems found in classical set theory.

For example, a major problem faced by researchers of control theory is that a small change in input results in a major change in output. This throws the whole control system into an unstable state. In addition there was also the problem that the representation of subjective knowledge was artificial and inaccurate. Fuzzy set theory is an attempt to confront these difficulties and in this chapter we show how it can be used in data mining tasks.

2 Basic Concepts of Fuzzy Set Theory

In this section we present some of the basic concepts of fuzzy logic. The main focus, however, is on those concepts used in the induction process when dealing with data mining. Since fuzzy set theory and fuzzy logic are much broader than the narrow perspective presented here, the interested reader is encouraged to read (Zimmermann, 2005)).

2.1 Membership function

In classical set theory, a certain element either belongs or does not belong to a set. Fuzzy set theory, on the other hand, permits the gradual assessment of the membership of elements in relation to a set.

Definition 1. *Let U be a universe of discourse, representing a collection of objects denoted generically by u. A fuzzy set A in a universe of discourse U is characterized by a membership function μ_A which takes values in the interval [0, 1]. Where $\mu_A(u) = 0$ means that u is definitely not a member of A and $\mu_A(u) = 1$ means that u is definitely a member of A.*

The above definition can be illustrated on the vague set of $Young$. In this case the set U is the set of people. To each person in U, we define the degree of membership to the fuzzy set $Young$. The membership function answers the question "to what degree is person u young?". The easiest way to do this is with a membership function based on the person's age. For example Figure 1 presents the following membership function:

$$\mu_{Young}(u) = \begin{cases} 0 & age(u) > 32 \\ 1 & age(u) < 16 \\ \frac{32-age(u)}{16} & otherwise \end{cases} \tag{1}$$

Given this definition, John, who is 18 years old, has degree of youth of 0.875. Philip, 20 years old, has degree of youth of 0.75. Unlike probability theory, degrees of membership do not have to add up to 1 across all objects and therefore either many or few objects in the set may have high membership. However, an objects membership in a set (such as "young") and the sets complement ("not young") must still sum to 1.

The main difference between classical set theory and fuzzy set theory is that the latter admits to partial set membership. A classical or crisp set, then, is a fuzzy set that restricts its membership values to $\{0, 1\}$, the endpoints of the unit interval. Membership functions can be used to represent a crisp set. For example, Figure 2 presents a crisp membership function defined as:

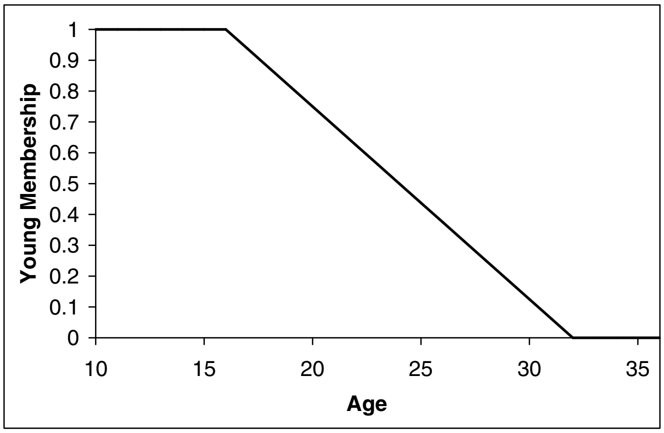


Fig. 1. Membership function for the young set.

$$\mu_{CrispYoung}(u) = \begin{cases} 0 \; age(u) > 22 \\ 1 \; age(u) \leq 22 \end{cases} \tag{2}$$

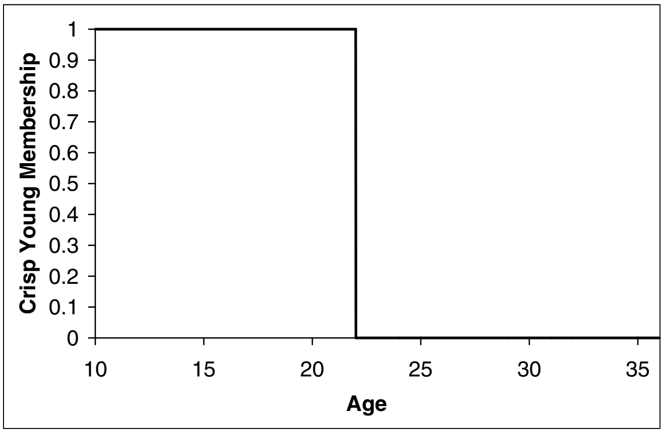


Fig. 2. Membership function for the crisp young set.

In regular classification problems, we assume that each instance takes one value for each attribute and that each instance is classified into only one of the mutually exclusive classes. To illustrate how fuzzy logic can help data mining tasks, we introduce the problem of modelling the preferences of TV viewers. In this problem there are 3 input attributes:

$A = \{\text{Time of Day,Age Group,Mood}\}$

and each attribute has the following values:

- $dom(\text{Time of Day}) = \{\text{Morning,Noon,Evening,Night}\}$
- $dom(\text{Age Group}) = \{\text{Young,Adult}\}$
- $dom(\text{Mood}) = \{\text{Happy,Indifferent,Sad,Sour,Grumpy}\}$

The classification can be the movie genre that the viewer would like to watch, such as $C = \{\text{Action,Comedy,Drama}\}$.

All the attributes are vague by definition. For example, peoples feelings of happiness, indifference, sadness, sourness and grumpiness are vague without any crisp boundaries between them. Although the vagueness of "Age Group" or "Time of Day" can be avoided by indicating the exact age or exact time, a rule induced with a crisp decision tree may then have an artificial crisp boundary, such as "IF Age < 16 THEN action movie". But how about someone who is 17 years of age? Should this viewer definitely not watch an action movie? The viewer preferred genre may still be vague. For example, the viewer may be in a mood for both comedy and drama movies. Moreover, the association of movies into genres may also be vague. For instance the movie "Lethal Weapon" (starring Mel Gibson and Danny Glover) is considered to be both comedy and action movie.

Fuzzy concept can be introduced into a classical problem if at least one of the input attributes is fuzzy or if the target attribute is fuzzy. In the example described above , both input and target attributes are fuzzy. Formally the problem is defined as following (Yuan and Shaw, 1995):

Each class c_j is defined as a fuzzy set on the universe of objects U. The membership function $\mu_{c_j}(u)$ indicates the degree to which object u belongs to class c_j. Each attribute a_i is defined as a linguistic attribute which takes linguistic values from $dom(a_i) = \{v_{i,1}, v_{i,2}, \ldots, v_{i,|dom(a_i)|}\}$. Each linguistic value $v_{i,k}$ is also a fuzzy set defined on U. The membership $\mu_{v_{i,k}}(u)$ specifies the degree to which object u's attribute a_i is $v_{i,k}$. Recall that the membership of a linguistic value can be subjectively assigned or transferred from numerical values by a membership function defined on the range of the numerical value.

Typically, before one can incoporate fuzzy concepts into a data mining application, an expert is required to provide the fuzzy sets for the quantitative attributes, along with their corresponding membership functions. Alternatively the appropriate fuzzy sets are determined using fuzzy clustering.

2.2 Fuzzy Set Operations

Like classical set theory, fuzzy set theory includes operations union, intersection, complement, and inclusion, but also includes operations that have no classical counterpart, such as the modifiers concentration and dilation, and the connective fuzzy aggregation. Definitions of fuzzy set operations are provided in this section.

Definition 2. *The membership function of the union of two fuzzy sets A and B with membership functions μ_A and μ_B respectively is defined as the maximum of the two individual membership functions:*

$$\mu_{A \cup B}(u) = max\{\mu_A(u), \mu_B(u)\} \tag{3}$$

Definition 3. *The membership function of the intersection of two fuzzy sets A and B with membership functions μ_A and μ_B respectively is defined as the minimum of the two individual membership functions:*

$$\mu_{A \cap B}(u) = min\{\mu_A(u), \mu_B(u)\} \tag{4}$$

Definition 4. *The membership function of the complement of a fuzzy set A with membership function μ_A is defined as the negation of the specified membership function:*

$$\mu_{\overline{A}}(u) = 1 - \mu_A(u). \tag{5}$$

To illustrate these fuzzy operations, we elaborate on the previous example. Recall that John has a degree of youth of 0.875. Additionally John's happiness degree is 0.254. Thus, the membership of John in the set Young ∪ Happy would be $max(0.875, 0.254) = 0.875$, and its membership in Young ∩ Happy would be $min(0.875, 0.254) = 0.254$.

It is possible to chain operators together, thereby constructing quite complicated sets. It is also possible to derive many interesting sets from chains of rules built up from simple operators. For example John's membership in the set $\overline{Young}$ ∪ Happy would be $max(1 - 0.875, 0.254) = 0.254$

The usage of the max and min operators for defining fuzzy union and fuzzy intersection, respectively is very common. However, it is important to note that these are not the only definitions of union and intersection suited to fuzzy set theory.

Definition 5. *The fuzzy subsethood $S(A, B)$ measures the degree to which A is a subset of B.*

$$S(A, B) = \frac{M(A \cap B)}{M(A)} \tag{6}$$

where $M(A)$ is the *cardinality* measure of a fuzzy set A and is defined as

$$M(A) = \sum\nolimits_{u \in U} \mu_A(u) \tag{7}$$

The subsethood can be used to measure the truth level of the rule of classification rules. For example given a classification rule such as "IF Age is Young AND Mood is Happy THEN Comedy" we have to calculate $S(Hot \cap Sunny, Swimming)$ in order to measure the truth level of the classification rule.

3 Fuzzy Supervised Learning

In this section we survey supervised methods that incoporate fuzzy sets. Supervised methods are methods that attempt to discover the relationship between input attributes and a target attribute (sometimes referred to as a dependent variable). The relationship discovered is represented in a structure referred to as a model. Usually models describe and explain phenomena, which are hidden in the dataset and can be used for predicting the value of the target attribute knowing the values of the input attributes.

It is useful to distinguish between two main supervised models: classification models (classifiers) and Regression Models. Regression models map the input space into a real-value domain. For instance, a regressor can predict the demand for a certain product given its characteristics. On the other hand, classifiers map the input space into pre-defined classes. For instance, classifiers can be used to classify mortgage consumers as good (fully payback the mortgage on time) and bad (delayed payback).

Fuzzy set theoretic concepts can be incorporated at the input, output, or into to backbone of the classifier. The data can be presented in fuzzy terms and the output decision may be provided as fuzzy membership values. In this chapter we will concentrate on fuzzy decision trees.

3.1 Growing Fuzzy Decision Tree

Decision tree is a predictive model which can be used to represent classifiers. Decision trees are frequently used in applied fields such as finance, marketing, engineering and medicine. In the opinion of many researchers decision trees gained popularity mainly due to their simplicity and transparency. Decision tree are self-explained. There is no need to be an expert in data mining in order to follow a certain decision tree.

There are several algorithms for induction of fuzzy decision trees, most of them extend existing decision trees methods. The UR-ID3 algorithm (Maher and Clair, 1993)) starts by building a strict decision tree, and subsequently fuzzifies the conditions of the tree. Tani and Sakoda (1992) use the ID3 algorithm to select effective numerical attributes. The obtained splitting intervals are used as fuzzy boundaries. Regression is then used in each subspace to form fuzzy rules. Cios and Sztandera (1992) use the ID3 algorithm to convert a decision tree into a layer of a feedforward neural network. Each neuron is represented as a hyperplane with a fuzzy boundary. The nodes within the hidden layer are generated until some fuzzy entropy is reduced to zero. New hidden layers are generated until there is only one node at the output layer.

Fuzzy-CART (Jang (1994)) is a method which uses the CART algorithm to build a tree. However, the tree, which is the first step, is only used to propose fuzzy sets of the continuous domains (using the generated thresholds). Then, a layered network algorithm is employed to learn fuzzy rules. This produces more comprehensible fuzzy rules and improves the CART's initial results.

Another complete framework for building a fuzzy tree including several inference procedures based on conflict resolution in rule-based systems and efficient approximate reasoning methods was presented in (Janikow, 1998).

Olaru and Wehenkel (2003) presented a new type of fuzzy decision trees called soft decision trees (SDT). This approach combines tree-growing and pruning, to determine the structure of the soft decision tree. Refitting and backfitting are used to improve its generalization capabilities. The researchers empirically showed that soft decision trees are significantly more accurate than standard decision trees. Moreover, a global model variance study shows a much lower variance for soft decision trees than for standard trees as a direct cause of the improved accuracy.

Peng (2004) has used FDT to improve the performance of the classical inductive learning approach in manufacturing processes. Peng proposed using soft discretization of continuous-valued attributes. It has been shown that FDT can deal with the noise or uncertainties existing in the data collected in industrial systems.

In this chapter we will focus on the algorithm proposed in (Yuan and Shaw, 1995). This algorithm can handle the classification problems with both fuzzy attributes and fuzzy classes represented in linguistic fuzzy terms. It can also handle other situations in a uniform way where numerical values can be fuzzified to fuzzy terms and crisp categories can be treated as a special case of fuzzy terms with zero fuzziness. The algorithm uses classification ambiguity as fuzzy entropy. The classification ambiguity directly measures the quality of classification rules at the decision node. It can be calculated under fuzzy partitioning and multiple fuzzy classes.

The fuzzy decision tree induction consists of the following steps:

- Fuzzifying numeric attributes in the training set.
- Inducing a fuzzy decision tree.
- Simplifying the decision tree.
- Applying fuzzy rules for classification.

Fuzzifying numeric attributes

When a certain attribute is numerical, it needs to be fuzzified into linguistic terms before it can be used in the algorithm. The fuzzification process can be performed manually by experts or can be derived automatically using some sort of clustering algorithm. Clustering groups the data instances into subsets in such a manner that similar instances are grouped together; different instances belong to different groups. The instances are thereby organized into an efficient representation that characterizes the population being sampled.

Yuan and Shaw (1995) suggest a simple algorithm to generate a set of membership functions on numerical data. Assume attribute a_i has numerical value x from the domain X. We can cluster X to k linguistic terms $v_{i,j}, j = 1, \dots, k$. The size of k is manually predefined. For the first linguistic term $v_{i,1}$, the following membership function is used:

$$\mu_{v_{i,1}}(x) = \begin{cases} 1 & x \leq m_1 \\ \frac{m_2 - x}{m_2 - m_1} & m_1 < x < m_2 \\ 0 & x \geq m_2 \end{cases} \tag{8}$$

For each $v_{i,j}$ when $j = 2, \ldots, k-1$ has a triangular membership function as follows:

$$\mu_{v_{i,j}}(x) = \begin{cases} 0 & x \leq m_{j-1} \\ \frac{x - m_{j-1}}{m_j - m_{j-1}} & m_{j-1} < x \leq m_j \\ \frac{m_{j+1} - x}{m_{j+1} - m_j} & m_j < x < m_{j+1} \\ 0 & x \geq m_{j+1} \end{cases} \tag{9}$$

Finally the membership function of the last linguistic term $v_{i,k}$ is:

$$\mu_{v_{i,k}}(x) = \begin{cases} 0 & x \leq m_{k-1} \\ \frac{x - m_{k-1}}{m_k - m_{k-1}} & m_{k-1} < x \leq m_k \\ 1 & x \geq m_k \end{cases} \tag{10}$$

Figure 3 illustrates the creation of four groups defined on the age attribute: "young", "early adulthood", "middle-aged" and "old age". Note that the first set ("young") and the last set ("old age") have a trapezoidal form which can be uniquely described by the four corners. For example, the "young" set could be represented as $(0, 0, 16, 32)$. In between, all other sets ("early adulthood" and "middle-aged") have a triangular form which can be uniquely described by the three corners. For example, the set "early adulthood" is represented as $(16, 32, 48)$.

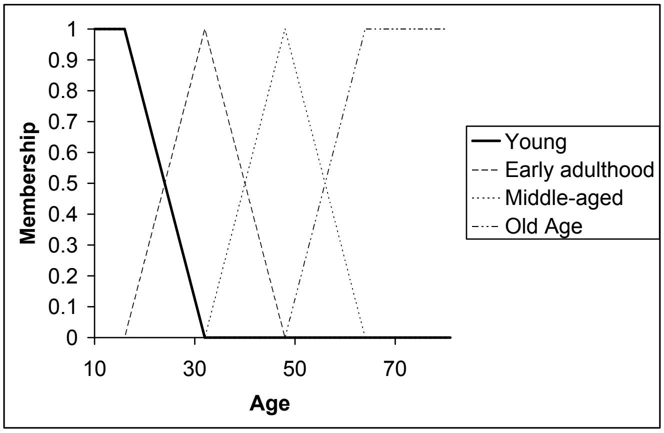


Fig. 3. Membership function for various groups in the age attribute.

The only parameters that need to be determined are the set of k centers $M = \{m_1, \ldots, m_k\}$. The centers can be found using the algorithm presented in

Algorithm 1. Note that in order to use the algorithm, a monotonic decreasing learning rate function should be provided.

Algorithm 1: Algorithm for fuzzifying numeric attributes

```
Input: $X$ - a set of values, $\eta(t)$ - some monotonic decreasing scalar function
    representing the learning rate.
Output: $M = \{m_1, \ldots, m_k\}$
1: Initially set $m_i$ to be evenly distributed on the range of $X$.
2: $t \leftarrow 1$
3: repeat
4:    Randomly draw one sample $x$ from $X$
5:    Find the closest center $m_c$ to $x$.
6:    $m_c \leftarrow m_c + \eta(t) \cdot (x - m_c)$
7:    $t \leftarrow t + 1$
8:    $D(X, M) \leftarrow \sum_{x \in X} \min_i \|x - m_i\|$
9: until $D(X, M)$ converges
```

The Induction Phase

The induction algorithm of fuzzy decision tree is presented in Algorithm 2. The algorithm measures the classification ambiguity associated with each attribute and split the data using the attribute with the smallest classification ambiguity. The classification ambiguity of attribute a_i with linguistic terms $v_{i,j}, j = 1, \ldots, k$ on fuzzy evidence S, denoted as $G(a_i|S)$, is the weighted average of classification ambiguity calculated as:

$$G(a_i\,|S\,) = \sum_{j=`1}^{k} w(v_{i,j}\,|S) \cdot G(v_{i,j}\,|S) \tag{11}$$

where $w(v_{i,j}\,|S)$ is the weight which represents the relative size of $v_{i,j}$ and is defined as:

$$w(v_{i,j}\,|S) = \frac{M(v_{i,j}\,|S)}{\sum\limits_{k} M(v_{i,k}\,|S)} \tag{12}$$

The classification ambiguity of $\mathrm{v}_{i,j}$ is defined as $G(v_{i,j}\,|S) = g\left(\mathbf{p}\left(C\,|v_{i,j}\,\right)\right)$, which is measured based on the possibility distribution vector $\mathbf{p}\left(C\,|v_{i,j}\,\right) = \left(p\left(c_1\,|v_{i,j}\,\right), ..., p\left(c_{|\mathrm{k}|}\,|v_{i,j}\,\right)\right)$.

Given $v_{i,j}$, the possibility of classifying an object to class c_l can be defined as:

$$p\left(c_l\left|v_{i,j}\right.\right)=\frac{S(v_{i,j},c_l)}{\max\limits_{k} S(v_{i,j},c_k)} \tag{13}$$

where $S(A, B)$ is the fuzzy subsethood that was defined in Definition 5. The function $g(\mathbf{p})$ is the possibilistic measure of ambiguity or nonspecificity and is defined as:

$$g(\mathbf{p})=\sum_{i=1}^{|\mathbf{p}|}\left(p_i^*-p_{i+1}^*\right)\cdot\ln(i) \tag{14}$$

where $\mathbf{p}^*=\left(p_1^*,\ldots,p_{|\mathbf{p}|}^*\right)$ is the permutation of the possibility distribution $\mathbf{p}$ sorted such that $p_i^*\geq p_{i+1}^*$.

All the above calculations are carried out at a predefined significant level α. An instance will take into consideration of a certain branch $v_{i,j}$ only if its corresponding membership is greater than α. This parameter is used to filter out insignificant branches.

After partitioning the data using the attribute with the smallest classification ambiguity, the algorithm looks for nonempty branches. For each nonempty branch, the algorithm calculates the truth level of classifying all instances within the branch into each class. The truth level is caluclated using the fuzzy subsethood measure $S(A, B)$.

If the truth level of one of the classes is above a predefined threshold β then no additional partitioning is needed and the node become a leaf in which all instance will be labeled to the class with the highest truth level. Otherwise the procedure continues in a recursive manner. Note that small values of β will lead to smaller trees with the risk of underfitting. A higher β may lead to a larger tree with higher classification accuracy. However, at a certain point, higher values β may lead to overfitting.

Algorithm 2: Fuzzy decision tree induction

```
Input: $S$ - Training Set $A$ - Input Feature Set $y$ - Target Feature
Output: Fuzzy Decision Tree
 1: Create a new fuzzy tree $FT$ with a single root node.
 2: if $S$ is empty OR Truth level of one of the classes $\geq \beta$ then
 3:     Mark $FT$ as a leaf with the most common value of $y$ in $S$ as a label.
 4:     Return $FT$.
 5: end if
 6: $\forall a_i \in A$ find $a$ with the smallest classification ambiguity.
 7: for each outcome $v_i$ of $a$ do
 8:     Recursively call procedure with corresponding partition $v_i$.
 9:     Connect the root to the subtree with an edge that is labeled as $v_i$.
10: end for
11: Return $FT$
```

Simplifying the decision tree

Each path of branches from root to leaf can be converted into a rule with the condition part representing the attributes on the passing branches from the root to the leaf and the conclusion part representing the class at the leaf with the highest truth level classification. The corresponding classification rules can be further simplified by removing one input attribute term at a time for each rule we try to simplify . Select the term to remove with the highest truth level of the simplified rule. If the truth level of this new rule is not lower than the threshold β or the truth level of the original rule, the simplification is successful. The process will continue until no further simplification is possible for all the rules.

Using the Fuzzy Decision Tree

In a regular decision tree, only one path (rule) can be applied for every instance. In a fuzzy decision tree, several paths (rules) can be applied for one instance. In order to classify an unlabeled instance, the following steps should be performed (Yuan and Shaw, 1995):

- Step 1: Calculate the membership of the instance for the condition part of each path (rule). This membership will be associated with the label (class) of the path.
- Step 2: For each class calculate the maximum membership obtained from all applied rules.
- Step 3: An instance may be classified into several classes with different degrees based on the membership calculated in Step 2.

3.2 Soft Regression

Regressions are used to compute correlations among data sets. The "classical" approach uses statistical methods to find these correlations. Soft regression is used when we want to compare data sets that are temporal and interdependent. The use of fuzzy logic can overcome many of the difficulties associated with the classical approach. The fuzzy techniques can achieve greater flexibility, greater accuracy and generate more information in comparison to econometric modeling based on (statistical) regression techniques. In particular, the fuzzy method can potentially be more successful than conventional regression methods, especially under circumstances that severely violate the fundamental conditions required for the reliable use of conventional methods.

Soft regression techniques have been proposed in (Shnaider et al., 1991, Shnaider and Schneider, 1988).

3.3 Neuro-fuzzy

Neuro-fuzzy refers to hybrids of artificial neural networks and fuzzy logic. Neuro-fuzzy is the most visible hybrid paradigm and has been adequately investigated (Mitra and Pal, 2005)

Neuro-fuzzy hybridization can be done in two ways (Mitra, 2000): fuzzy-neural network (FNN) which is a neural network equipped with the capability of handling fuzzy information and a neural-fuzzy system (NFS) which is a fuzzy system augmented by neural networks to enhance some of its characteristics like flexibility, speed, and adaptability.

A neurofuzzy system can be viewed as a special 3layer neural network (Nauck, 1997). The first layer represents input variables, the hidden layer represents fuzzy rules and the third layer represents output variables. Fuzzy sets are encoded as (fuzzy) connection weights. Usually after learning the obtained model is interpreted as a system of fuzzy rules.

4 Fuzzy Clustering

The goal of clustering is descriptive, that of classification is predictive. Since the goal of clustering is to discover a new set of categories, the new groups are of interest in themselves, and their assessment is intrinsic. In classification tasks, however, an important part of the assessment is extrinsic, since the groups must reflect some reference set of classes.

Clustering of objects is as ancient as the human need for describing the salient characteristics of men and objects and identifying them with a type. Therefore, it embraces various scientific disciplines: from mathematics and statistics to biology and genetics, each of which uses different terms to describe the topologies formed using this analysis. From biological "taxonomies", to medical "syndromes" and genetic "genotypes" to manufacturing "group technology" — the problem is identical: forming categories of entities and assigning individuals to the proper groups within it.

Clustering groups data instances into subsets in such a manner that similar instances are grouped together, while different instances belong to different groups. The instances are thereby organized into an efficient representation that characterizes the population being sampled. Formally, the clustering structure is represented as a set of subsets $C = C_1, \ldots, C_k$ of S, such that: $S = \bigcup_{i=1}^{k} C_i$ and $C_i \cap C_j = \emptyset$ for $i \neq j$. Consequently, any instance in S belongs to exactly one and only one subset.

Traditional clustering approaches generate partitions; in a partition, each instance belongs to one and only one cluster. Hence, the clusters in a hard clustering are disjointed. Fuzzy clustering extends this notion and suggests a *soft clustering* schema. In this case, each pattern is associated with every cluster using some sort of membership function, namely, each cluster is a fuzzy set of all the patterns. Larger membership values indicate higher confidence in the assignment of the pattern to the cluster. A hard clustering can be obtained from a fuzzy partition by using a threshold of the membership value.

The most popular fuzzy clustering algorithm is the fuzzy c-means (FCM) algorithm. Even though it is better than the hard K-means algorithm at avoiding local minima, FCM can still converge to local minima of the squared error criterion. The design of membership functions is the most important problem in fuzzy clustering; different choices include those based on similarity decomposition and centroids of clusters. A generalization of the FCM algorithm has been proposed through a family of objective functions. A fuzzy c-shell algorithm and an adaptive variant for detecting circular and elliptical boundaries have been presented.

FCM is an iterative algorithm. The aim of FCM is to find cluster centers (centroids) that minimize a dissimilarity function. To accommodate the introduction of fuzzy partitioning, the membership matrix(U) is randomly initialized according to Equation 15.

$$\sum_{i=1}^{c} u_{ij} = 1, \forall j = 1, ..., n \tag{15}$$

The algorithm minimizes a dissimilarity (or distance) function which is given in Equation 16:

$$J(U, c_1, c_2, ..., c_c) = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^m d_{ij}^2 \tag{16}$$

where, u_{ij} is between 0 and 1; c_i is the centroid of cluster i; d_{ij} is the Euclidian distance between i-th centroid and j-th data point; m is a weighting exponent.

To reach a minimum of dissimilarity function there are two conditions. These are given in Equation 17 and Equation 18.

$$c_i = \frac{\sum_{j=1}^{n} u_{ij}^m x_j}{\sum_{j=1}^{n} u_{ij}^m} \tag{17}$$

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d_{ij}}{d_{kj}} \right)^{2/(m-1)}} \tag{18}$$

Algorithm 3 presents the fuzzy c-means that was originally proposed in (Bezdek, 1973).

By iteratively updating the cluster centers and the membership grades for each data point, FCM iteratively moves the cluster centers to the "right" location within a data set. However, FCM does not ensure that it converges to an optimal solution. The random initilization of U might have uncancelled effect on the final performance.

There are several extensions to the basic FCM algorithm, The Fuzzy Trimmed C Prototype (FTCP) algorithm (Kim et al., 1996) increases the

Algorithm 3: FCM Algorithm

```
Input: $X$ - Data Set
    $c$ - number of clusters
    $t$ - convergence threshold (termination criterion)
    $m$ - exponential weight
Output: $U$ - membership matrix
1: Randomly initialize matrix $U$ with $c$ clusters and fulfils Eq. 15
2: repeat
3:    Calculate $c_i$ by using Equation 17.
4:    Compute dissimilarity between centroids and data points using Eq. 16.
5:    Compute a new $U$ using Eq. 18
6: until The improvement over previous iteration is below $t$.
```

robustness of the clusters by trimming away observations with large residuals. The Fuzzy C Least Median of Squares (FCLMedS) algorithm (Nasraoui and Krishnapuram, 1997) replaces the summation presented in Equation 16 with the median.

5 Fuzzy Association Rules

Association rules are rules of the kind "70% of the customers who buy vine and cheese also buy grapes". While the traditional field of application is market basket analysis, association rule mining has been applied to various fields since then, which has led to a number of important modifications and extensions.

In this section, an algorithm based on the *apriori* data mining algorithm is described to discover large itemsets. Fuzzy sets are used to handle quantitative values, as described in (Hong et al., 1999). Our algorithm is applied with some differences. We will use the following notation:

- n – number of transactions in the database.
- m – number of items (attributes) in the database.
- d_i – the i-th transaction.
- I_j – the j-th attribute.
- I_{ij} – the value of $I_j for d_i$.
- μ_{ijk} – the membership grade of I_{ij} in the region k.
- R_{jk} – the k-th fuzzy region of the attribute I_j.
- $num(R_{jk})$ – number of occurrences of the attribute region R_{jk} in the whole database, where $\mu_{ijk} > 0$.
- C_r – the set of candidate itemsets with r attributes.
- c_r – candidate itemset with r attributes.
- f_j^i – the membership value of d_i in region s_j.
- f_{cr}^i – the fuzzy value of the itemset c_r in the transaction d_i.

Algorithm 4: Fuzzy Association Rules Algorithm

```
1: for all transaction $i$ do
2:    for all attribute $j$ do
3:       $I^f_{ij}$=($\mu_{ij1}/\mathrm{R}_{j1} + \mu_{ij2}/\mathrm{R}_{j2}+\ldots+\mu_{ijk}/\mathrm{R}_{jk}$) {where the superscript
         $f$ denotes fuzzy set}
4:    end for
5: end for
6: For each attribute region $\mathrm{R}_{jk}$, count the number of occurrences, where
   $\mu_{ijk} > 0$, in the whole database. The output is $num(R_{jk})$.
   $num(R_{jk}) = \sum_{i=1}^{n} 1\{\mu_{ijk}/R_{jk} \neq 0\}$
7: $L_1$={$\mathrm{R}_{jk}$|num($\mathrm{R}_{jk}$) ≥minnum, 1≤j≤m, 1≤k≤numR($\mathrm{I}_j$)}.
8: r=1 (r is the number of items that composed the large itemsets in the
   current stage).
9: Generate the candidate set $\mathrm{C}_{r+1}$ from $\mathrm{L}_r$
10: for all newly formed candidate itemset $\mathrm{c}_{r+1}$ in $\mathrm{C}_{r+1}$, that is composed of
    the items ($\mathrm{s}_1$, $\mathrm{s}_2$,...,$\mathrm{s}_{r+1}$) do
11:    For each transaction $d_i$ calculate its intersection fuzzy value as:
       $f^i_{(cr+1)} = f^i_1 \cap \mathrm{f}^i_2 \cap \ldots \cap \mathrm{f}^i_{r+1}$.
12:    Calculate the frequency of $\mathrm{c}_{r+1}$ on the transactions, where $f^i_{(cr+1)} > 0$.
       $num(c_{r+1})$ is output.
13:    If the frequency of the itemset is larger than or equal to the predefined
       number of occurrences minnum, put it in the set of large r+1-itemsets
       $L_{r+1.}$
14: end for
15: if $L_{r+1}$ is not empty then
16:    $r = r + 1$
17:    go to Step 9.
18: end if
19: for all large itemset $l_r$, r≥ 2 do
20:    Calculate its support as: sup($\mathrm{l}_r$) = $\Sigma \mathrm{f}^i_{(lr)}$.
21:    Calculate its strength as: str($\mathrm{l}_r$)= sup($\mathrm{l}_r$)/num($\mathrm{l}_r$).
22: end for
23: For each large itemset $\mathrm{l}_r$, r≥2, generate the possible association rules as
    in (Agrawal et al., 1993).
24: For each association rule $s_1, s_2, \ldots, s_n \geq s_{n+1}, \ldots, s_r$, calculate its
    confidence as: num($\mathrm{s}_1$, $\mathrm{s}_2 \ldots \mathrm{s}_n$,$\mathrm{s}_{n+1} \ldots \mathrm{s}_r$)/num($\mathrm{s}_1$, $\mathrm{s}_2 \ldots \mathrm{s}_n$).
25: if the confidence is higher than the predefined threshold minconf then
26:    output the rule as an association rule.
27: end if
28: For each association rule $s_1, s_2, \ldots, s_n \geq s_{n+1}, \ldots, s_r$, record its strength
    as str($\mathrm{s}_1$, $\mathrm{s}_2 \ldots \mathrm{s}_n$,$\mathrm{s}_{n+1} \ldots \mathrm{s}_r$), and its support as sup($\mathrm{l}_r$).
```

- L_r – the set of large itemsets with r items.
- l_r – a large itemset with r items.
- $num(I_1, \ldots, I_s)$ – the occurrences number of the itemset $(I_1, \ldots, I_s)$.
- $numR(I_j)$ – the number of the membership function regions for the attribute I_j.

Algorithm 4 presents the fuzzy association algorithm proposed in (Komem and Schneider, 2005). The quantitative values are first transformed into a set of membership grades, by using predefined membership functions. Every membership grade represents the agreement of a quantitative value with a linguistic term. In order to avoid discriminating the importance level of data, each point must have membership grade of 1 in one membership function; Thus, the membership functions of each attribute produce a continuous line of $\mu = 1$. Additionally, in order to diagnose the bias direction of an item from the center of a membership function region, almost each point get another membership grade which is lower than 1 in other membership functions region. Thus, each end of membership function region is touching, close to, or slightly overlapping an end of another membership function (except the outside regions, of course).

By this mechanism, as point “a” moves right, further from the center of the region “middle”, it gets a higher value of the label “middle-high”, additionally to the value 1 of the label “middle”.

6 Conclusion

This chapter discussed how fuzzy logic can be used to solve several different data mining tasks, namely classification clustering, and discovery of association rules. The discussion focused mainly one representative algorithm for each of these tasks.

There are at least two motivations for using fuzzy logic in data mining, broadly speaking. First, as mentioned earlier, fuzzy logic can produce more abstract and flexible patterns, since many quantitative features are involved in data mining tasks. Second, the crisp usage of metrics is better replaced by fuzzy sets that can reflect, in a more natural manner, the degree of belongingness/membership to a class or a cluster.

References

R. Agrawal, T. Imielinski and A. Swami: Mining Association Rules between Sets of Items in Large Databases. Proceeding of ACM SIGMOD, 207-216. Washington, D.C, 1993.

J. C. Bezdek. Fuzzy Mathematics in Pattern Classification. PhD Thesis, Applied Math. Center, Cornell University, Ithaca, 1973.

Cios K. J. and Sztandera L. M., Continuous ID3 algorithm with fuzzy entropy measures, Proc. IEEE lnternat. Con/i on Fuzz)' Systems,1992, pp. 469-476.

T.P. Hong, C.S. Kuo and S.C. Chi: A Fuzzy Data Mining Algorithm for Quantitative Values. 1999 Third International Conference on Knowledge-Based Intelligent Information Engineering Systems. Proceedings. IEEE 1999, pp. 480-3.

T.P. Hong, C.S. Kuo and S.C. Chi: Mining Association Rules from Quantitative Data. Intelligent Data Analysis, vol.3, no.5, nov. 1999, pp363-376.

Jang J., "Structure determination in fuzzy modeling: A fuzzy CART approach," in Proc. IEEE Conf. Fuzzy Systems, 1994, pp. 480485.

Janikow, C.Z., Fuzzy Decision Trees: Issues and Methods, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 28, Issue 1, pp. 1-14. 1998.

Kim, J., Krishnapuram, R. and Dav, R. (1996). Application of the Least Trimmed Squares Technique to Prototype-Based Clustering, Pattern Recognition Letters, 17, 633-641.

Joseph Komem and Moti Schneider, On the Use of Fuzzy Logic in Data Mining, in The Data Mining and Knowledge Discovery Handbook, O. Maimon, L. Rokach (Eds.), pp. 517-533, Springer, 2005.

Maher P. E. and Clair D. C, Uncertain reasoning in an ID3 machine learning framework, in Proc. 2nd IEEE Int. Conf. Fuzzy Systems, 1993, pp. 712.

S. Mitra, Y. Hayashi, "Neuro-fuzzy Rule Generation: Survey in Soft Computing Framework." IEEE Trans. Neural Networks, Vol. 11, N. 3, pp. 748-768, 2000.

S. Mitra and S. K. Pal, Fuzzy sets in pattern recognition and machine intelligence, Fuzzy Sets and Systems 156 (2005) 381386

Nasraoui, O. and Krishnapuram, R. (1997). A Genetic Algorithm for Robust Clustering Based on a Fuzzy Least Median of Squares Criterion, Proceedings of NAFIPS, Syracuse NY, 217-221.

Nauck D., Neuro-Fuzzy Systems: Review and Prospects Paper appears in Proc. Fifth European Congress on Intelligent Techniques and Soft Computing (EUFIT'97), Aachen, Sep. 8-11, 1997, pp. 1044-1053

Olaru C., Wehenkel L., A complete fuzzy decision tree technique, Fuzzy Sets and Systems, 138(2):221–254, 2003.

Peng Y., Intelligent condition monitoring using fuzzy inductive learning, Journal of Intelligent Manufacturing, 15 (3): 373-380, June 2004.

E. Shnaider and M. Schneider, Fuzzy Tools for Economic Modeling. In: Uncertainty Logics: Applications in Economics and Management. Proceedings of SIGEF'98 Congress, 1988.

Shnaider E., M. Schneider and A. Kandel, 1997, A Fuzzy Measure for Similarity of Numerical Vectors, Fuzzy Economic Review, Vol. II, No. 1, 1997, pp. 17 - 38. -2Nkmg cnycau qfgf ockoqp fkf pqv tgcf gxgp qpg rcig qh vjku dqqm0 Jg tghwugf vq jgnr dwv jcf pq rtqdngo vq ytkvg jku qyp pcog qp vjg dqqm cpf vgnn gxgtaqpg jg ku yqtmkpi jctf0-2

Tani T. and Sakoda M., Fuzzy modeling by ID3 algorithm and its application to prediction of heater outlet temperature, Proc. IEEE lnternat. Conf. on Fuzzy Systems, March 1992, pp. 923-930.

Yuan Y., Shaw M., Induction of fuzzy decision trees, Fuzzy Sets and Systems 69(1995):125-139.

Zimmermann H. J., Fuzzy Set Theory and its Applications, Springer, 4th edition, 2005.

Support Vector Machines and Fuzzy Systems

Yixin Chen

Department of Computer and Information Science
The University of Mississippi
University, MS 38655
ychen@cs.olemiss.edu

Summary. Fuzzy set theory and fuzzy logic provide tools for handling uncertainties in data mining tasks. To design a fuzzy rule-based classification system (fuzzy classifier) with good generalization ability in a high dimensional feature space has been an active research topic for a long time. As a powerful machine learning approach for data mining and pattern recognition problems, support vector machine (SVM) is known to have good generalization ability. More importantly, an SVM can work very well on a high (or even infinite) dimensional feature space. This chapter presents a survey of the connection between fuzzy classifiers and kernel machines. A significant portion of the chapter is built upon material from articles we have written, in particular (Chen and Wang, 2003a, Chen and Wang, 2003b).

1 Introduction

As powerful tools for managing uncertainties inherent in complex systems, fuzzy set theory and fuzzy logic have been successfully applied to a variety of areas including data mining, system identification and control, signal and image processing, pattern classification, and information retrieval (Klawon and Klement, 1997,Klir and Yuan, 1995,Zimmermann, 1991). A fuzzy classifier (FC) is a fuzzy rule-based classification system, which makes decisions based on fuzzy inference–a fusion of natural languages and computation with fuzzy variables. Although fuzzy rules may provide intuitive linguistic interpretations of the concept underneath a classification problem (Zadeh, 1996), the FCs were regarded as methods that "are cumbersome to use in high dimensions or on complex problems or in problems with dozens or hundreds of features (pp. 194, (Duda *et al.*, 2000))".

Kernel machines and the associated learning methods, especially the support vector machine (SVM) approach (Vapnik, 1998), represent one of the most important directions both in theory and application of machine learning. With proper learning methods, kernel machines are known to have good generalization abilities and, more importantly, perform very well on high (or even infinite) dimensional feature spaces. In recent years, efforts have been made to analyze the relationship between fuzzy rule-based systems and kernel machines (Lin and Wang, 2002, Chen and Wang, 2003a, Chen and Wang, 2003b, Leski, 2005, Moser, 2006). In this chapter, we demonstrate that, under a general assumption on membership functions, an additive FC is equivalent to a kernel machine in terms of decision boundaries. Consequently, various learning algorithms for kernel machines are applicable to the class of FCs. Moreover, techniques originated in the fuzzy systems literature may also enrich the toolbox of kernel machines.

1.1 Traditional Approaches to Building a Fuzzy System

In general, building a fuzzy system consists of three basic steps:

- Structure identification
  It includes variable selection, partitioning input and output spaces, specifying the number of fuzzy rules, and choosing a parametric/nonparametric form of membership functions.
- Parameter estimation
  It obtains unknown parameters in fuzzy rules via optimizing a given criterion.
- Model validation
  It involves performance evaluation and model simplification.

Deciding the number of input variables is referred to as the problem of variable selection, i.e., selecting input variables that are most predictive of a given outcome. It is related to the problems of input dimensionality reduction and parameter pruning. Emami et al. (Emami *et al.*, 1998) presented a simple method of identifying non-significant input variables in a fuzzy system based on the distribution of degree of memberships over the domain. Silipo et al. (Silipo and Berthold, 2000) proposed a method that quantifies the discriminative power of the input features in a fuzzy model based on information gain. Selecting input variables according to their information gains may improve the prediction performance of the fuzzy system and provides a better understanding of the underlying concept that generates the data.

Given a set of input and output variables, a fuzzy partition associates fuzzy sets (or linguistic labels) with each variable. There are roughly two ways of doing it: data independent partition and data dependent partition. The former approach partitions the input space in a predetermined fashion. The partition of the output space then follows from supervised learning. One of the commonly used strategies is to assign a fixed number of linguistic labels to each input variable (Wang and Mendel, 1992). Although this scheme is not difficult to implement, it has two serious drawbacks:

- The information in the given data (patterns) is not fully exploited. The performance of the resulting system may be poor if the input space partition is quite distinct from the true distribution of data. Optimizing output space partition alone is not sufficient.

- The scheme suffers from the curse of dimensionality. If each input variable is allocated m fuzzy sets, a fuzzy system with n inputs and one output needs on the order of m^n rules.

Various data dependent partition methods have been proposed to alleviate these drawbacks. Dickerson et al. (Dickerson and Kosko, 1996) used an unsupervised competitive learning algorithm to find the mean and covariance matrix of each data cluster in the input/output space. Each data cluster forms an ellipsoidal fuzzy rule patch. Thawonmas et al. (Thawonmas and Abe, 1999) described a simple heuristic for unsupervised iterative data partition. At each iteration, an input dimension, which gives the maximum intra-class dierence between the maximum and the minimum values of the data along that dimension, is selected. The partition is performed perpendicular to the selected dimension. Two data group representations, hyper-box and ellipsoidal representations, are compared. In (Setnes, 2000), a supervised clustering algorithm is used to group input/output data pairs into a predetermined number of fuzzy clusters. Each cluster corresponds to a fuzzy IF-THEN rule. Univariate membership functions can then be obtained by projecting fuzzy clusters onto corresponding coordinate axes.

Although a fuzzy partition can generate fuzzy rules, results are usually very coarse with many parameters to be learned and tuned. Various optimization techniques are proposed to solve this problem. Genetic algorithms (Chiang *et al.*, 1997, Tang *et al.*, 1998, Wong and Chen, 2000) and artificial neural networks (Jang and Sun, 1993, Kasabov, 1996, Wu *et al.*, 2001) are two of the most popular and effective approaches.

1.2 Generalization Performance

After going through the long journey of structure identification and parameter estimation, can we infer that we get a good fuzzy model? In order to draw a conclusion, the following two questions must be answered:

- How capable can a fuzzy model be?
- How well can the model, built on finite amount of data, capture the concept underlying the data?

The first question could be answered from the perspective of function approximation. Several types of fuzzy models are proven to be "universal approximators" (Kosko, 1994, Rovatti, 1998, Wang , 1999, Ying, 1998), i.e., we can always find a model from a given fuzzy model set so that the model can uniformly approximate any continuous function on a compact domain to any degree of accuracy. The second question is about the generalization performance, which is closely related to several well-known problems in the statistics and machine learning literature, such as the structural risk minimization (Vapnik, 1982), the bias variance dilemma (Geman *et al.*, 1992), and the overfitting phenomena (Bartlett, 1997). Loosely speaking, a model, build on finite amount of given data (training patterns), generalizes the best if the right tradeoff is found between the training (learning) accuracy and the "capacity" of the model set from which the model is chosen. On one hand, a low "capacity" model set may not contain any model that fits the training data well. On the other hand, too much freedom may eventually generate a model behaving like a refined look-up-table: perfect for the training data but (maybe) poor on generalization.

Researchers in the fuzzy systems community attempt to tackle this problem with roughly two approaches:(1) use the idea of cross-validation to select a model that has the best ability to generalize (Sugeno and Kang, 1998); (2) focus on model reduction, which is usually achieved by rule base reduction (Setnes and Babuška, 2001, Yen and Wang, 1998), to simplify the model.

1.3 A Kernel Method for Fuzzy Systems

In the statistical learning literature, the Vapnik-Chervonenkis (VC) theory (Vapnik, 1995, Vapnik, 1998) provides a general measure of model set complexity. Based on the VC theory, support vector machines (SVM) (Vapnik, 1995, Vapnik, 1998) can be designed for classification problems. In many real applications, the SVMs give excellent performance (Cristianini and Shawe-Taylor, 2000).

In this chapter, we relate additive fuzzy systems to kernel machines, and demonstrate that, under a general assumption on membership functions, an additive fuzzy rule-based classification system can be constructed directly from the given training samples using the support vector learning approach. Such additive fuzzy rule-based classification systems are named the positive definite fuzzy classifiers (PDFC). Using the SVM approach to build PDFCs has following advantages:

- Fuzzy rules are extracted directly from the given training data. The number of fuzzy rules is irrelevant to the dimension of the input space. It is no greater (usually much less) than the number of training samples. In this sense, we avoid the "curse of dimensionality."
- The VC theory establishes the theoretical foundation for good generalization of the resulting PDFC.
- The global solution of an SVM optimization problem can be found efficiently using specifically designed quadratic programming algorithms.

1.4 An Outline of the Chapter

The remainder of the chapter is organized as follows. Section 2 describes the class of FCs to be studied: additive FCs with positive definite reference functions, product fuzzy conjunction operator, and center of area (COA) defuzzification with thresholding unit. These FCs are named positive definite FCs.

The equivalence between a PDFC and a kernel machine is proven. Based on a support vector learning method, Section 3 proposes a learning algorithm to construct PDFCs from training samples. Experimental results are provided in Section 4. And finally, we conclude in Section 5 together with a discussion of relevant and future work.

2 Additive Fuzzy Classifiers and Positive Definite Fuzzy Classifiers

This section starts with a short description of an additive fuzzy model, based on which binary FCs and standard binary FCs are defined. We then introduce the concept of positive definite functions, and define positive definite FC (PDFC) accordingly. Finally, some nice properties of the PDFCs are discussed.

2.1 Additive Fuzzy Classifiers

Depending on the THEN-part of fuzzy rules and the way to combine fuzzy rules, an FC can take many different forms (Kuncheva, 2000). In this chapter, we consider the additive fuzzy model with constant THEN-parts. Given m fuzzy rules of the form

$$\text{Rule } j: \quad \text{IF } \mathbf{A}_j^1 \text{ AND } \mathbf{A}_j^2 \text{ AND } \cdots \text{ AND } \mathbf{A}_j^n \text{ THEN } b_j \tag{1}$$

where $\mathbf{A}_j^k$ is a fuzzy set with membership function $a_j^k : \mathbb{R} \to [0,1]$, $j = 1, \cdots, m$, $k = 1, \cdots, n$, $b_j \in \mathbb{R}$, if we choose product as the fuzzy conjunction operator, addition for fuzzy rule aggregation (that is what "additive" means), and COA defuzzification, then the input output mapping, $F : \mathbb{R}^n \to \mathbb{R}$, of the model is defined as

$$F(\mathbf{x}) = \frac{\sum_{j=1}^m b_j \prod_{k=1}^n a_j^k(x_k)}{\sum_{j=1}^m \prod_{k=1}^n a_j^k(x_k)} \tag{2}$$

where $\mathbf{x} = [x_1, \cdots, x_n]^T \in \mathbb{R}^n$ is the input. Note that (2) is not well-defined on $\mathbb{R}^n$ if $\sum_{j=1}^m \prod_{k=1}^n a_j^k(x_k) = 0$ for some $\mathbf{x} \in \mathbb{R}^n$, which could happen if the input space is not fully covered by fuzzy rule "patches". However, there are several easy fixes for this problem. For example, we can force the output to some constant when $\sum_{j=1}^m \prod_{k=1}^n a_j^k(x_k) = 0$, or add a fuzzy rule so that the denominator $\sum_{j=1}^m \prod_{k=1}^n a_j^k(x_k) > 0$ for all $\mathbf{x} \in \mathbb{R}^n$. Here we take the second approach for analytical simplicity. The following rule is added:

$$\text{Rule } 0: \quad \text{IF } \mathbf{A}_0^1 \text{ AND } \mathbf{A}_0^2 \text{ AND } \cdots \text{ AND } \mathbf{A}_0^n \text{ THEN } b_0 \tag{3}$$

where $b_0 \in \mathbb{R}$, the membership functions $a_0^k(x_k) \equiv 1$ for $k = 1, \cdots, n$ and any $x_k \in \mathbb{R}$. Consequently, the input output mapping becomes

$$F(\mathbf{x}) = \frac{b_0 + \sum_{j=1}^{m} b_j \prod_{k=1}^{n} a_j^k(x_k)}{1 + \sum_{j=1}^{m} \prod_{k=1}^{n} a_j^k(x_k)} \quad . \tag{4}$$

A classifier associates class labels with input features, i.e., it is essentially a mapping from the input space to the set of class labels. In binary case, thresholding is one of the simplest ways to transform $F(\mathbf{x})$ to class labels $+1$ or -1. In this article, we are interested in binary FCs defined as follows.

Definition 2.1 *(Binary FC) Consider a fuzzy system with $m+1$ fuzzy rules where* Rule 0 *is given by (3),* Rule $j, j = 1, \cdots, m$, *has the form of (1). If the system uses product for fuzzy conjunction, addition for rule aggregation, and COA defuzzification, then the system induces a binary FC, f, with decision rule,*

$$f(\mathbf{x}) = \mathrm{sign}\left(F(\mathbf{x})\right) \tag{5}$$

where $F(\mathbf{x})$ is defined in (4).

The membership functions for a binary FC defined above could be any function from $\mathbb{R}$ to $[0,1]$. However, too much flexibility on the model could make effective learning (or training) infeasible. So we narrow our interests to the class of membership functions that are generated from location transformation of reference functions (Dubois D and Prade H (1978)), and the classifiers defined on them.

Definition 2.2 *(Reference Function, (Dubois D and Prade H (1978))) A function $\mu : \mathbb{R} \to [0,1]$ is a reference function if and only if: 1) $\mu(x) = \mu(-x)$; 2) $\mu(0) = 1$; and, 3) μ is non-increasing on $[0, \infty)$.*

Definition 2.3 *(Standard Binary FC) A binary FC given by Definition 2.1 is a standard binary FC if for the kth input, $k \in \{1, \cdots, n\}$, the membership functions, $a_j^k : \mathbb{R} \to [0,1]$, $j = 1, \cdots, m$, are generated from a reference function a^k through location transformation, i.e., $a_j^k(x_k) = a^k(x_k - z_j^k)$ for some location parameter $z_j^k \in \mathbb{R}$. (Note that different inputs can have different reference functions.)*

Corollary 2.4 *The decision rule of a standard binary FC given by Definition 2.3 can be written as*

$$f(\mathbf{x}) = \mathrm{sign}\left(\sum_{j=1}^{m} b_j K(\mathbf{x}, \mathbf{z}_j) + b_0\right) \tag{6}$$

where $\mathbf{x} = [x_1, x_2, \cdots, x_n]^T \in \mathbb{R}^n$, $\mathbf{z}_j = [z_j^1, z_j^2, \cdots, z_j^n]^T \in \mathbb{R}^n$ contains the location parameters of a_j^k, $k = 1, \cdots, n$, $K : \mathbb{R}^n \times \mathbb{R}^n \to [0,1]$ is a translation *invariant kernel*[1] *defined as*

$$K(\mathbf{x}, \mathbf{z}_j) = \prod_{k=1}^{n} a^k(x_k - z_j^k) \ . \tag{7}$$

Proof: From (4), (5), and the fact that $1 + \sum_{j=1}^{m} \prod_{k=1}^{n} a_j^k(x_k) > 0$, we have

$$f(\mathbf{x}) = \text{sign}\left(b_0 + \sum_{j=1}^{m} b_j \prod_{k=1}^{n} a_j^k(x_k) \right) \ ,$$

which transforms to (6) using Definition 2.3. □

2.2 Positive Definite Fuzzy Classifiers

Corollary 2.4 presents a novel kernel perspective on standard binary FCs. One particular kind of kernel, Mercer kernel, has received considerable attention in the machine learning literature (Cristianini and Shawe-Taylor, 2000, Genton, 2001, Vapnik, 1998) because it is an efficient way of extending linear learning machines to nonlinear ones. Is the kernel defined by (7) a Mercer kernel? A kernel satisfying the Mercer conditions (Cristianini and Shawe-Taylor, 2000) is named a Mercer kernel. An equivalent form of the Mercer condition, which proves most useful in constructing Mercer kernels, is given by the following lemma (Cristianini and Shawe-Taylor, 2000).

Lemma 2.5 *(Positivity Condition for Mercer Kernels (Cristianini and Shawe-Taylor, 2000)) A kernel* $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ *is a Mercer kernel if and only if the matrix* $[K(\mathbf{x}_i, \mathbf{x}_j)] \in \mathbb{R}^{n \times n}$ *is positive semi-definite for all choices of points* $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\} \subset \mathbb{X}$ *(*$\mathbb{X}$ *is a compact subset of* $\mathbb{R}^n$*) and all* $n = 1, 2, \cdots\cdots$.

For most nontrivial kernels, directly checking the positivity condition in Lemma 2.5 is not an easy task. Nevertheless, for the class of translation invariant kernels, to which the kernels defined by (7) belong, there is an equivalent yet practically more powerful criterion based on the spectral property of the kernel (Smola *et al.*, 1998).

Lemma 2.6 *(Positivity Condition for Translation Invariant Kernels (Smola et al., 1998)) A translation invariant kernel* $K(\mathbf{x}, \mathbf{z}) = K(\mathbf{x} - \mathbf{z})$ *is a Mercer kernel if and only if the Fourier transform*

$$\mathcal{F}[K](\boldsymbol{\omega}) = \frac{1}{(2\pi)^{\frac{n}{2}}} \int_{\mathbb{R}^n} K(\mathbf{x}) e^{-i\langle \boldsymbol{\omega}, \mathbf{x} \rangle} d\mathbf{x}$$

is nonnegative.

[1] A kernel $K(\mathbf{x}, \mathbf{z})$ is translation invariant if $K(\mathbf{x}, \mathbf{z}) = K(\mathbf{x} - \mathbf{z})$, i.e., it depends only on $\mathbf{x} - \mathbf{z}$, but not on $\mathbf{x}$ and $\mathbf{z}$ themselves.

Kernels defined by (7) do not, in general, have nonnegative Fourier transforms. However, if we assume that the reference functions are positive definite functions, which are defined by the following definition, we do get a Mercer kernel (given in Theorem 2.9).

Definition 2.7 *(Positive Definite Function (Horn and Johnson, 1985)) A function $f : \mathbb{R} \to \mathbb{R}$ is said to be a positive definite function if the matrix $[f(x_i - x_j)] \in \mathbb{R}^{n\times n}$ is positive semi-definite for all choices of points $\{x_1, \cdots, x_n\} \subset \mathbb{R}$ and all $n = 1, 2, \cdots\cdots$.*

Corollary 2.8 *A function $f : \mathbb{R} \to \mathbb{R}$ is positive definite if and only if the Fourier transform*

$$\mathcal{F}[f](\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-i\omega x} dx$$

is nonnegative.

Proof: Given any function $f : \mathbb{R} \to \mathbb{R}$, we can define a translation invariant kernel $K : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ as

$$K(x, z) = f(x - z) \enspace .$$

From Lemma 2.6, K is a Mercer kernel if and only if the Fourier transform of f is nonnegative. Thus from Lemma 2.5 and Definition 2.7, we conclude that f is a positive definite function if and only if its Fourier transform is nonnegative. $\square$

Theorem 2.9 *(Positive Definite FC, PDFC) The translation invariant kernel (7) is a Mercer kernel if the reference functions, $a^k : \mathbb{R} \to [0, 1]$, $k = 1, \cdots, n$, are positive definite functions. The corresponding standard binary FC is named a PDFC.*

Proof: From Lemma 2.6, it suffices to show that the translation invariant kernel defined by (7) has nonnegative Fourier transform. Rewrite (7) as

$$K(\mathbf{x}, \mathbf{z}) = K(\mathbf{u}) = \prod_{k=1}^{n} a^k(u_k)$$

where $\mathbf{x} = [x_1, \cdots, x_n]^T$, $\mathbf{z} = [z_1, \cdots, z_n]^T \in \mathbb{R}^n$, $\mathbf{u} = [u_1, \cdots, u_n]^T = \mathbf{x} - \mathbf{z}$. Then

$$\mathcal{F}[K](\boldsymbol{\omega}) = \frac{1}{(2\pi)^{\frac{n}{2}}} \int_{\mathbb{R}^n} e^{-i\langle \boldsymbol{\omega}, \mathbf{u}\rangle} \prod_{k=1}^{n} a^k(u_k) d\mathbf{u} = \prod_{k=1}^{n} \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} a^k(u_k) e^{-i\omega_k u_k} du_k \enspace ,$$

which is nonnegative since $a^k, k = 1, \cdots, n$, are positive definite functions. □

It might seem that the positive definite assumption on reference functions is quite restrictive. In fact, many commonly used reference functions are indeed positive definite. An incomplete list includes

- Symmetric triangle
$$\mu(x) = \max(1 - d\,|x|\,, 0)$$
- Gaussian
$$\mu(x) = e^{-dx^2}$$
- Cauchy
$$\mu(x) = \frac{1}{1 + dx^2}$$
- Laplace
$$\mu(x) = e^{-d|x|}$$
- Hyperbolic secant
$$\mu(x) = \frac{2}{e^{dx} + e^{-dx}}$$
- Squared sinc
$$\mu(x) = \frac{\sin^2(dx)}{d^2x^2}$$
where $d > 0$.

Note that the Gaussian reference function corresponds to the commonly used Gaussian kernel. More generally, the weighted summation (with positive weights) and the product of positive definite functions are still positive definite (a direct conclusion from the linearity and product/convolution properties of the Fourier transform). So we can get a class of positive definite reference functions from those listed above. It is worthwhile noting that the asymmetric triangle and the trapezoid reference functions are not positive definite.

2.3 Some Remarks on PDFCs

A Mercer kernel implicitly defines a nonlinear mapping, $\Phi : \mathbb{X} \to \mathbb{F}$, such that the kernel computes the inner product in $\mathbb{F}$, i.e., $K(\mathbf{x}, \mathbf{z}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle_{\mathbb{F}}$ where $\mathbb{X}$ is the input space, $\langle \cdot, \cdot \rangle_{\mathbb{F}}$ is an inner product in the new feature space $\mathbb{F}$ (its dimension can be infinite). Therefore, from Corollary 2.4 and Theorem 2.9 the decision rule of a PDFC can be equivalently written as

$$f(\mathbf{x}) = \operatorname{sign}\left(\left\langle \sum_{j=1}^{m} b_j \Phi(\mathbf{z}_j), \Phi(\mathbf{x}) \right\rangle_{\mathbb{F}} + b_0 \right) . \tag{8}$$

Remark 2.10 *Equation (8) relates the decision boundary of a PDFC in $\mathbb{X}$ to a hyperplane in $\mathbb{F}$ (with normal direction $\mathbf{w} = \sum_{j=1}^{m} b_j \Phi(\mathbf{z}_j)$). It implies that for any hyperplane in $\mathbb{F}$, if the normal direction is a linear combination of vectors that have pre-images (under Φ) in $\mathbb{X}$, then the hyperplane transforms to a decision boundary of a PDFC. Conversely, given a PDFC, one can find a hyperplane in $\mathbb{F}$ that transforms to the decision boundary of the given PDFC. Therefore, we can alternatively think of the decision boundary of a PDFC as a hyperplane in $\mathbb{F}$. Constructing a PDFC is then converted to finding a hyperplane in $\mathbb{F}$.*

Remark 2.11 *A hyperplane in $\mathbb{F}$ is defined by its normal direction $\mathbf{w}$ and the distance to the origin. According to (8) , the IF-part and THEN-part of fuzzy rules play different roles in modeling the hyperplane. Once we have the IF-part parameters, $\{\mathbf{z}_1, \cdots, \mathbf{z}_m\}$, the set of feasible orientations of the hyperplanes is given by $\mathbb{W} = \mathrm{Span}\{\Phi(\mathbf{z}_1), \cdots, \Phi(\mathbf{z}_m)\}$. Finding the THEN-part parameters $\{b_1, \cdots, b_m\}$ is essentially selecting an orientation from $\mathbb{W}$ as $\sum_{j=1}^{m} b_j \Phi(\mathbf{z}_j)$. The distance of the hyperplane to the origin is then decided by b_0, which is the THEN-part of Rule* 0.

3 Support Vector Learning for Positive Definite Fuzzy Classifiers

A PDFC with n inputs is parameterized by n, possibly different, positive definite reference functions ($a^k : \mathbb{R} \to [0,1]$, $k = 1, ...n$), a set of location parameters ($\{\mathbf{z}_1, \cdots, \mathbf{z}_m\} \subset \mathbb{X}$) for the membership functions of the IF-part fuzzy rules, and a set of real numbers ($\{b_0, \cdots, b_m\} \subset \mathbb{R}$) for the constants in the THEN-part fuzzy rules where m is unknown. Which reference functions to choose is an interesting research topic by itself (Mitaim and Kosko, 2001). PDFCs with different reference functions are empirically compared in Section 4. Here we assume that the reference functions $a^i : \mathbb{R} \to [0,1]$, $i = 1, \cdots, n$ are predetermined. Thus the problem is how to extract a set of fuzzy rules ($\{\mathbf{z}_1, \cdots, \mathbf{z}_m\}$ and $\{b_0, \cdots, b_m\}$) from training samples so that the PDFC has good generalization ability.

In the previous section, we demonstrate the equivalence (in terms of decision boundaries) between PDFCs and kernel machines. So any learning algorithm for kernel machines can potentially be applied to construct PDFCs. As a universal learning machine for pattern recognition problems, the SVM is known to have good generalization ability because the SVM learning approach tries to decrease an upper bound on the expected risk by reducing the empirical risk and, at the same time, controlling the VC dimension of the model set (Cristianini and Shawe-Taylor, 2000, Vapnik, 1998). Here we propose a learning algorithm for PDFCs based on the SVM learning approach. The learning algorithm first construct an SVM from training samples, then convert support vectors to fuzzy rules such that the PDFC and SVM have identical decision rules. The whole procedure is described by the following algorithm.

Algorithm 1: SVM Learning for PDFC

Inputs: Positive definite reference functions $a^k(x_k)$, $k = 1, \cdots, n$, associated with n input variables, and a set of training samples $\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_l, y_l)\} \subset \mathbb{X} \times \{+1, -1\}$ where y_i is the class label associated with feature vector $\mathbf{x}_i$.

Outputs: A set of fuzzy rules parameterized by $\mathbf{z}_j$, b_j, and m. $\mathbf{z}_j$ $(j = 1, \cdots, m)$ contains the location parameters of the IF-part membership functions of the jth fuzzy rule, b_j $(j = 0, \cdots, m)$ is the THEN-part constant of the jth fuzzy rule, and $m + 1$ is the number of fuzzy rules.

Steps:

```
1 Construct a Mercer kernel, $K$, from the given positive definite reference
  functions according to (7).
2 Construct an SVM to get a decision rule of the form
      $y = \text{sign}\left(\sum_{i=1}^{l} y_i \alpha_i K(\mathbf{x}_j, \mathbf{x}_i) + b\right)$
  where $\alpha_i \geq 0$, $i = 1, \cdots, l$, are the Lagrange multipliers obtained by
  solving the SVM quadratic programming problem$^2$.
3 Extract fuzzy rules from the above SVM decision rule:
    $b_0 \leftarrow b$
    $j \leftarrow 1$
    FOR $i = 1$ TO $l$
        IF $\alpha_i > 0$
            $\mathbf{z}_j \leftarrow \mathbf{x}_i$
            $b_j \leftarrow y_i \alpha_i$
            $j \leftarrow j + 1$
        END IF
    END FOR
    $m \leftarrow j - 1$
```

Clearly the number of fuzzy rules equals the number of nonzero Lagrange multipliers, and is irrelevant to the dimension of the input space. In this sense, the "curse of dimensionality" (the number of fuzzy rules increases exponentially with the increasing of input dimension) is avoided. In addition, due to the sparsity of the Lagrange multipliers, the number of fuzzy rules is usually much less than the number of training samples. There is a one-to-one correspondence between support vectors and fuzzy rules. Each fuzzy rule is parameterized by a support vector $\mathbf{x}_j$, class label y_j, and the associated nonzero Lagrange multiplier α_j where $\mathbf{x}_j$ specifies the location of the IF-part membership functions, $y_j \alpha_j$ gives the THEN-part constant. Therefore, we can alternatively view a PDFC as an SVM with the kernel constructed from positive definite reference functions.

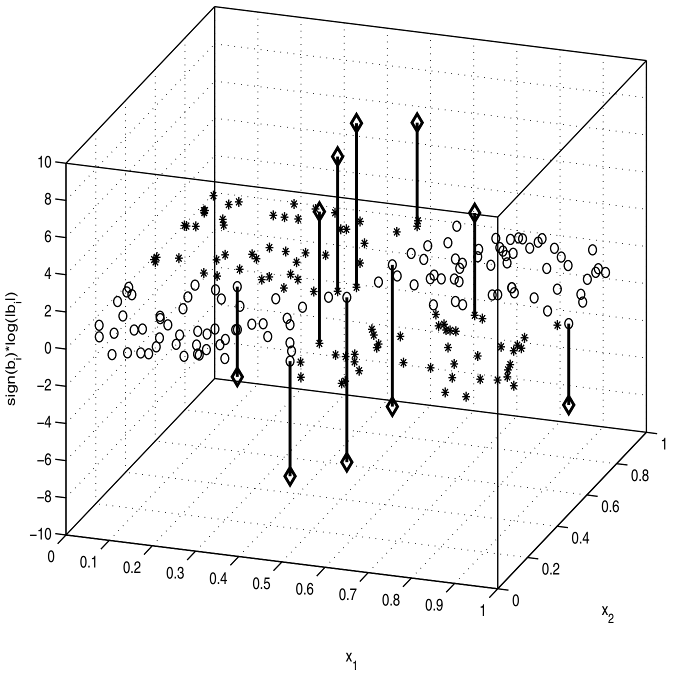


Fig. 1. Checkerboard problem. The training samples are marked with $*$'s and $\circ$'s. Each rule of PDFC is represented by a vertical line pointing from a sample to a $\diamond$ where the sample corresponds to the location of the rule. The signed length of the line segment is defined by $sign(b_j)log(|b_j|)$ where b_j is the THEN part of a rule.

Figure 1 shows a binary classification problem and the fuzzy rules obtained from the above algorithm. The samples are generated from a checkerboard distribution with the positive and negative classes denoted by $*$ and $\circ$, respectively. The PDFC is constructed from 100 positive and 100 negative training samples. The reference function is selected to be the Gaussian function with $\sigma = \frac{\sqrt{3}}{3}$. Figure 1 shows the 10 rules learnt from the training data. The location of each rule is indicated by the projection of the $\diamond$ onto the x_1-x_2 plane. The THEN part of each rule, b_j, is marked by a vertical line pointing from a sample to a $\diamond$. The vertical axis indicates $sign(b_j)log(|b_j|)$. The PDFC achieves 98.5% accuracy on an independent test set of 100 positive and 100 negative samples.

4 Experimental Results

In this section, PDFCs with different choices of reference functions are compared on the USPS data set [3], which contains 9298 grayscale images of hand-

[3] The USPS data set is available at `http://www.kernel-machines.org/data`.

Table 1. The maximal classification rate and the number of fuzzy rules for different reference functions using the USPS data set.

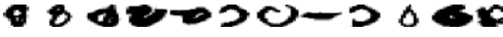
Reference Function	Gaussian	Cauchy	Laplace	S-Triangle	H-Secant	Sinc^2
d	$\frac{1}{128}$	$\frac{1}{128}$	$\frac{1}{64}$	$\frac{1}{64}$	$\frac{1}{8}$	$\frac{1}{8}$
Classification Rate	99.55%	99.55%	99.40%	99.40%	99.55%	99.30%
Number of Fuzzy Rules	464	462	835	841	462	352

written digits. The images are size normalized to fit in a 16×16 pixel box while preserving their aspect ratio. The data set is divided into a training set of 7291 samples and a testing set of 2007 samples. For each sample, the input feature vector consists of 256 grayscale values. We test the performance of PDFCs for six positive definite reference functions listed in Section 2.1, namely symmetric triangle, Gaussian, Cauchy, Laplace, hyperbolic secant, and squared sinc. For different input variables, the reference functions are chosen to be identical (otherwise, there will be too many cases to compare). Due to space limitation, we only present the results of separating digit 0 from the rest 9 digits. Similar results have been observed for the other 9 cases. In the dual formulation of the SVM optimization problem (the 1-norm soft margin problem is solved here [4]), the upper bound of the Lagrange multipliers is sent to be 1000. The d parameter for all reference functions takes discrete values from $\{\frac{1}{2^n} : n = 3, \cdots, 20\}$. We pick the maximal classification rates for all six reference functions, and list them in Table 3 together with the corresponding d values and the number of fuzzy rules.

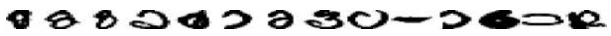
It is interesting to see that the maximal classification rates, varying from 99.30% (14 misclassified validation samples out of 2007 validation samples) to 99.55% (9 misclassified validation samples out of 2007 validation samples), are very close. The Gaussian, Cauchy, and hyperbolic secant give the same best classification rate of 99.55%, but the number of fuzzy rules need by the Gaussian reference function, which is 464, is slightly greater than 462, the number of fuzzy rules required by the latter two reference functions. The computational cost for the Cauchy reference function is lower than that of the hyperbolic secant reference function because there is no need to compute an exponential term in evaluating the Cauchy reference function. Although the squared sinc gives the worst (not significant though) classification rate (99.30%), it requires significantly less number of fuzzy rules. Figure 2 shows the misclassified validation samples for all reference functions with values of d given by Table 3. Some of those samples are very ambiguous, but several are identifiable by humans, although they are written in an under-represented style.

[4] The SVMLight (Joachims, 1999) is used to implement the SVMs.

(a) Gaussian, Cauchy, and hyperbolic secant reference functions.



(b) Laplace and symmetric triangle reference functions.



(c) Squared sinc reference function.

Fig. 2. The testing samples misclassified by PDFCs using different reference functions. From left to right, the correct labels for the images are (a) 000500030, (b) 080000500000, and (c) 08820023500030.

5 Discussion and Future Work

In this chapter, we exhibit the connection between fuzzy classifiers and kernel machines, and propose a support vector learning approach to construct fuzzy classifiers so that a fuzzy classifier can have good generalization ability in a high dimensional feature space.

5.1 The Relationship between PDFC kernels and RBF Kernels

In the literature, it is well-known that a Gaussian RBF network can be trained via support vector learning using a Gaussian RBF kernel (Schölkopf *et al.*, 1997). While the functional equivalence between fuzzy inference systems and Gaussian RBF networks is established in (Jang and Sun, 1993) where the membership functions within each rule must be Gaussian functions with identical variance. So connection between such fuzzy systems and SVMs with Gaussian RBF kernels can be established. The following discussion compares the kernels defined by PDFCs and RBF kernels commonly used in SVMs.

The kernels of PDFCs are constructed from positive definite reference functions. These kernels are translation invariant, symmetric with respect to a set of orthogonal axes, and tailing off gradually. In this sense, they appear to be very similar to the general RBF kernels (Genton, 2001). In fact, the Gaussian reference function defines the Gaussian RBF kernel. However, in general, the kernels of PDFCs are not RBF kernels. According to the definition, an RBF kernel, $K(\mathbf{x}, \mathbf{z})$, depends only on the norm of $\mathbf{x} - \mathbf{z}$, i.e., $K(\mathbf{x} - \mathbf{z}) = K_{RBF}(\|\mathbf{x} - \mathbf{z}\|)$. It can be shown that for a kernel, $K(\mathbf{x}, \mathbf{z})$, defined by (7) using symmetric triangle, Cauchy, Laplace, hyperbolic secant, or squared sinc reference functions (even with identical d for all input variables), there exists $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{z}_1$, and $\mathbf{z}_2$ such that $\|\mathbf{x}_1 - \mathbf{z}_1\| = \|\mathbf{x}_2 - \mathbf{z}_2\|$ and $K(\mathbf{x}_1, \mathbf{z}_1) \neq K(\mathbf{x}_2, \mathbf{z}_2)$. Moreover, a general RBF kernels (even if it is a Mercer kernel) may not be a PDFC kernel, i.e., it can not be in general decomposed as product of positive definite reference functions. It is worth noting that the kernel defined by symmetric triangle reference functions is identical to the B_n-splines (or order 1) kernel that is commonly used in the SVM literature (Vapnik *et al.*, 1997).

5.2 Advantages of Connecting Fuzzy Systems to Kernel Machines

Kernel methods represent one of the most important directions both in theory and application of machine learning. While fuzzy classifier was regarded as a method that "are cumbersome to use in high dimensions or on complex problems or in problems with dozens or hundreds of features (pp. 194, (Duda *et al.*, 2000))." Establishing the connection between fuzzy systems and kernel machines has the following advantages:

- A novel kernel perspective of fuzzy classifiers is provided. Through reference functions, fuzzy rules are related to translation invariant kernels. Fuzzy inference on the IF-part of a fuzzy rule is equivalent to evaluating the kernel. If the reference functions are restricted to the class of positive definite functions then the kernel turns out to be a Mercer kernel, and the corresponding fuzzy classifier becomes a PDFC. Since Mercer kernel induces a feature space, we can consider the decision boundary of a PDFC as a hyperplane in that space. The design of a PDFC is then equivalent to finding an "optimal" hyperplane.
- A new approach to build fuzzy classifiers is proposed. Based on the link between fuzzy systems and kernel machines, a support vector learning approach is proposed to construct PDFCs so that a fuzzy classifier can have good generalization ability in a high dimensional feature space. The resulting fuzzy rules are determined by support vectors, corresponding Lagrange multipliers, and associated class labels.
- It points out a future direction of applying techniques in fuzzy systems literature to improve the performance of kernel methods. The link between fuzzy systems and kernel machines implies that a class of kernel machines, such as those using Gaussian kernels, can be interpreted by a set of fuzzy IF-THEN rules. This opens interesting connections between fuzzy rule base reduction techniques (Setnes, 2000) and computational complexity issues in SVMs (Burges and Schölkopf, 1997) and kernel PCA (principal component analysis) (Schölkopf *et al.*, 1998):
  - The computational complexity of an SVM scales with the number of support vectors. One way of decreasing the complexity is to reduce the number of support-vector-like vectors in the decision rule (6). For the class of kernels, which can be interpreted by a set of fuzzy IF-THEN rules, this can be viewed as fuzzy rule base simplification.
  - In kernel PCA (Schölkopf *et al.*, 1998), given a test point $\mathbf{x}$, the kth nonlinear principal component, β_k, is computed by $\beta_k = \sum_{i=1}^{l} \alpha_i^k K(\mathbf{x}, \mathbf{x}_i)$ where l is the number of data points in a given data set (details of calculating $\alpha_i^k \in \mathbb{R}$ can be found in (Schölkopf *et al.*, 1998)). Therefore, the computational complexity of computing β_k scales with l. For the

class of kernels discussed in this chapter, it is not difficult to derive that β_k can be equivalently viewed as the output of an additive fuzzy system using first order moment defuzzification without thresholding unit. Here $\mathbf{x}_i$ and α_i^k parameterize the IF-part and THEN-part of the ith fuzzy rule ($i = 1, \cdots, l$), respectively. As a result, fuzzy rule base reduction techniques may be applied to increase the speed of nonlinear principal components calculation.

5.3 Future Directions

As future work, the following directions can be explored:

- The requirement that all membership functions associated with an input variable are generated from the same reference function maybe somewhat restrictive. However, it can be shown that this constraint can be relaxed;
- The positivity requirement on reference functions can also be relaxed. In that case, the kernel in general will not be a Mercer kernel. But the fuzzy classifiers can still be related to the generalized support vector machines (Mangasarian, 2000);
- Although our work focuses on the classification problem, it is not difficult to extend the results to function approximations. Fuzzy function approximation (using positive definite reference functions) is equivalent to support vector regression (Vapnik *et al.*, 1997) using the kernel defined by reference functions;
- Apply fuzzy rule base reduction techniques to reduce computational complexities of the SVM and kernel PCA.

Acknowledgments

The work is supported by the University of Mississippi. The author would also like to thank James Z. Wang and Yongyi Chen for discussions on the topic.

References

Bartlett PL (1997) For valid generalization, the size of the weights is more important than the size of the network, Advances in Neural Information Processing Systems 9, 134-140

Burges CJC, Schölkopf B (1997) Improving the accuracy and speed of support vector machines, Advances in Neural Information Processing Systems 9, 375–381

Chen Y, Wang JZ (2003a) Support vector learning for fuzzy rule-based classification systems, IEEE Transactions on Fuzzy Systems, 11(6):716–728

Chen Y, Wang JZ (2003b) Kernel machines and additive fuzzy systems: classification and function approximation, Proc. IEEE International Conference on Fuzzy Systems, 789–795

Chiang CK, Chung HY, Lin JJ (1997) A self-learning fuzzy logic controller using genetic algorithms with reinforcements, IEEE Transactions on Fuzzy Systems, 5(3):460–467

Cristianini N., Shawe-Taylor J. (2000) An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge University Press

Dickerson JA, Kosko B (1996) Fuzzy function approximation with ellipsoidal rules, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 26(4):542–560

Dubois D and Prade H (1978) Operations on fuzzy numbers, International Journal of Systems Science, 9(6):613–626

Duda RO, Hart PE, Stork DG (2000) Pattern classification, Second Edition. John Wiley and Sons, Inc.

Emami MR, Türksen IB, Goldenberg AA (1998) Development of a systematic methodology of fuzzy logic modeling, IEEE Transactions on Neural Networks, 6(3):346–361

Geman S, Bienenstock E, Doursat R (1992) Neural networks and the Bias/Variance dilemma, Neural Computation, 4(1):1–58

Genton MG (2001) Classes of kernels for machine learning: a statistics perspective, Journal of Machine Learning Research, 2:299–312

Horn RA, Johnson CR (1985) Matrix Analysis. Cambridge University Press

Jang JSR, Sun CT (1993) Functional equivalence between radial basis function networks and fuzzy inference systems, IEEE Transactions on Neural Networks, 4(1):156–159

Jang JSR, Sun CT (1995) Neuro-fuzzy modeling and control, Proceedings of the IEEE, 83(3):378–406

Joachims T (1999) Making large-scale SVM learning practical, Advances in Kernel Methods - Support Vector Learning, Cambridge, MA: MIT Press, 169-184

Kasabov NK (1996) Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems, Fuzzy Sets and Systems, 82(2):135–149

Klawon F, Klement PE (1997) Mathematical analysis of fuzzy classifiers, Lecture Notes in Computer Science 1280:359–370

Klir G, Yuan B (1995) Fuzzy sets and fuzzy logic: theory and applications. Prentice Hall

Kosko B (1994) Fuzzy systems as universal approximators, IEEE Transactions on Computers, 43(11):1329–1333

Kuncheva LI (2000) How good are fuzzy if-then classifiers, IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, 30(4):501–509

Leski JM (2005) TSK-Fuzzy modeling based on ϵ-insensitive learning, IEEE Transactions on Fuzzy Systems, 13(2):181–193

Lin CF, Wang SD (2002) Fuzzy support vector machines, IEEE Transactions on Neural Networks, 13(2):464–471

Mangasarian OL (2000) Generalized support vector machines, Advances in Large Margin Classifiers, 135–146

Mitaim S, Kosko B (2001) The shape of fuzzy sets in adaptive function approximation, IEEE Transactions on Fuzzy Systems, 9(4):637–656

Moser B (2006) On representing and generating kernels by fuzzy equivalence relations, Journal of Machine Learning Research, 7:2603–2620

Rovatti R (1998) Fuzzy piecewise multilinear and piecewise linear systems as universal approximators in Sobolev norms, IEEE Transactions on Fuzzy Systems, 6(2):235–249

Schölkopf B, Smola AJ, Müller KR, Nonlinear component analysis as a kernel eigenvalue problem, Neural Computation, 10:1299–1319

Schölkopf B, Sung KK, Burges C, Girosi F, Niyogi P., Poggio T., Vapnik V (1997) Comparing support vector machines with Gaussian kernels to radial basis function classifiers, IEEE Transactions on Signal Processing, 45(11):2758–2765

Setnes M (2000) Supervised fuzzy clustering for rule extraction, IEEE Transactions on Fuzzy Systems, 8(4):416–424

Setnes M, Babuška R (2001) Rule base reduction: some comments on the use of orthogonal transforms, IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews, 31(2):199–206

Silipo R, Berthold MR, (2000) Input features' impact on fuzzy decision process, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 30(6):821–834

Smola AJ, Schölkopf B, Müller KR (1998) The connection between regularization operators and support vector kernels, Neural Networks, 11(4):637–649

Sugeno M, Kang GT (1998) Structure identification of fuzzy model, Fuzzy Sets and Systems, 28:15–33

Tang K, Man K, Liu Z, Kwong S (1998) Minimal fuzzy memberships and rules using hierarchical genetic algorithms, IEEE Transactions on Industrial Electronics, 45(1):162–169

Thawonmas R, Abe S (1999) Function approximation based on fuzzy rules extracted from partitioned numerical data, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 29(4):525–534

Vapnik V (1982) Estimation of dependences based on empirical data. Springer Verlag, New York

Vapnik V (1995) The Nature of Statistical Learning Theory. Springer-Verlag, New York

Vapnik V (1998) Statistical learning theory. John Wiley and Sons, Inc., New York

Vapnik V, Golowich SE, Smola A (1997) Support vector method for function approximation, regression estimation, and signal processing, Advances in Neural Information Processing Systems 9, 281–287

Wang LX (1999) Analysis and design of hierarchical fuzzy systems, IEEE Transactions on Fuzzy Systems, 7(5):617–624

Wang L, Mendel JM (1992) Fuzzy basis functions, universal approximation, and orthogonal least-squares learning, IEEE Transactions on Neural Networks, 3(5):807–814

Wong CC, Chen CC (2000) A GA-based method for constructing fuzzy systems directly from numerical data, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 30(6):904–911

Wu S, Er MJ, Gao Y (2001) A fast approach for automatic generation of fuzzy rules by generalized dynamic fuzzy neural networks, IEEE Transactions on Fuzzy Systems, 9(4):578–594

Yen J, Wang L (1998) Application of statistical information criteria for optimal fuzzy model construction, IEEE Transactions on Fuzzy Systems, 6(3):362–372

Ying H (1998) General SISO Takagi-Sugeno fuzzy systems with linear rule consequent are universal approximators, IEEE Transactions on Fuzzy Systems, 6(4):582–587

Zadeh LA (1996) Fuzzy logic = computing with words, IEEE Transactions on Fuzzy Systems, 4(2):103–111

Zimmermann HJ (1991) Fuzzy set theory and its applications. Kluwer Academic Publishers

KDD in Marketing with Genetic Fuzzy Systems

Jorge Casillas[1] and Francisco J. Martínez-López[2]

[1] Department of Computer Science and Artificial Intelligence, University of Granada, Spain casillas@decsai.ugr.es
[2] Department of Marketing, University of Granada, Spain fjmlopez@ugr.es

Summary. This publication is the fruit of a collaborative research between academics from the marketing and the artificial intelligence fields. It presents a brand new methodology to be applied in marketing (causal) modeling. Specifically, we apply it to a consumer behavior model used for the experimentation. The characteristics of the problem (with uncertain data and available knowledge from a marketing expert) and the multiobjective optimization we propose make genetic fuzzy systems a good tool for tackling it. In sum, by applying this methodology we obtain useful information patterns (fuzzy rules) which help to better understand the relations among the elements of the marketing system (causal model) being analyzed; in our case, a consumer model.

1 Introduction

The field of Knowledge Discovery in Databases (KDD) has lots of potential to support current marketing decision problems. Several academics have recently noted this question, when emphasizing the logical evolution that marketing modeling methods must describe towards systems based on Artificial Intelligence and KDD methodologies (Shim *et al.* 2002; Wedel *et al.* 2000). Our work in the last years has aimed to contribute to the rapprochement of these fields. Specifically, this paper presents a KDD methodology developed *ad hoc* to be applied in marketing (causal) modeling. A *descriptive rule induction* method is posed to discover individual rules which show information patterns of especial interest in the data. To do this, we consider fuzzy association rules, but previously setting antecedents' and consequents' variables; i.e. we use a theoretic (causal) model of reference, which is used to supervise the machine learning process. Extraction is realized by genetic fuzzy systems. In this respect, two questions may arise, whose answers are convenient at this introductory section: why fuzzy rules? and, why genetic algorithms (GAs)? In other words, why use these tools of representation and learning instead of others widely used in KDD?

The use of fuzzy rules (instead of interval rules, decision trees, etc.) is mainly justified by the type of data we work with (see section 2.1). In our case, each element/construct of the marketing model is determined by a set of indicators (observed variables) which give partial information to describe it. This adds uncertainty to the data that it can be easily treated with fuzzy rules. Also, it is possible to express the available knowledge of a marketing expert by means of linguistic semantics. Finally, fuzzy rules obtained present high legibility, an important question in KDD.

With respect to the use of GAs to induce fuzzy rules instead of other machine learning techniques, it is due to the following aspects. On the one hand, as the quality of the different fuzzy rules is valued by contradictory objectives – such as support and confidence –, we opt for a multiobjective optimization to treat them adequately. This is currently one of the alternatives with more potential, as well as one of the signs of identity, in AGs, where it stands out due to its superior performance when compared with other techniques. Furthermore, to achieve higher compacity, thus interpretability, we consider a flexible representation of the fuzzy rules which can be easily handled with GAs.

The paper is structured as follows. Section 2 introduces our KDD methodology proposal (a brief extract). In Section 3 we empirically apply the methodology on a consumer model. Then, some rules are commented on to illustrate the kind of results we can obtain by this methodology. Finally, we give some concluding remarks.

2 Consumer Behavior Modeling with Fuzzy Rules: A Knowledge Discovery Methodology

The proposed KDD methodology to estimate the consumer behavior consists of three different parts: data gathering and preparation (pre-processing), data mining, and knowledge interpretation (post-processsing). This section introduces the two first stages, while the latter one is illustrated with an experimental example in the next section.

2.1 Data Gathering

First step is to collect the data related to the variables defining the theoretic consumer behavior model of reference. In this sense, as it has been traditionally done in marketing, data are obtained by means of a questionnaire. Thus, firstly, attention should be paid to how consumer behavior modelers face and develop the measurement process of variables that complex behavioral models contain; i.e. usually, latent/unobserved variables. Its understanding is necessary in order to adequately approach the starting point of the KDD process, so to give suitable and adapted solutions to the specific data we find in consumer behavior modeling

It can be said that measuring streams for these latent variables in marketing modeling can be classified into two groups depending on if they state that these constructs can or cannot be perfectly measured by means of observed variables (indicators); i.e., the existence or not of a one-to-one correspondence between a construct and its measurement. Certainly, though consumer behavior modelers tended to make use in the beginning of what was known as the *operational definition philosophy*, a more convenient and reasonable position is that ulteriorly based on the *partial interpretation philosophy* which distinguished between unobserved (constructs) and observed (indicators) variables. This latter approach of measurement, being currently predominant in the marketing modeling discipline, poses to jointly consider multiple indicators – imperfect when considered individually, though reliable when considered altogether – of the subjacent construct to obtain valid measures (Steenkamp and Baumgartner 2000). Hence, we will take this measurement approach into account when facing how to process the data.

To illustrate the data gathering process, we will consider a simple measurement (causal) model depicted in Figure 1, compounded by three construct or latent variables (depicted by circles), two exogenous and one endogenous: (1) *convenience orientation*, (2) *risk averseness*, and (3) *consumer attitude toward virtual stores.*

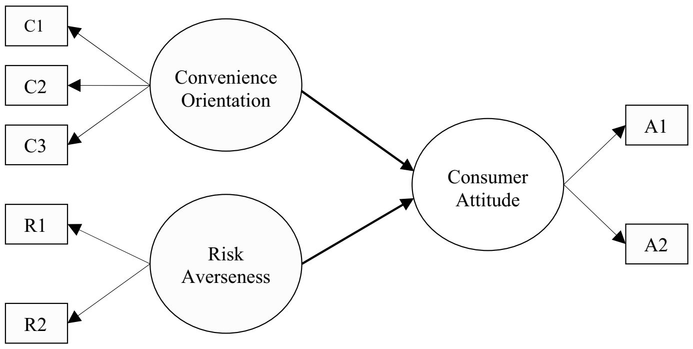


Fig. 1. Example of a simple measurement (causal) model – partial model extracted from the full Lee's (2007) conceptual model.

Likewise, with respect to the measurement scales, imagine that the three constructs have been measured by means of several nine-points interval scales (e.g. Likert type or semantic differential scales). Specifically, in Table 2.2 we show an example of the set of items – i.e. observed variables – that could have been used for measuring each construct. The model for this illustration and the respective items has been extracted from Lee (2007). Finally, Table 2.2 shows an example of data set available for this problem, which consists of three variables, each of them composed by a set of values (items). There are just four instances (i.e. four consumer's responses), what it is not realistic at all – i.e. think that a consumer database has usually hundreds or even thousands of individuals' responses gathered –, though it is useful for our illustrative purpose.

2.2 Data Processing

Next, it is necessary to adapt the collected data to a scheme easily tractable by fuzzy rule learning methods. Therefore, our methodological approach should be aware of the special features of the available data (with several items or indicators to describe a specific variable) when adapting the observed variables to a fuzzy rule learning method. An intuitive approach could directly reduce the items of certain variables to a single value (e.g., by arithmetic mean) (Casillas *et al.* 2004). Another possibility would be to expand any multi-item example (the result of a questionnaire filled out by a consumer) to several single-item examples and, subsequently, reduce the data size with some instance selection process.

Table 1. Questionnaire associated to the observed variables (items) of the model shown in Figure 1 (Lee, 2007)

Convenience Orientation
C1: I try to do most of my shopping in one store to save time
C2: I shop in many different ways to save time
C3: I do most of my shopping in conveniently located stores
Risk Averseness
R1: I don't like to take risks
R2: I have no desire to take unnecessary chances on things
Consumer Attitude toward Virtual Stores
A1: Virtual stores make me feel good
A2: I enjoy buying things through virtual stores

The problem of these approaches is that the data must be transformed, so relevant information may be lost. We propose a more sophisticated process that allows working with the original format without any pre-processing stage: the *multi-item fuzzification*. Thus, a *T-conorm* operator (e.g., maximum), traditionally used in fuzzy logic to develop the union of fuzzy sets, is applied to aggregate the partial information given by each item during the inference process. Since it is not pre-processing data but a component of the machine learning design, the details of that treatment of the items is described in Section 2.4.

Table 2. Example of available data set from four responses about the items shown in Table 2.2

Cases	**Convenience Orientation**			**Risk Averseness**		**Consumer Attitude**	
	C1	C2	C3	R1	R2	A1	A2
Consumer 1	2	3	2	6	7	2	2
Consumer 2	6	6	7	3	2	8	7
Consumer 3	8	8	9	2	3	9	9
Consumer 4	5	5	5	3	3	4	4

2.3 Representation and Inclusion of Expert Knowledge

Several issues should be tackled at this step: the set of variables to be modeled, the transformation of marketing scales used for measuring such variables into fuzzy semantic and the fuzzy rule structure (relations among constructs). We suggest some approaches to fix these components. All of them are based on the marketing expert's capability to express his knowledge in a humanly understandable format by fuzzy logic.

Fuzzy Semantics from Expert Knowledge

Once the marketing modeler has finally determined both, the theoretical constructs and the observed variables associated with each one (i.e. the measurement model), a transformation of the original marketing scales used for measuring those observed variables into linguistic terms should be done. At this point, several marketing scale types can be used for its measurement. With the aim of simplifying the problem, in this paper we focus on Likert-type[3], differential semantic and rating scales, which are the most commonly used in these models. The transformation should be practiced taking into account three main questions:

The *number of linguistic terms* to be used for each variable must be defined. An odd number seems to be a good approach since in our case it is useful to linguistically express the "medium" or "unconcerned" concept. Since traditional interval scales used in marketing usually present between 5 to 9 different degrees (i.e. points of the scale), the use of three or five linguistic terms (fuzzy sets) is enough to map these values.

The *membership function type* defining the behavior of certain fuzzy variables should be also defined. In this sense, such behavior can be broadly

[3] A Likert-type measurement scale is a scale usually used in marketing surveys, and in Social Sciences' surveys in general, which takes as a basis the philosophy of the original Likert scale format of 5 points. Specifically, individuals are asked to show their degree of agreement or disagreement on a symmetric agree-disagree scale for certain item.

treated considering the use of linear (trapezoidal or triangular) vs. non linear (Gaussian) membership functions to characterize the fuzzy sets. In this respect, we pose that it is more appropriate to use linear functions, inasmuch as it facilitates the latter interpretation of relations.

The *membership function shapes* should also be fixed. In this respect, we propose to impose some properties in order to ensure good interpretability. Extreme values of the interval should have a membership degree 1 to extreme labels. Mean value of the interval should have membership 1 to medium label. Likewise, we consider strong Ruspini's fuzzy partitions (Ruspini, 1969) – where the sum of the membership degrees of every value to the set of linguistic terms is 1 – in order to ensure good interpretability. Finally, in order to statistically unbias the significance of every linguistic term, we impose the same covering degree. Thus, we define the membership function shapes where, given the set S = {min, . . . ,max} defining the interval, they hold the following condition:

$$\sum_{k \in S} \mu_{A_i}(k) = \frac{\max - \min}{l}, \ \forall A_i \in \mathrm{A}, \tag{1}$$

with l being the number of linguistic terms and $\mathrm{A} = \{A_1, \ldots, A_l\}$ the set of them.

To sum up, Figure 2 shows an example based on the transformation of a nine-point rating scale (a typical marketing scale used to measure the observed variables/indicators related to certain construct) into a fuzzy semantic with the three linguistic terms *Low*, *Medium*, and *High*.

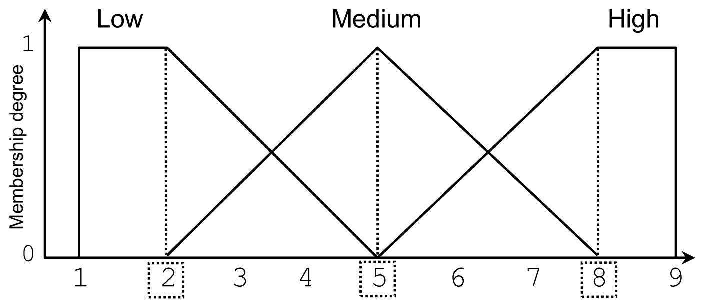


Fig. 2. Fuzzy semantic from a transformation of a 9-point marketing scale (rating scale)

Input/Output Linguistic Variables from Expert Knowledge

Furthermore, once the structure of the model has been fixed by the marketing expert under the base of the theoretic model, fuzzy rules are used to relate input (antecedents) with output (consequents) variables. Obviously, hypotheses contained in the model can be directly used to define IF-THEN structures by considering the dependencies shown among the variables. Thus, we obtain a fuzzy rule base for each consequent (endogenous construct) considered and its respective set of antecedents.

For example, if we take for illustrative purposes the model depicted in Figure 1, the fuzzy rule structure that represents the relations between the elements "Convenience Orientation" and "Risk Averseness" with the consequent "Consumer Attitude" will have the following form:

IF *Convenience Orientation* is A_1 and *Risk Averseness* is A_2 **THEN** *Consumer Attitude* is B

2.4 Data Mining Process

Once the linguistic variables that properly represent the tackled information have been fixed, a machine learning process must be used to automatically extract the knowledge existing in the database. This process is, without any doubt, the most important issue from the KDD point of view.

As mentioned in Section 1, in this paper we are interested in descriptive induction. Therefore, we will use GAs Michigan-style to obtain rules individually relevant. We consider two quality criteria, support (degree of representativity of the rule with respect to the set of data) and confidence (degree of accuracy of the relation shown by the rule). It is intuitive to check that the higher the support, the higher the difficulty to maintain high degrees of confidence. To jointly consider both criteria, we propose the use of *multiobjective GAs*, as they offer good results when working with multiple contradictory objectives. The next section describes the main elements of this method we propose.

Fuzzy Rule Structure

In data mining it is crucial to use a learning process with a high degree of interpretability. To do that, we opt for a compact description based on the disjunctive normal form (DNF). This kind of fuzzy rule structure has the following form:

$$\textbf{IF } X_1 \text{ is } \tilde{A}_1 \text{ and} \ldots \text{and } X_n \text{ is } \tilde{A}_n \textbf{ THEN } Y_1 \text{ is } B$$

where each input variable $X_i,\ i \in \{1, \ldots, n\}$ takes as a value a set of linguistic terms $\tilde{A}_i = \{A_{i1} \text{ or } \ldots \text{or } A_{in_i}\}$, whose members are joined by a disjunctive operator. We use the bounded sum $\min\{1, a+b\}$ as *T-conorm*[4]. The structure is a natural support to allow the absence of some input variables in each rule, simply making $\tilde{A}_i$ to be the whole set of linguistic terms available.

Multi-item Fuzzification

In order to properly consider the set of indicators available for each input/output variable (as discussed in Section 2.2), we propose an extension of the membership degree computation, the so-called *multi-item fuzzification*. The process is based on a union of the partial information provided by each item. Given X_i and Y_j measured by the vectors of items $\mathbf{x}_i = (x_1^{(i)}, \ldots, x_{h_i}^{(i)}, \ldots, x_{p_i}^{(i)})$ and $\mathbf{y} = (y_1, \ldots, y_t, \ldots, y_q)$, respectively, the fuzzy propositions X_i is $\tilde{A}_i$ and Y is B are respectively interpreted as follows:

$$\mu_{\tilde{A}_i}(\mathbf{x}_i) = \min\left\{1, \bigcup_{h_i=1}^{p_i} \sum_{A\in\tilde{A}_i}^{\mu_A(x_{h_i}^{(i)})}\right\} \tag{2}$$

$$\mu_B(\mathbf{y}) = \bigcup_{t=1}^{q} \mu_B(y_t), \tag{3}$$

with $\cup$ being a T-conorm (the maximum in this paper).

Subgroup Discovery

To do the descriptive rules induction process, we have applied a method with certain similarities to the subgroups discovery technique – widely used in classification learning rules (Lavrac 2004) –, where the property of interest is the class associated with the variables of the consequent. Therefore, we try to group the set of data into differentiated subgroups, including in each of them those examples represented by the consequent with the aim of discovering a representative set of rules for each subgroup. In this regard, the most usual approach is based on running the algorithm designed for each subgroup of data which satisfies the property set for the consequent.

However, instead of this approach, we carry out a simultaneous subgroup discovery in the algorithm we propose. This variant allows us to form niches of fuzzy rules differentiated by the consequent which are optimized in parallel to finally generate a set of suboptimal solutions for each class of the consequent. With the aim of developing this simultaneous process, as it is shown in the next sections, we vary the concept of multiobjective dominance by making the genetic operators act only on the antecedents of the rules.

[4] This family of binary operators is used in fuzzy logic to interpret the disjunction 'or'

Coding Scheme

Each individual of the population represents a fuzzy rule; i.e. a Michigan-style genetic algorithm. The coding scheme will be binary to represent the antecedent and whole for the consequent. Thus, the allele "1" in the antecedent part means that the linguistic term related to the gene is used in the corresponding variable. For the consequent, we will directly code the index of the linguistic term used. Hence, the size to code a DNF fuzzy rule is equal to the sum of the number of linguistic terms employed in each input variable (antecedent) plus the number of output variables. For instance, if we had three linguistic terms for each variable, the rule [IF X1 is Small and X2 is {Medium or High} THEN Y is Medium], would be coded as [100 011|2].

Objective Functions

In this algorithm, we consider the two criteria most frequently used to value the quality of the association rules (Dubois *et al.* 2005): support and confidence. However, we adapt the calculus of these criteria to fuzzy association rules, also considering the especial characteristics of the multi-item variables (elements of the model) which we work with.

Support. This objective function values the degree of representation of certain fuzzy rule on the set of data analyzed. It is calculated as the average degree covered by the rule considering every one of these data (individuals' responses). To obtain the degree of cover we conjointly consider the membership degrees in relation to the diverse variables; i.e. the set of antecedents as well as the consequent. The measure of support (for maximization) for a fuzzy rule R comes defined as follows:

$$Support(R) = \frac{1}{N}\sum_{e=1}^{N} T(\mu_A(\mathrm{x}^{(e)}), \mu_B(\mathbf{y}^{(e)})), \tag{4}$$

where N is the size of the database (the sample size or number of respondents), $\mathrm{x}^{(e)} = (\mathbf{x}_1^{(e)}, \ldots, \mathbf{x}_n^{(e)})$ and $\mathbf{y}^{(e)}$ is the eth instance multi-item of input and output respectively, T the *product* T-norm, and

$$\mu_A(\mathrm{x}^{(e)}) = \min_{i\in\{1,\ldots,n\}} \mu_{\tilde{A}_i}(\mathbf{x}_i^{(e)}) \tag{5}$$

the coverage degree of the antecedent of the rule R for this example (i.e. it is considered the T-norm of the minimum to interpret the connector "and" of the fuzzy rule). Also, it is convenient to point out that we employ the multi-item fuzzification shown in section 2.4 to calculate $\mu_{\tilde{A}_i}(\mathbf{x}_i^{(e)})$ and $\mu_B(\mathbf{y}^{(e)})$.

Confidence. This objective function measures the reliability of the relationship between antecedent and consequent described by the analyzed fuzzy rule. We have used a confidence degree that avoids accumulation of low cardinalities (Dubois *et al.* 2005). It is computed (for maximizing) as follows:

$$Confidence(R) = \frac{\sum_{e=1}^{N} T(\mu_A(\mathrm{x}^{(e)}), I(\mu_A(\mathrm{x}^{(e)}), \mu_B(\mathbf{y}^{(e)})))}{\sum_{e=1}^{N} \mu_A(\mathrm{x}^{(e)})}, \tag{6}$$

The Dienes' S-implication $I(a,b) = \max\{1-a, b\}$ is used. We consider again T-norm of product and multi-fuzzification.

Evolutionary Scheme

A generational approach with the multi-objective NSGA-II replacement strategy (Deb *et al.* 2002) is adopted. Crowding distance in the objective function space is used. Binary tournament selection based on the nondomination rank (or the crowding distance when both solutions belong to the same front) is applied.

To correctly develop the simultaneous subgroup discovery we will need to redefine the concept of dominance. In order to do this, one solution (rule) will dominate another when, besides being better or equal in all the objectives and better in at least one of them, it presents the same consequent as the other rule. Hence, those rules with different consequents do not dominate each other. Consequently, we force the algorithm to form so many niches of search (Pareto sets) as diverse consequents (subgroups) are considered.

Genetic Operators

The initial population is built defining so many groups (equal in size) as there are different consequents. In each of them, chromosomes are generated fixing such consequents and randomly building a simple antecedent where each input variable is related to a linguistic term. The two operators of reproduction only act in the part of the antecedent of the rule. This fact ensures that the size of every subgroup in the population is constant. In this way, we allow the algorithm to independently explore, but simultaneously, each group.

We employ a multipoint crossover operator which selects two crossover points (in the part of the antecedent) and interchanges the central sub-chain. The operator of mutation randomly selects a variable of the antecedent of the fuzzy rule coded in the chromosome and carries out some of the three following operations: *expansion*, which flips to 1 a gene of the selected variable; *contraction*, which flips to 0 a gene of the selected variable; or *shift*, which flips to 0 a gene of the variable and flips to 1 the gene immediately before or after it. The selection of one of these mechanisms is made randomly among the available choices (e.g., contraction cannot be applied if only a gene of the selected variable has the allele 1).

3 Experimental Results and Knowledge Interpretation

The experimentation of the descriptive rule induction method we present has been made based on a causal model already proposed by Novak *et al.* (2000). It analyzes the consumer's flow state in interactive computer-mediated environments. As the authors allow the use of their database for academic purposes, we have opted for experimenting our methodology with a consumer model already validated and widely known by the academics. This is a plausible and orthodox alternative, as we can see by analyzing other research previously developed (see, as e.g.: Beynon *et al.* 2001; Fish *et al.* 2004; Hurtley *et al.* 1995; Levy and Yoon 1995; Rhim and Cooper 2005).

3.1 Some Theoretical Notes about the Model Used for the Experimentation

In order to briefly introduce this concept, so the reader better understands the variable we want to explain in this empirical application of our methodology, we now synthetically present some ideas about it. *Flow* has been recently imported from motivational psychology and successfully adapted to explain consumer behavior phenomena on the Web (Hoffman and Novak 1996; Korzan 2003; Luna *et al.* 2002; Novak *et al.* 2000; Novak *et al.* 2003). In general terms, *flow* state is defined as "the process of optimal experience" or the mental state that individuals sometimes experience when they are deeply immersed in certain events, objects or activities (Csikszentmihalyi 1975, 1977). This concept has been adapted to the Web environment. In this context, *flow* state is achieved when the consumer is so deeply involved in the process of navigation on the Web that "nothing else seems to matter" (Hoffman and Novak 1996, p. 57).

Though the model we consider for the experimentation has 12 elements (constructs) interconnected, with 6 fuzzy rule based systems, due to the space constraints, in this paper we focus on that system which considers the four primary antecedents of the consumer's *flow*. Specifically, we consider the following four constructs, as antecedents of the consumer's flow state (consequent):

- *speed of interaction* refers to the user's perception about how quick is the process of interaction when using the Web
- *skill/control* gathers the consumer's opinion regarding his own capacity to develop successful navigating process on the Web
- *challenge/arousa*l gathers how challenging and stimulating is surfing the Web
- *telepresence/time distortion* is also a compound construct which refers to the consumer's perception about the predominance of the computer virtual (Web) environment over the physical environment where the consumer is placed when surfing the Web, as well as to the lost of the consumer's self consciousness on the notion of time when developing such process of navigation.

Novak *et al.* (2000) hypothesized that these four elements are positively related to this central construct of the model.

All these constructs were gathered by multi-item Likert-type scales with 9 points; i.e. metric scales. The fuzzy semantic we have applied to all the variables is shown in Figure 2.

Training data are composed of 1,154 examples (consumers' responses). We have run the algorithm 10 times, obtaining the following values for the parameters: 300 generations, size of the population 100, crossover probability 0.7 and the probability of mutation per chromosome 0.1.

3.2 Analysis of the Pareto Front

The Pareto front we have obtained is shown in Figure 3. With respect to the value taken by the consequent *flow* in the rules generated, it can be easily observed that the most plausible output is "medium." Indeed, there is a clear supremacy of the rules with this label in the consequent over the two other outputs in terms of support and confidence. This fact is intensified as the support of the rules grows, without noticing a relevant loss of reliability in the rules which represent medium *flow* states. Therefore, it can be inferred that the most representative state of *flow*, for the whole consumers' database, is moderate.

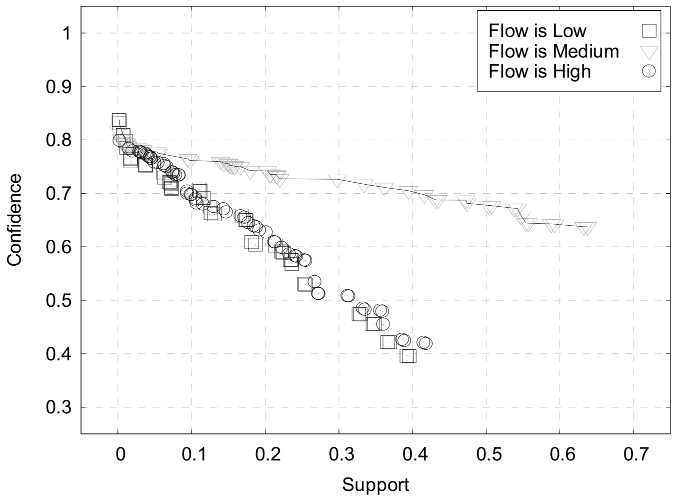


Fig. 3. Sub-Pareto fronts for every output of the consequent, as well as the absolute Pareto front (the best rules from the whole set of rules) joined by a line.

3.3 Illustrative Analysis of the Rules

An individual analysis of the rules generated by this descriptive method is very useful to better understand the consumer behavior being analyzed. Specifically, it is recommendable to do a selection of rules from the whole set compounding the absolute Pareto front, paying attention to its support (degree of representativity of the consumers' database) and, especially, to its confidence (degree of reliability of the information pattern shown by the rule). In this regard, we have done an illustrative selection shown in Table 3.3.

Table 3. Illustrative selection of rules from the absolute Pareto front. L stands for Low, M stands for medium, H stands for high.

	Speed of Interaction			*Skill/ Control*			*Challenge/ Arousal*			*Telepresence/ Time Distort.*			*Flow*	*Sup*	*Conf*
R_1	L		H		M					L			L	0.0104	0.7980
R_2		M		L		H			H		M		M	0.0102	0.7937
R_3		M									M	H	M	0.3947	0.7051

Considering the absolute Pareto front, R_1 is the rule with highest confidence, associated with low states of *flow*. Likewise, R_2 represents the most reliable rule from those with moderate *flow* states. Finally, we have also considered the rule R_3, being the one with highest support among the whole set of rules with confidence higher than 0.7; i.e. the confidence threshold value we have set to give reliability to the information patterns shown by the rules.

Synthetically, from the four antecedents considered, it highlights the influence of the perception about *telepresence/time distortion* (TP/TD) in determining consumers' states of *flow*; it can be observed how its value is determinant in explaining low (R_1) or moderate (R_2 and R_3) states of *flow*. Likewise, the rest of the antecedents seem to exert a poor or null influence on the consequent. This fact can also be due to the element TP/TD that eclipses the influence of the rest. In any case, it conforms to the main idea we extracted when the Pareto front was analyzed; i.e. a non existence of combinations of antecedents (rules) producing high states of *flow*, with significant levels of reliability and representativity. In this sense, it is quite illustrative to see how even when the most influential antecedent – i.e. TP/TD – takes high values, the consumer's *flow* state in the process of navigation tends to remain moderate.

4 Concluding Remarks

We have faced an interesting problem of KDD in relation to marketing causal modeling and its resolution by genetic fuzzy systems. The problem presents a specific type of data with uncertainty which justifies the use of fuzzy rules. Furthermore, we have practiced a multi-objective optimization in order to obtain rules with high degrees of support and confidence. The KDD methodology proposed has been successfully applied to a real problem of consumer behavior in online environments.

In our research agenda, we have the use of other metrics such as consistency and interest of the rules. Also, the unsupervised learning of fuzzy association rules, i.e. without using any antecedent or consequent previously fixed by the marketing expert.

Acknowledgements

Research supported in part by the Spanish Ministry of Education and Science under project no. TIN2005-08386-C05-01

References

Beynon M, Curry B, Morgan P. (2001) Knowledge discovery in marketing. An approach through rough set theory. European Journal of Marketing 35(7/8): 915–935.

Casillas J, Martínez-López FJ, Martínez FJ (2004) Fuzzy association rules for estimating consumer behaviour models and their application to explaining trust in Internet shopping. Fuzzy Economic Review IX(2): 3–26.

Csikszentmihalyi M (1975) Play and intrinsic rewards. Journal of Humanistic Psychology 15(3): 41–63.

Csikszentmihalyi M (1977) Beyond boredom and anxiety (Second edition). San Francisco: Jossey-Bass.

Deb K, Pratap A, Agarwal S, Meyarevian T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2): 182–197.

Dubois D, Prade H, Sudkamp T (2005) On the representation, measurement, and discovery of fuzzy associations. IEEE Transactions on Fuzzy Systems 13(2): 250–262.

Fish KE, Johnson JD, Dorsey RE, Blodgett JG (2004) Using an artificial neural network trained with a genetic algorithm to model brand share. Journal of Business Research 57 (1): 79–85.

Gatignon H (2000) Commentary on Peter Leeflang and Dick Wittink's "Building models form marketing decisions: past, present and future". International Journal of Research in Marketing 17: 209–214.

Hoffman D, Novak T (1996) Marketing in hypermedia computer-mediated environments: conceptual foundations Journal of Marketing 60 (July): 50–68.

Hurley S, Moutinho L, Stephens NM (1995) Solving marketing optimization problems using genetic algorithms. European Journal of Marketing 29 (4): 39–56.

Korzaan ML (2003) Going with the flow: predicting online purchase intentions. Journal of Computer Information Systems (Summer): 25–31.

Lavrac N, Cestnik B, Gamberger D, Flach P (2004) Decision support through subgroup discovery: three case studies and the lessons learned. Machine Learning 57 (1–2): 115–143.

Lee, B.C.Y. (2007) Consumer attitude toward virtual stores and its correlates: Journal of Retailing and Consumer Services 14(3): 182-191.

Levy JB, Yoon E (1995) Modeling global market entry decision by fuzzy logic with an application to country risk assessment. European Journal of Operational Research 82: 53–78.

Luna D, Peracchio LA, De Juan MD (2002) Cross-cultural and cognitive aspects of Web site navigation. Journal of the Academy of Marketing Science 30(4): 397–410.

Novak T, Hoffman D, Duhachek A (2003) The influence of goal-directed and experiential activities on online flow experiences. Journal of Consumer Psychology 13 (1/2): 3–16.

Novak T, Hoffman D, Yung Y (2000) Measuring the customer experience in online environments: A structural modeling approach. Marketing Science 19 (1): 22–42.

Rhim H, Cooper LG (2005) Assessing potential threats to incumbent brands: New product positioning under price competition in a multisegmented market. International Journal of Research in Marketing 22: 159–182.

Ruspini E (1969) A new approach to clustering, Information and Control 15: 22-32.

Shim JP, Warkentin M, Courtney JF, Power, DJ, Sharda R, Carlsson C (2002) Past, present and future of decision support technology. Decision Support Systems 33: 111–126.

Steenkamp J, Baumgartner H (2000) On the use of structural equation models for marketing modeling. International Journal of Research in Marketing 17: 195–202.

Wedel M, Kamakura W. Böckenholt U (2000) Marketing data, models and decisions. International Journal of Research in Marketing 17: 203–208.

Knowledge Discovery in a Framework for Modelling with Words

Zengchang Qin[1] and Jonathan Lawry[2]

[1] Berkeley Initiative in Soft Computing (BISC), Computer Science Division, EECS Department, University of California, Berkeley, CA 94720, US.
`zqin@eecs.berkeley.edu`
[2] Artificial Intelligence Group, Department of Engineering Mathematics, University of Bristol, BS8 1TR, UK.
`j.lawry@bris.ac.uk`

Summary. The learning of transparent models is an important and neglected area of data mining. The data mining community has tended to focus on algorithm accuracy with little emphasis on the knowledge representation framework. However, the transparency of a model will help practitioners greatly in understanding the trends and idea hidden behind the system. In this chapter, a random set based knowledge representation framework for learning linguistic models is introduced. This framework is referred to as label semantics and a number of data mining algorithms are proposed. In this framework, a vague concept is modelled by a probability distribution over a set of appropriate fuzzy labels which is called as mass assignment. The idea of mass assignment provides a probabilistic approach for modelling uncertainty based on pre-defined fuzzy labels.

1 Introduction

Fuzzy Logic was first proposed by Zadeh (Zadeh, 1965) as an extension of traditional binary logic. In contrast to a classical set, which has a crisp boundary, the boundary of a fuzzy set is blurred and the transition is characterized by *membership functions*. In early research fuzzy logic was successfully applied in control systems and expert systems where the linguistic interpretation fuzzy sets allowed for an interface between the human user and a computer system. Because our language is fuzzy, the concepts represented by language is full of uncertainty and impreciseness. Therefore, fuzzy sets can be used to model language. This idea also motivates related research into Computing with Words (Zadeh, 1996) and Perception-based Reasoning (Zadeh, 2002).

Almost all the labels we give to characterize a group of objects are fuzzy. Given a fuzzy set, an object may belong to this set with a certain *membership value*. In traditional set theory, this membership value only has two possible values, 1 and 0, representing the case where the object belongs to or does not belong to the set, respectively. In a fuzzy set, the membership values are continuous real values from 0 to 1. We use a fuzzy term such as 'big' to label a particular group, because they share the property of objects within this group (i.e., they are big). The objects within this group will have different membership values varying from 0 to 1 qualifying the degree to which they satisfy the concept 'big'. An object with membership of 0.8 is more likely to be described as 'big' than an object with membership of 0.4. If we consider this problem in another way. Given an object, label 'big' can be used to describe this object with some appropriateness degrees. Follow this idea, we discuss a new approach based on random set theory to interpret imprecise concepts. This framework, first proposed by Lawry (Lawry, 2001) and is referred to as *Label Semantics*, can be regarded as an approach to Modelling with Words (Lawry *et al.*, 2003).

Modeling with Words is a new research area which emphasis "modelling" rather than "computing". For example, Zadeh's theories on Perception-based Computing (Zadeh, 2002) and Precisiated Natural Language (Zadeh, 2005) are the approaches of "computing". However, the relation between it and Computing with Words (Zadeh, 1996) is close is likely to become even closer (Zadeh, 2003). Both of the research areas are aimed at enlarging the role of natural languages in scientific theories, especially, in knowledge management, decision and control. In this chapter, the framework is mainly used for modelling and building intelligent machine learning and data mining systems. In such systems, we use words or fuzzy labels for modelling uncertainty. Therefore, the research presented here is considered as a framework for modelling with words.

This chapter is organized as follows: A systematic introduction on label semantics is given in the first section. Based on the framework we introduced, we will give the details of several data mining models based on label semantics: Linguistic Decision Trees in section 3, Label semantics based Bayesian estimation in section 4, and Linguistic Rule Induction in section 5. Finally, we give the summary and discussions in the final section.

2 Label Semantics

Vague or imprecise concepts are fundamental to natural language. Human beings are constantly using imprecise language to communicate each other. We usually say 'Peter is tall and strong' but not 'Peter is exactly 1.85 meters in height and he can lift 100kg weights'. We will focus on developing an understanding of how an intelligent agent can use vague concepts to convey information and meaning as part of a general strategy for practical reasoning and decision making. Such an agent can could be an artificial intelligence program or a human, but the implicit assumption is that their use of vague concepts is governed by some underlying internally consistent strategy or algorithm. We may notice that *labels* are used in natural language to describe what we see, hear and feel. Such labels may have different degrees of vagueness (i.e., when we say Peter is *young* and he is *male*, the label *young* is more vague than the label *male* because people may have more widely different opinions on being *young* than being *male*. For a particular concept, there could be more than one label that is appropriate for describing this concept, and some labels could be more appropriate than others. Here, we will use a random set framework to interpret these facts. *Label Semantics*, proposed by Lawry (Lawry, 2001), is a framework for modelling with linguistic expressions, or labels such as *small*, *medium* and *large*. Such labels are defined by overlapping fuzzy sets which are used to cover the universe of continuous variables.

2.1 Mass Assignment on Fuzzy Labels

For a variable x into a domain of discourse Ω we identify a finite set of linguistic labels $\mathcal{L} = \{L_1, \cdots, L_n\}$ with which to label the values of x. Then for a specific value $x \in \Omega$ an individual I identifies a subset of $\mathcal{L}$, denoted D_x^I to stand for the description of x given by I, as the set of labels with which it is appropriate to label x. The underlying question posed by label semantics is how to use linguistic expressions to label numerical values. If we allow I to vary across a population V with prior distribution P_V, then D_x^I will also vary and generate a random set denoted D_x into the power set of $\mathcal{L}$ denoted by $\mathcal{S}$. We can view the random set D_x as a description of the variable x in terms of the labels in $\mathcal{L}$. The frequency of occurrence of a particular label, say S, for D_x across the population then gives a distribution on D_x referred to as a mass assignment on labels. More formally,

Definition 1 (*Label Description*) *For $x \in \Omega$ the label description of x is a random set from V into the power set of $\mathcal{L}$, denoted D_x, with associated distribution m_x, which is referred to as mass assignment:*

$$\forall S \subseteq \mathcal{L}, \quad m_x(S) = P_V(\{I \in V | D_x^I = S\}) \tag{1}$$

where P_V is the prior distribution of population V. $m_x(S)$ is called the mass associated with a set of labels S and

$$\sum_{S \subseteq \mathcal{L}} m_x(S) = 1 \tag{2}$$

Intuitively mass assignment is a distribution on appropriate label sets and $m_x(S)$ quantifies the evidence that S is the set of appropriate labels for x.

For example, given a set of labels defined on the temperature outside: $\mathcal{L}_{Temp} = \{low, medium, high\}$. Suppose 3 of 10 people agree that '*medium* is the only appropriate label for the temperature of 15° and 7 agree 'both *low* and *medium* are appropriate labels'. According to def. 1,

$$m_{15}(medium) = 0.3 \ and \ m_{15}(low, medium) = 0.7$$

so that the mass assignment for 15° is $m_{15} = \{medium\} : 0.3$, $\{low, medium\}$: 0.7. More details about the theory of mass assignment can be found in (Baldwin *et al.*, 1995).

2.2 Appropriateness Degrees

Consider the previous example, can we know how appropriate for a single label, say *low*, to describe 15°? In this framework, *appropriateness degrees* are used to evaluate how appropriate a label is for describing a particular value of variable x. Simply, given a particular value α of variable x, the appropriateness degree for labeling this value with the label L, which is defined by fuzzy set F, is the membership value of α in F. The reason we use the new term 'appropriateness degrees' is partly because it more accurately reflects the underlying semantics and partly to highlight the quite distinct calculus based on this framework (Lawry, 2001). This definition provides a relationship between mass assignments and appropriateness degrees.

Definition 2 (*Appropriateness Degrees*)

$$\forall x \in \Omega, \ \forall L \in \mathcal{L} \quad \mu_L(x) = \sum_{S \subseteq \mathcal{L}: L \in S} m_x(S)$$

Consider the previous example, we then can obtain $\mu_{medium}(15) = 0.7 + 0.3 = 1$, $\mu_{low}(15) = 0.7$. It is also important to note that, given definitions for the appropriateness degrees on labels, we can isolate a set of subsets of $\mathcal{L}$ with non-zero masses. These are referred to as *focal sets* and the appropriate labels with non-zero masses as *focal elements*, more formally,

Definition 3 (*Focal Set*) *The focal set of $\mathcal{L}$ is a set of focal elements defined as:*

$$\mathcal{F} = \{S \subseteq \mathcal{L} | \exists x \in \Omega, m_x(S) > 0\}$$

Given a particular universe, we can then always find the unique and consistent translation from a given data element to a mass assignment on focal elements, specified by the function $\mu_L : L \in \mathcal{L}$.

2.3 Linguistic Translation

Based on the underlying semantics, we can translate a set of numerical data into a set of mass assignments on appropriate labels based on the reverse of definition 2 under the following assumptions: consonance mapping, full fuzzy covering and 50% overlapping (Qin and Lawry, 2005b). Consonance assumption implies that voters are agreed with the natural order of fuzzy labels. A voter won't set 'small' and 'large' as appropriate labels without 'medium'. These assumptions are fully described in (Qin and Lawry, 2005b) and justified in (Lawry, 2004). These assumptions guarantee that there is unique mapping from appropriate degrees to mass assignments on labels. For example, Figure 1 shows the universes of two variables x_1 and x_2 which are fully covered by 3 fuzzy sets with 50% overlap, respectively. For x_1, the following focal elements occur:

$$\mathcal{F}_1 = \{\{small_1\}, \{small_1, medium_1\}, \{medium_1\}, \{medium_1, large_1\}, \{large_1\}\}$$

Since $small_1$ and $large_1$ do not overlap, the set $\{small_1,\ large_1\}$ cannot occur as a focal element according to def. 3. We can always find a unique translation from a given data point to a mass assignment on focal elements, as specified by the function μ_L. Given a particular data element, the sum of associated mass is 1. This is referred to as *linguistic translation*. Suppose we are given a numerical data set $\mathcal{D} = \{\langle x_1(i), \ldots, x_n(i)\rangle | i = 1, \ldots, N\}$ and focal set on attribute j: $\mathcal{F}_j = \{F_j^1, \ldots, F_j^{h_j} | j = 1, \ldots, n\}$, we can obtain the following new data base by applying linguistic translation described in Algorithm 1.

Algorithm 1: Linguistic translation

```
input : Given a database $\mathcal{D} = \{\langle x_1(i), \cdots, x_n(i)\rangle | i = 1, \cdots, |\mathcal{D}|\}$ with
        associated classes $\mathcal{C} = \{C_1, \cdots, C_{|\mathcal{C}|}\}$
output: Linguistic dataset $\mathcal{LD}$

1 for $j \leftarrow 1$ to $n$ do
2     foreach $x_j$ do : Cover the universe of $x_j$ with $N_F$ trapezoidal fuzzy
      sets with 50% overlap.
3     for $i \leftarrow 1$ to $|\mathcal{D}|$ do
4         foreach Data element $x_j(i)$ do
5         Read appropriateness degrees for $x_j(i)$ from corresponding fuzzy
          set.
6         Calculating corresponding mass assignments:
          $\mathcal{LD}_{i,j} = \langle m_{x(i)}(F_j^1), \cdots, m_{x(i)}(F_j^{h_j})\rangle$ on focal elements from
          appropriateness degrees.
7 Save dataset $\mathcal{LD}$ where $\mathcal{LD} = \{\mathcal{LD}_{i,j} | i = 1, \cdots, |\mathcal{D}|, j = 1, \cdots, n\}$
```

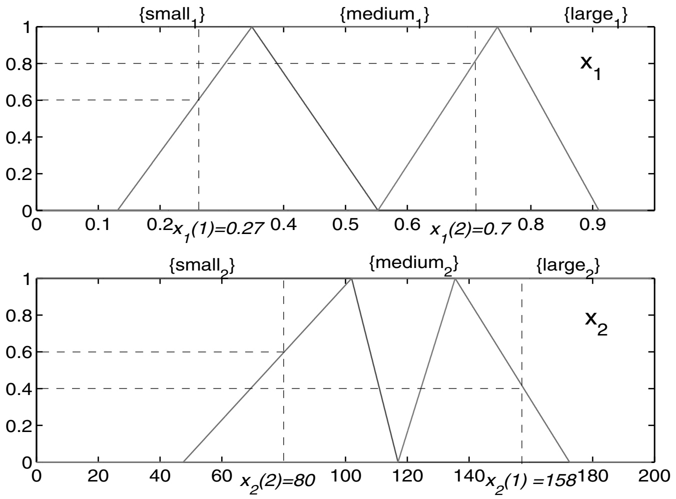


Fig. 1. A full fuzzy covering (discretization) using three fuzzy sets with 50% overlap on two attributes x_1 and x_2, respectively.

For a particular attribute with an associated focal set, linguistic translation is a process of replacing its data elements with the focal element masses of these data elements. See figure 1. $\mu_{small_1}(x_1(1) = 0.27) = 1$, $\mu_{medium_1}(0.27) = 0.6$ and $\mu_{large_1}(0.27) = 0$. They are simply the memberships read from the fuzzy sets. We then can obtain the mass assignment of this data element according to def. 2 under the consonance assumption (Qin and Lawry, 2005b): $m_{0.27}\ (small_1) = 0.4$, $m_{0.27}(small_1,\ medium_1) = 0.6$. Similarly, the linguistic translations for two data:

$$\mathbf{x}_1 = \langle x_1(1) = 0.27\rangle, \langle x_2(1) = 158\rangle$$

$$\mathbf{x}_2 = \langle x_1(2) = 0.7\rangle, \langle x_2(2) = 80\rangle$$

are illustrated on each attribute independently as follows:

$$\begin{bmatrix} x_1 \\ \hline x_1(1) = 0.27 \\ x_1(2) = 0.7 \end{bmatrix} \overset{LT}{\rightarrow} \begin{bmatrix} m_x(\{s_1\}) & m_x(\{s_1, m_1\}) & m_x(\{m_1\}) & m_x(\{m_1, l_1\}) & m_x(\{l_1\}) \\ \hline 0.4 & 0.6 & 0 & 0 & 0 \\ 0 & 0 & 0.2 & 0.8 & 0 \end{bmatrix}$$

$$\begin{bmatrix} x_2 \\ \hline x_2(1) = 158 \\ x_2(2) = 80 \end{bmatrix} \overset{LT}{\rightarrow} \begin{bmatrix} m_x(\{s_2\}) & m_x(\{s_2, m_2\}) & m_x(\{m_2\}) & m_x(\{m_2, l_2\}) & m_x(\{l_2\}) \\ \hline 0 & 0 & 0 & 0.4 & 0.6 \\ 0.4 & 0.6 & 0 & 0 & 0 \end{bmatrix}$$

Therefore, we can obtain:

$$\mathbf{x}_1 \rightarrow \langle \{s_1\} : 0.4, \{s_1, m_1\} : 0.6\rangle, \langle \{m_2, l_2\} : 0.4, \{l_2\} : 0.6\rangle$$

$$\mathbf{x}_2 \rightarrow \langle \{m_1\} : 0.2, \{m_1, l_1\} : 0.8 \rangle, \langle \{s_2\} : 0.4, \{s_2, m_2\} : 0.6 \rangle$$

We may notice that the new mass assignment based data generated by linguistic translation is depending on the way of universe discretization. Different discretizations may result in different data. Since we will use the new data for training data mining models in the following sections. We hope our data could be as discriminate as possible. A few empirical experiments have been done in (Qin and Lawry, 2005b) and the percentile-based (or equal point) discretization is a fairly good method where each fuzzy label covers approximately the same number of data points. In this chapter, unless otherwise stated, we will use this method for discretizing the continuous universe.

2.4 Linguistic Reasoning

As a high-level knowledge representation language for modelling vague concepts, label semantics allows linguistic reasoning. Given a universe of discourse Ω containing a set of objects or instances to be described, it is assumed that all relevant expressions can be generated recursively from a finite set of basic labels $\mathcal{L} = \{L_1, \ldots, L_n\}$. Operators for combining expressions are restricted to the standard logical connectives of negation "$\neg$", conjunction "$\wedge$", disjunction "$\vee$" and implication "$\rightarrow$". Hence, the set of logical expressions of labels can be formally defined as follows:

Definition 4 (*Logical Expressions of Labels*) *The set of logical expressions, LE, is defined recursively as follows:*

(i) $L_i \in LE$ *for* $i = 1, \ldots, n$.
(ii) If $\theta, \varphi \in LE$ *then* $\neg\theta, \theta \wedge \varphi, \theta \vee \varphi, \theta \rightarrow \varphi \in LE$

Basically, we interpret the main logical connectives as follows: $\neg L$ means that L is not an appropriate label, $L_1 \wedge L_2$ means that both L_1 and L_2 are appropriate labels, $L_1 \vee L_2$ means that either L_1 or L_2 are appropriate labels, and $L_1 \rightarrow L_2$ means that L_2 is an appropriate label whenever L_1 is. As well as labels for a single variable, we may want to evaluate the appropriateness degrees of a complex logical expression $\theta \in LE$. Consider the set of logical expressions LE obtained by recursive application of the standard logical connectives in $\mathcal{L}$. In order to evaluate the appropriateness degrees of such expressions we must identify what information they provide regarding the the appropriateness of labels. In general, for any label expression θ we should be able to identify a maximal set of label sets, $\lambda(\theta)$ that are consistent with θ so that the meaning of θ can be interpreted as the constraint $D_x \in \lambda(\theta)$.

Definition 5 (λ*-function*) *Let* θ *and* φ *be expressions generated by recursive application of the connectives* $\neg, \vee, \wedge$ *and* $\rightarrow$ *to the elements of* $\mathcal{L}$ *(i.e.*

$\theta, \varphi \in LE$). Then the set of possible label sets defined by a linguistic expression can be determined recursively as follows:

(i) $\lambda(L_i) = \{S \subseteq \mathcal{F} | \{L_i\} \subseteq S\}$
(ii) $\lambda(\neg\theta) = \overline{\lambda(\theta)}$
(iii) $\lambda(\theta \wedge \varphi) = \lambda(\theta) \cap \lambda(\varphi)$
(iv) $\lambda(\theta \vee \varphi) = \lambda(\theta) \cup \lambda(\varphi)$
(v) $\lambda(\theta \rightarrow \varphi) = \overline{\lambda(\theta)} \cup \lambda(\varphi)$

It should also be noted that the λ-function provides us with notion of logical equivalence '$\equiv_L$' for label expressions

$$\theta \equiv_L \varphi \Longleftrightarrow \lambda(\theta) = \lambda(\varphi)$$

Basically, the λ-function provides a way of transferring logical expressions of labels (or linguistic rules) to random set descriptions of labels (i.e. focal elements). $\lambda(\theta)$ corresponds to those subsets of $\mathcal{F}$ identified as being possible values of D_x by expression θ. In this sense the imprecise linguistic restriction 'x is θ' on x corresponds to the strict constraint $D_x \in \lambda(\theta)$ on D_x. Hence, we can view label descriptions as an alternative to linguistic variables as a means of encoding linguistic constraints.

2.5 High Level Label Description

In this section, we will consider how to use a high level fuzzy label to describe another fuzzy label. Here the term *high level* does not mean a hieracrhial structure. We will actually consider two set of fuzzy labels which are independently defined on the same universe. If the cardinality of a set of labels $\mathcal{L}$ is denoted by $|\mathcal{L}|$. We then can say $\mathcal{L}_1$ higher level labels of $\mathcal{L}_2$ if $\mathcal{L}_1 < \mathcal{L}_2$. We will acutally consider the methodology of using one set of fuzzy labels to represent the other set of fuzzy labels.

For example, a fuzzy concept *about_m* is defined by an interval on [a, b] (see the left-hand side figure of fig. 2), so that the appropriateness degree of using fuzzy label *small* to label *about_m* is:

$$\mu_{small}(about\_m) = \frac{1}{b-a}\int_a^b \mu_{small}(u)du \qquad (3)$$

If the vagueness of the concept *about_m* depends on the interval denoted by δ where the length of the interval $|\delta| = b - a$. We then can obtain:

$$\mu_{small}(about\_m) = \frac{1}{|\delta|}\int_{u\in\delta} \mu_{small}(u)du \qquad (4)$$

If *about_m* is defined by other fuzzy labels rather than an interval, for example, a triangular fuzzy set (e.g., the right-hand side figure of fig. 2). How can we define the appropriateness degrees?

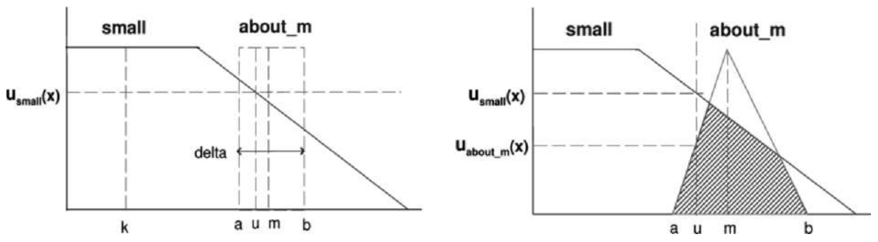


Fig. 2. The appropriateness degree of using *small* to label vague concept *about_m* is defined by the ratio of the area covered by both labels to the area covered by *about_m* only.

We begin by considering a data element $x \in [a, b]$, the function $\mu_{about\_m}(x)$ represents the degree of x belonging to the fuzzy label F. Function $\mu_{small}(x)$ defines the appropriateness degrees of using label *small* to describe x [3]. We essentially hope to obtain the appropriateness degrees of using *small* to label *about_m*. We then consider the each elements belonging to *about_m*. If $\mu_{about\_m}(x) = 1$, which means x is absolutely belonging to *about_m*, then the appropriateness degree is just $\mu_{small}(x)$. However, if $\mu_{about\_m} < \mu_{small}(x)$, we can only say it is belonging to *about_m* in certain degrees. Logically, fuzzy operation AND is used, and in practical calculation, the min(·) function is employed. The appropriateness is then defined by:

$$\mu_{small}(about\_m) = \frac{\int_{u \in \delta} \min(\mu_{small}(u), \mu_{about\_m}(u))du}{\int_{u' \in \delta} \mu_{about\_m}(u')du'} \tag{5}$$

where function $\min(x, y)$ returns the minimum value between x and y. Equation 4 is a special case of equation 5 where the following equations always hold:

$$\mu_{small}(u) = \min(\mu_{small}(u), \mu_{about\_m}(u))$$

$$|\delta| = \int_{u \in \delta} \mu_{about\_m}(u)du$$

Definition 6 *Given a vague concept (or a fuzzy label) F and a set of labels $\mathcal{L} = \{L_1, \ldots, L_m\}$ defined on a continuous universe Ω. The appropriateness degrees of using label L ($L \in \mathcal{L}$) to describe F is:*

[3] Here we interpret $\mu(\cdot)$ in different manners: membership function and appropriateness degrees, though they are mathematically the same.

$$\mu_L(F) = \frac{\int_{u\in\delta} \min(\mu_L(u), \mu_F(u))du}{\int_{u'\in\delta} \mu_F(u')du'} \tag{6}$$

where δ is the universe covered by fuzzy label F.

Given appropriateness degrees, the mass assignment can be obtained from the appropriateness degrees by the consonance assumption. Equation 5 is a general form for all kinds of fuzzy sets which are not limited to an interval or a triangular fuzzy sets.

3 Linguistic Decision Tree

Tree induction learning models have received a great deal of attention over recent years in the fields of machine learning and data mining because of their simplicity and effectiveness. Among them, the ID3 (Quinlan, 1986) algorithm for decision trees induction has proved to be an effective and popular algorithm for building decision trees from discrete valued data sets. The C4.5 (Quinlan, 1990) algorithm was proposed as a successor to ID3 in which an entropy based approach to crisp partitioning of continuous universes was adopted. One inherent disadvantage of crisp partitioning is that it tends to make the induced decision trees sensitive to noise. This noise is not only due to the lack of precision or errors in measured features but is often present in the model itself since the available features may not be sufficient to provide a complete model of the system. For each attribute, disjoint classes are separated with clearly defined boundaries. These boundaries are 'critical' since a small change close to these points will probably cause a complete change in classification. Due to the existence of uncertainty and imprecise information in real-world problems, the class boundaries may not be defined clearly. In this case, decision trees may produce high misclassification rates in testing even if they perform well in training. To overcome this problems, many fuzzy decision tree models have been proposed (Baldwin *et al.*, 1997, Janikow, 1998, Olaru and Wehenkel, 2003, Peng and Flach, 2001).

Linguistic decision tree (LDT) (Qin and Lawry, 2005b) is a tree-structured classification model based on label semantics. The information heuristics used for building the tree are modified from Quinlan's ID3 (Quinlan, 1986) in accordance with label semantics. Given a database of which each instance is labeled by one of the classes: $\{C_1, \cdots, C_M\}$. A linguistic decision tree with S consisting branches built from this database can be defined as follows:

$$T = \{\langle B_1, P(C_1|B_1), \cdots, P(C_M|B_1)\rangle, \cdots \langle B_S, P(C_1|B_S), \cdots, P(C_M|B_S)\rangle\}$$

where $P(C_k|B)$ is the probability of class C_k given a branch B. A branch B with d nodes (i.e., the length of B is d) is defined as: $B = \langle F_1, \cdots, F_d\rangle$, where

$d \leq n$ and $F_j \in \mathcal{F}_j$ is one of the focal elements of attribute j. For example, consider the branch: $\langle\langle\{small_1\}, \{medium_2, large_2\}\rangle, 0.3, 0.7\rangle$. This means the probability of class C_1 is 0.3 and C_2 is 0.7 given attribute 1 can only be described as *small* and attribute 2 can be described as both *medium* and *large*.

These class probabilities are estimated from a training set $\mathcal{D} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ where each instance $\mathbf{x}$ has n attributes: $\langle x_1, \cdots, x_n\rangle$. We now describe how the relevant branch probabilities for a LDT can be evaluated from a database. The probability of class C_k ($k = 1, \cdots, M$) given B can then be evaluated as follows. First, we consider the probability of a branch B given $\mathbf{x}$:

$$P(B|\mathbf{x}) = \prod_{j=1}^{d} m_{x_j}(F_j) \tag{7}$$

where $m_{x_j}(F_j)$ for $j = 1, \cdots, d$ are mass assignments of single data element x_j. For example, suppose we are given a branch $B = \langle\{small_1\}, \{medium_2, large_2\}\rangle$ and data $\mathbf{x} = \langle 0.27, 158\rangle$ (the linguistic translation of $\mathbf{x}_1$ was given in section 2.3). According to eq. 7:

$$P(B|\mathbf{x}) = m_{x_1}(\{small_1\}) \times m_{x_2}(\{medium_2, large_2\}) = 0.4 \times 0.4 = 0.16$$

The probability of class C_k given B can then be evaluated by:

$$P(C_k|B) = \frac{\sum_{i \in \mathcal{D}_k} P(B|\mathbf{x}_i)}{\sum_{i \in \mathcal{D}} P(B|\mathbf{x}_i)} \tag{8}$$

where $\mathcal{D}_k$ is the subset consisting of instances which belong to class k. In the case where the denominator is equals to 0, which may occur when the training database for the LDT is small, then there is no non-zero linguistic data covered by the branch. In this case, we obtain no information from the database so that equal probabilities are assigned to each class. $P(C_k|B) = \frac{1}{M}\ for\ k = 1, \cdots, M$. In the case that a data element appears beyond the range of training data set, we then assign the appropriateness degrees of the minimum or maximum values of the universe to the data element depending on which side of the range it appears.

According to the Jeffrey's rule (Jeffrey, 1965) the probabilities of class C_k given a LDT with S branches are evaluated as follows:

$$P(C_k|\mathbf{x}) = \sum_{s=1}^{S} P(C_k|B_s)P(B_s|\mathbf{x}) \tag{9}$$

where $P(C_k|B_s)$ and $P(B_s|\mathbf{x})$ are evaluated based on equations 7 and 8.

3.1 Linguistic ID3 Algorithm

Linguistic ID3 (LID3) is the learning algorithm we propose for building the linguistic decision tree based on a given linguistic database. Similar to the ID3 algorithm (Quinlan, 1986), search is guided by an information based heuristic, but the information measurements of a LDT are modified in accordance with label semantics. The measure of information defined for a branch B and can be viewed as an extension of the entropy measure used in ID3.

Definition 7 (*Branch Entropy*) *The entropy of branch B given a set of classes $\mathcal{C} = \{C_1, \ldots, C_{|\mathcal{C}|}\}$ is*

$$E(B) = -\sum_{t=1}^{|\mathcal{C}|} P(C_t|B) \log_2 P(C_t|B) \tag{10}$$

Now, given a particular branch B suppose we want to expand it with the attribute x_j. The evaluation of this attribute will be given based on the *Expected Entropy* defined as follows:

$$EE(B, x_j) = \sum_{F_j \in \mathcal{F}_j} E(B \cup F_j) \cdot P(F_j|B) \tag{11}$$

where $B \cup F_j$ represents the new branch obtained by appending the focal element F_j to the end of branch B. The probability of F_j given B can be calculated as follows:

$$P(F_j|B) = \frac{\sum_{i \in \mathcal{D}} P(B \cup F_j|\mathbf{x}_i)}{\sum_{i \in \mathcal{D}} P(B|\mathbf{x}_i)} \tag{12}$$

We can now define the *Information Gain (IG)* obtained by expanding branch B with attribute x_j as:

$$IG(B, x_j) = E(B) - EE(B, x_j) \tag{13}$$

The pseudo-code is listed in Algorithm 2. The goal of tree-structured learning models is to make subregions partitioned by branches be less “impure”, in terms of the mixture of class labels, than the unpartitioned dataset. For a particular branch, the most suitable free attribute for further expanding (or partitioning), is the one by which the “pureness” is maximally increased with expanding. That corresponds to selecting the attribute with maximum information gain. As with ID3 learning, the most informative attribute will form the root of a linguistic decision tree, and the tree will expand into branches associated with all possible focal elements of this attribute. For each branch, the free attribute with maximum information gain will be the next node, from level to level, until the tree reaches the maximum specified depth has been reached.

3.2 Degrees of Fuzziness

Through linguistic translation, all numerical data can be represented as mass assignments based on a predefined fuzzy discretization method. In this section,

Algorithm 2: Decision Tree Learning

```
input  : $\mathcal{LD}$: Linguistic dataset obtained from Algorithm 1.
output: $LDT$: Linguistic Decision Tree
1  Set a maximum depth $M_{dep}$ and a threshold probability $T$.
2  for $l \leftarrow 0$ to $M_{dep}$ do
3      $\mathcal{B} \leftarrow \emptyset$ when $l = 0$
4      The set of branches of LDT at depth $l$ is $\mathcal{B}_l = \{B_1, \cdots, B_{|\mathcal{B}_l|}\}$
5      for $v \leftarrow 1$ to $|\mathcal{B}|$ do
6          foreach $B_v$ do :
7          for $t \leftarrow 1$ to $|\mathcal{C}|$ do
8              foreach $t$ do Calculating conditional probabilities:
               $P(C_t|B_v) = \sum_{i \in \mathcal{D}_t} P(B_v|\mathbf{x}_i) / \sum_{i \in \mathcal{D}} P(B_v|\mathbf{x}_i)$
9              if $P(C_t|B_v) \geq T$ then
10                 break (step out the loop)
11         if $\exists$ $x_j$: $x_j$ is free attribute then
12             foreach $x_j$ do : Calculate: $IG(B_v, x_j) = E(B_v) - EE(B_v, x_j)$
13             $IG_{max}(B_v) = \max_{x_j}[IG(B_v, x_j)]$
14             Expanding $B_v$ with $x_{max}$ where $x_{max}$ is the free attribute we
               can obtain the maximum $IG$ value $IG_{max}$.
15             $\mathcal{B}'_v \leftarrow \bigcup_{F_j \in \mathcal{F}_j} \{B_v \cup F_j\}$.
16         else
17             exit;
18     $\mathcal{B}_{l+1} \leftarrow \bigcup_{r=1}^{s} \mathcal{B}'_r$.
19 $LDT = \mathcal{B}$
```

unless otherwise stated, we will use a percentile-based (or equal points) discretization. The idea is to cover approximately the same number of data points for each fuzzy label. The justification for using this discretization method is given in (Qin and Lawry, 2005b).

Basically, fuzzy discretization provides an interpretation between numerical data and their corresponding linguistic data based on label semantics. We may notice that different fuzzy discretization may result in different linguistic data. We introduce a new parameter PT by which to measure the degrees of overlapping between fuzzy labels. As we can see from figure 3, given two fuzzy labels F and G, m is the distance between the weighting centers of a fuzzy labels to the meeting point of their membership functions. a is actually the length of the overlapping area. PT is calculated as follows:

$$PT = a/2m \tag{14}$$

$PT = 0.5$ represents 50% of overlapping between each two neighboring fuzzy labels (e.g., figure 3-A). $PT = 0$ represents no overlapping at all (figure 3-C), i.e., the labels are discrete but not fuzzy. Figure 3-B shows a situation that

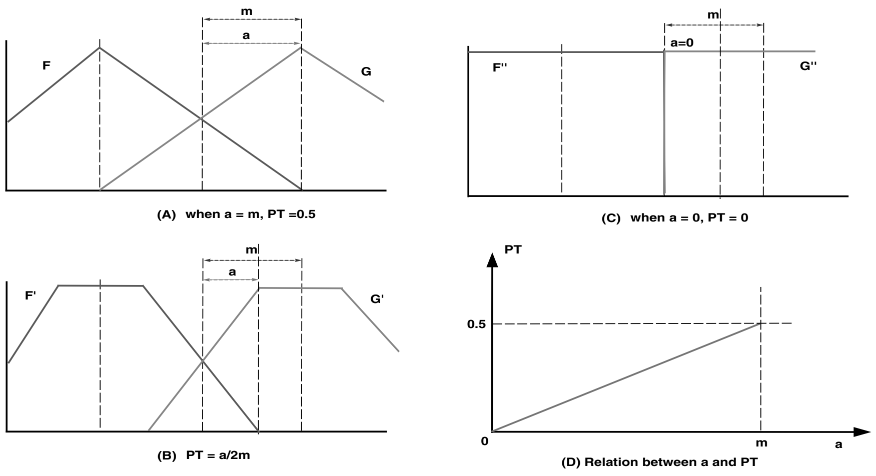


Fig. 3. A schematic illustration of calculating the overlap parameter PT given different degrees of overlaps.

the degree of overlapping is between 0 and 0.5. Figure 3-D also shows the linear relation of parameter a and PT.

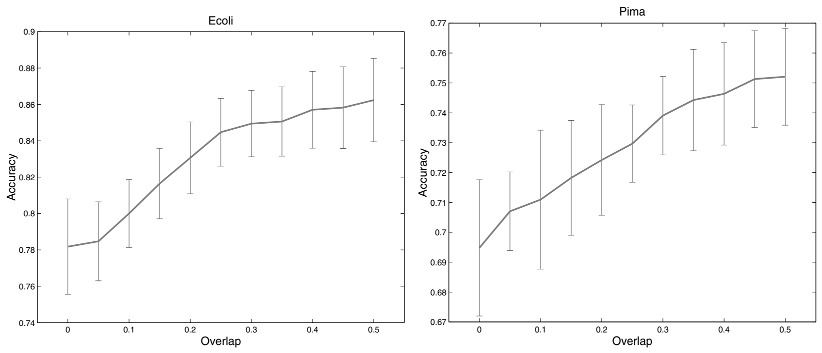


Fig. 4. Monotonically increased performance for linguistic decision trees with increasing degrees of fuzziness.

As we can see from these two figures, the performance these two datesets are roughly monotonic increased with the increase of PT. It implies that more fuzziness tends to increase the robustness of the LDT model and get better performance. From all the results, we can see that LDTs with fuzzy labels generally outperform the ones with discrete labels (where $PT = 0$). Due to the page limit, we cannot put all the results but they are available in (Qin and Lawry, 2007). Therefore, in summary, for the case of LDT model, we can say that fuzziness will bring greater performance. The increase is almost monotonically. But the optimal overlapping degrees are depends on the dataset you tested.

3.3 Linguistic Constraints

Here we assume that the linguistic constraints take the form of $\theta = \langle x_1$ is $\theta_1, \ldots, x_n$ is $\theta_n \rangle$, where θ_j represents a label expression based on $\mathcal{L}_j : j = 1, \ldots, n$. Consider the vector of linguistic constraint $\boldsymbol{\theta} = \langle \theta_1, \cdots, \theta_n \rangle$, where θ_j is the linguistic constraints on attribute j. We can evaluate a probability value for class C_t conditional on this information using a given linguistic decision tree as follows. The mass assignment given a linguistic constraint θ is evaluated by

$$\forall F_j \in \mathcal{F}_j \quad m_{\theta_j}(F_j) = \begin{cases} \frac{pm(F_j)}{\sum_{F_j \in \lambda(\theta_j)} pm(F_j)} & if : F_j \in \lambda(\theta_j) \\ 0 & otherwise \end{cases} \tag{15}$$

where $pm(F_j)$ is the prior mass for focal elements $F_j \in \mathcal{F}_j$ derived from the prior distribution $p(x_j)$ on Ω_j as follows:

$$pm(F_j) = \int_{\Omega_j} m_x(F_j) p(x_j) dx_j \tag{16}$$

Usually, we assume that $p(x_j)$ is the uniform distribution over Ω_j so that

$$pm(F_j) \propto \int_{\Omega_j} m_x(F_j) dx_j \tag{17}$$

For branch B with s nodes, the probability of B given $\boldsymbol{\theta}$ is evaluated by

$$P(B|\boldsymbol{\theta}) = \prod_{r=1}^{|B|} m_{\theta_{j_r}}(F_{j_r}) \tag{18}$$

and therefore, by Jeffrey's rule (Jeffrey, 1965)

$$P(C_t|\boldsymbol{\theta}) = \sum_{v=1}^{|LDT|} P(C_t|B_v) P(B_v|\boldsymbol{\theta}) \tag{19}$$

The methodology for classification under linguistic constraints allows us to fuse the background knowledge in linguistic form into classification. This is one of the advantages of using high-level knowledge representation language models such as label semantics.

3.4 Classification given fuzzy data

In previous sections LDTs have only been used to classify crisp data where objects are described in terms of precise attribute values. However, in many real-world applications limitations of measurement accuracy means that only imprecise values can be realistically obtained. In this section we introduce the idea of fuzzy data and show how LDTs can be used for classification in this context. Formally, a fuzzy database is defined to be a set of elements or objects each described by linguistic expressions rather than crisp values. In other words

$$\mathcal{FD} = \{\langle\theta_1(i), \ldots, \theta_n(i)\rangle : i = 1, \ldots, N\}$$

Currently there are very few benchmark problems of this kind with fuzzy attribute values. This is because, traditionally only crisp data values are recorded even in cases where this is inappropriate. Hence, we have generated a fuzzy database from a toy problem where the aim is to identify the interior of a figure of eight shape. Specifically, a figure of eight shape was generated according to the equation $x = 2^{(-0.5)}(sin(2t) - sin(t))$ and $y = 2^{(-0.5)}(sin(2t) + sin(t))$ where $t \in [0, 2\pi]$. (See figure 5). Points in $[-1.6, 1.6]^2$ are classified as legal if they lie within the 'eight' shape (marked with $\times$) and illegal if they lie outside (marked with points).

To form the fuzzy database we first generated a crisp database by uniformly sampling 961 points across $[-1.6, 1.6]^2$. Then each data vector $\langle x_1, x_2\rangle$ was converted to a vector of linguistic expressions $\langle\theta_1, \theta_2\rangle$ as follows: $\theta_j = \theta_{R_j}$ where $R_j = \{F \in \mathcal{F}_j : m_{x_j}(F) > 0\}$ A LDT was then learnt by applying the LID3 algorithm to the crisp database. This tree was then used to classify both the crisp and fuzzy data.

Suppose a LDT is trained on the 'eight' database where each attribute is discretized by five fuzzy sets uniformly: *verysmall* (*vs*), *small* (*s*),*medium* (*m*), *large* (*l*) and *verylarge* (*vl*). Further, suppose we are given the following description of data points:

$$\boldsymbol{\theta}_1 = \langle x \text{ is } vs \vee s \wedge \neg m, y \text{ is } vs \vee s \wedge \neg m\rangle$$
$$\boldsymbol{\theta}_2 = \langle x \text{ is } m \wedge l, y \text{ is } s \wedge m\rangle$$
$$\boldsymbol{\theta}_3 = \langle x \text{ is } s \wedge m, y \text{ is } l \vee vl\rangle$$

Experimental results obtained based on the approach introduced in 3.3 are as follows:

$$Pr(C_1|\boldsymbol{\theta}_1) = 1.000 \quad Pr(C_2|\boldsymbol{\theta}_1) = 0.000$$
$$Pr(C_1|\boldsymbol{\theta}_2) = 0.000 \quad Pr(C_2|\boldsymbol{\theta}_2) = 1.000$$
$$Pr(C_1|\boldsymbol{\theta}_3) = 0.428 \quad Pr(C_2|\boldsymbol{\theta}_3) = 0.572$$

As we can see from figure 5, the above 3 linguistic constraints roughly correspond to the area 1, 2 and 3, respectively. By considering the occurrence of legal and illegal examples within these areas, we can verify the correctness of our approach.

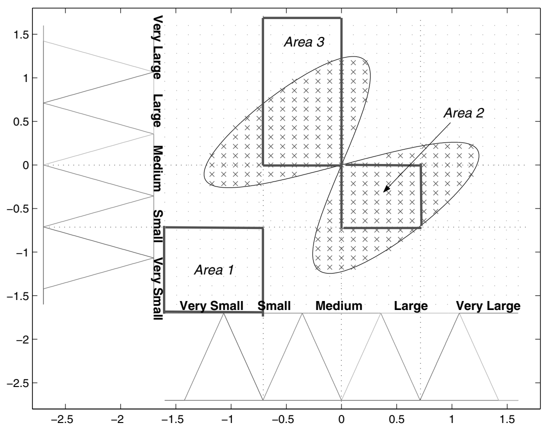


Fig. 5. Testing on the 'eight' problem with linguistic constraints $\boldsymbol{\theta}$, where each attribute is discretized by 5 trapezoidal fuzzy sets: *very small, small, medium, large and very large.*

3.5 Linguistic Decision Trees for Predictions

Consider a database for prediction:
$\mathcal{D} = \{\langle x_1(i), \cdots, x_n(i), x_t(i)\rangle \, | i = 1, \cdots, |\mathcal{D}|\}$
where $x_1, \cdots, x_n$ are potential explanatory attributes and x_t is the continuous target attribute. Unless otherwise stated, we use trapezoidal fuzzy sets with 50% overlap to discretized each continuous attribute individually (x_t) universe and assume the focal sets are $\mathcal{F}_1$, $\cdots$, $\mathcal{F}_n$ and $\mathcal{F}_t$. For the target attribute x_t: $\mathcal{F}_t = \{F_t^1, \cdots, F_t^{|\mathcal{F}_t|}\}$. For other attributes: x_j: $\mathcal{F}_j = \{F_j^1, \ldots, F_j^{|\mathcal{F}_j|}\}$. The inventive step is, to regard the focal elements for the target attribute as class labels. Hence, the LDT[4] model for prediction has the following form: A linguistic decision tree for prediction is a set of branches with associated probability distribution on the target focal elements of the following form:

$$\begin{aligned} LDT = \{&\langle B_1, P(F_t^1|B_1), \cdots, P(F_t^{|\mathcal{F}_t|}|B_1)\rangle, \cdots, \\ &\langle B_{|LDT|}, P(F_t^1|B_{|LDT|}), \cdots, P(F_t^{|\mathcal{F}_t|})|B_{|LDT|})\rangle\} \end{aligned}$$

where $F_t^1, \cdots, F_t^{|\mathcal{F}_t|}$ are the target focal elements (i.e. the focal elements for the target attribute or the output attribute).

$$P(F_t^j|\mathbf{x}) = \sum_{v=1}^{|LDT|} P(F_t^j|B_v)P(B_v|\mathbf{x}) \tag{20}$$

[4] We will use the same name 'LDT' for representing both linguistic decision trees (for classification) and linguistic prediction trees.

Given value $\mathbf{x} = \langle x_1, \cdots, x_n \rangle$ we need to estimate the target value $\widehat{x}_t$ (i.e. $\mathbf{x}_i \to \widehat{x}_t$). This is achieved by initially evaluating the probabilities on target focal elements: $P(F_t^1|\mathbf{x}), \cdots, P(F_t^{|\mathcal{F}_t|}|\mathbf{x})$ as described above. We then take the estimate of x_t, denoted $\widehat{x}_t$, to be the expected value:

$$\widehat{x}_t = \int_{\Omega_t} x_t \; p(x_t|\mathbf{x}) \; dx_t \tag{21}$$

where:

$$p(x_t|\mathbf{x}) = \sum_{j=1}^{|\mathcal{F}_t|} p(x_t|F_t^j) \; P(F_t^j|\mathbf{x}) \tag{22}$$

and

$$p(x_t|F_t^j) = \frac{m_{x_t}(F_t^j)}{\int_{\Omega_t} m_{x_t}(F_t^j) \; dx_t} \tag{23}$$

so that, we can obtain:

$$\widehat{x}_t = \sum_j \; P(F_t^j|\mathbf{x}) \; E(x_t|F_t^j) \tag{24}$$

where:

$$E(x_t|F_t^j) = \int_{\Omega_t} x_t \; p(x_t|F_t^j) \; dx_t = \frac{\int_{\Omega_t} x_t \; m_{x_t}(F_t^j) \; dx_t}{\int_{\Omega_t} m_{x_t}(F_t^j) \; dx_t} \tag{25}$$

We test our model on a toy problem of surface regression: 529 points were *uniformly* generated describing a surface defined by equation $z = sin(x \times y)$ where $x, y \in [0, 3]$. 2209 points are sampled uniformly as the test set. The attributes are discretized uniformly by fuzzy labels, the detailed results with different number of fuzzy labels are available in (Qin and Lawry, 2005c). We compared the prediction surface by the LDT model and the original surface in figure in 6. As we can see from the figures that these results are quite comparable though LDT didn't capture the small change at the tail. In this experiment, we use 7 fuzzy labels for discretization. If we use more labels, we can get the results as good as as we want, but it just needs more computational time.

4 Bayesian Estimation Based on Label Semantics

Bayesian reasoning provides a probabilistic approach to inference based on the Bayesian theorem. Given a test instance, the learner is asked to predict its class according to the evidence provided by the training data. The classification of unknown example $\mathbf{x}$ by Bayesian estimation is on the basis of the following probability,

$$P(C_k|\mathbf{x}) = \frac{P(\mathbf{x}|C_k)P(C_k)}{P(\mathbf{x})} \tag{26}$$

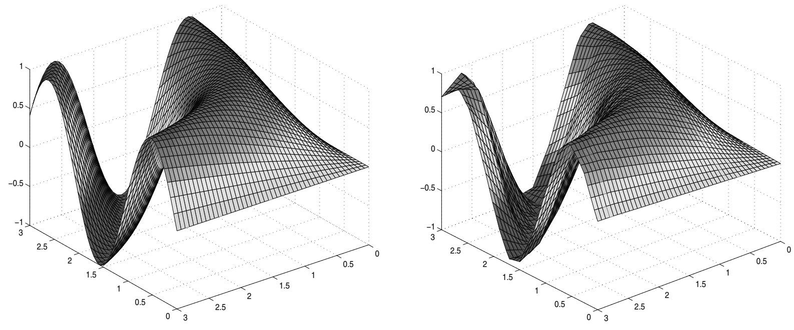


Fig. 6. Left-hand: the surface of $z = sin(x \times y)$. Right-hand: the prediction surface by linguistic decision trees.

Since the denominator in eq. 26 is invariant across classes, we can consider it as a normalization parameter. So, we obtain:

$$P(C_k|\mathbf{x}) \propto P(\mathbf{x}|C_k)P(C_k) \tag{27}$$

Now suppose we assume for each variable x_j that its outcome is independent of the outcome of all other variables given class C_k. In this case we can obtain the so-called naive Bayes classifier as follows:

$$P(C_k|\mathbf{x}) \propto \prod_{j=1}^{n} P(x_j|C_k)P(C_k) \tag{28}$$

where $P(x_j|C_k)$ is often called the likelihood of the data x_j given C_k. For a qualitative attribute, it can be estimated from corresponding frequencies. For a quantitative attribute, either probability density estimation or discretization can be employed to estimate its probabilities.

4.1 Fuzzy Naive Bayes

In label semantics framework, suppose we are given focal set $\mathcal{F}_j$ for each attribute j. Assuming that attribute x_j is numeric with universe Ω_j, then the likelihood of x_j given C_k can be represented by a density function $p(x_j|C_k)$ determine from the database $\mathcal{D}_k$ and prior density according to Jeffrey's rule (Jeffrey, 1965).

$$p(x_j|C_k) = \sum_{F \in \mathcal{F}_j} p(x_j|F)P(F|C_k) \tag{29}$$

From Bayes theorem, we can obtain:

$$p(x_j|F) = \frac{P(F|x_j)p(x_j)}{P(F)} = \frac{m_{x_j}(F)p(x_j)}{pm(F)} \tag{30}$$

where,

$$pm(F) = \int_{\Omega_j} P(F|x_j)p(x_j)dx_j = \frac{\sum_{\mathbf{x}\in\mathcal{D}} m_{x_j}(F)}{|\mathcal{D}|} \tag{31}$$

Substituting equation 30 in equation 29 and re-arranging gives

$$p(x_j|C_k) = p(x_j) \sum_{F\in\mathcal{F}_j} m_{x_j}(F) \frac{P(F|C_k)}{pm(F)} \tag{32}$$

where $P(F|C_k)$ can be derived from $\mathcal{D}_k$ according to

$$P(F|C_k) = \frac{\sum_{\mathbf{x}\in\mathcal{D}_k} m_{x_j}(F)}{|\mathcal{D}_k|} \tag{33}$$

This model is called fuzzy Naive Bayes (FNB). If we weaken the independence assumption, we can obtain a fuzzy semi-Naive Bayes (FSNB). More details of FNB and FSNB can be found in (Randon and Lawry, 2006).

4.2 Fuzzy Semi-Naive Bayes

The main advantage of using Semi-Naive Bayes over Naive Bayes is that it allows us to solve non-decomposable problems such as XOR by weakening the independence assumption of Naive Bayes. However, in order to utilize Semi-Naive Bayes it is necessary to find effective groupings of attributes within which dependencies must be taken into account. In this chapter, we present and evaluate a number of heuristic search algorithms for finding such groups of attributes.

Given a set of attributes: $x_1, x_2, \cdots, x_n$, they are partitioned into subsets $S_1, \cdots, S_w$ where $w \geq n$ and for each S_i a joint mass assignment $m_{i,j}$ is determined as follows: suppose, w.l.o.g $S_i = \{x_1, \cdots, x_v\}$ then the join mass assignment is

$$\forall T_1 \times \cdots \times T_v \in 2^{LA_1} \times \cdots \times 2^{LA_v} \tag{34}$$

$$m_{i,j}(T_1, \cdots, T_v) = \frac{1}{|DB_j|} \sum_{k\in\mathcal{D}} \prod_{r=1}^{w} m_{r,j}(T_i : x_i \in S_r) \tag{35}$$

Hence the prototype describing C_j is defined as $\langle m_{i,j}, \cdots, m_{w,j} \rangle$. A prototype of this form naturally defines a joint mass assignment m_j on the whole cross product space $2^{LA_1} \times \cdots \times 2^{LA_n}$ conditional on C_j as follows:

$$\forall T_1 \times \cdots \times T_n \in 2^{LA_1} \times \cdots \times 2^{LA_n} m_j(T_1, \cdots, T_n) = \prod_{r=1}^{w} m_{r,j}(T_i : x_i \in S_r) \tag{36}$$

In this formulation we are encoding variable dependence within the variable groupings $S_i : i = 1, \cdots w$ and assuming independence between the groups.

In order to estimate classification probabilities given input vectors of rea attribute values we need a mechanism for mapping from mass assignments on label space onto density functions on attribute space.

Definition 8 (Conditional Density Given a Mass Assignment) *Let x be a variable into Ω with prior distribution $p(x)$, LA be a set of labels for x and m be a posterior mass assignment for the set of appropriate labels of x inferred from some database $\mathcal{D}$. Then the posterior distribution of x conditional on m is given by*

$$\forall x \in \Omega,\ p(x|m) = p(x) \sum_{S \subseteq LA} \frac{m(S)}{pm(S)} m_x(S) \tag{37}$$

where $pm(S)$ is the prior mass assignment generated by the prior distribution $p(x)$ according to

$$pm(S) = \int_{\Omega} m_x(S) p(x) dx \tag{38}$$

This definition is motivated by the following argument based on the theorem of total probability which for a mass assignment, describing variables x on Ω.

We now consider methods for finding attribute groupings that increase discrimination in the model. Two measures has been proposed in (Randon and Lawry, 2006):

Definition 9 (Importance Measure) *Let the joint mass assignment for S_i given C_j be denoted $m_{i,j}$. For any input vector S_i the probability of cloass C_j can be estimated using Bayes theorem where*

$$P(C_j|S_i) = \frac{p(S_i|m_{i,j})|C_j|}{p(S_i|m_{i,j})|C_j| + p(S_i|m_{i,\neg j})|C_{\neg j}|} \tag{39}$$

where $m_{i,\neg j}$ denotes the mass assignments for S_j given $\neg C_j$. The importance measured of group S_i for class C_j is then defined by

$$IM_j(S_i) = \frac{\sum_{k \in \mathcal{D}_j} P(C_j|S_i(k))}{\sum_{k \in \mathcal{D}} P(C_j|S_i(k))} \tag{40}$$

Effectively, $IM_j(S_i)$ is a measure of the importance of the set of variables S_i as discriminators of C_j from the other classes.

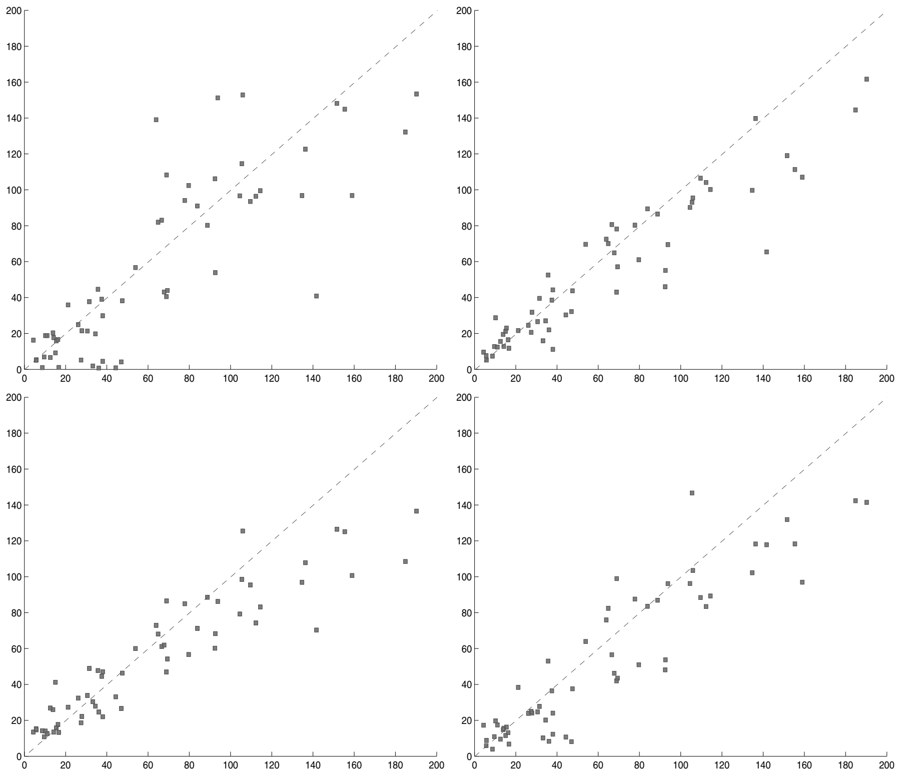


Fig. 7. Scatter plot showing original data verses prediction data on sunspot prediction problems. Upper left: Fuzzy Naive Bayes; upper right: Support Vector Regression; lower left: non-merged LDT with 5 fuzzy labels; lower right: Semi-naive Bayes.

Definition 10 (Correlation Measure) *Let $\mathcal{F}_1$ be the focal sets for S_1 and $\mathcal{F}_2$ the focal sets for S_2. Now let $m_{1,2,j}$ be the joint mass of $S_1 \cup S_2$ given C_j*

$$C(S_1, S_2) = \sqrt{\frac{1}{|\mathcal{F}_1||\mathcal{F}_1|}\sum_{R\subseteq\mathcal{F}_1}\sum_{T\subseteq\mathcal{F}_2}(m_{1,2,j}(R,T) - m_{1,j}(R)m_{2,j}(T))^2} \quad (41)$$

Here a threshold must be used to determine whether attributes should be grouped. The nearer the correlation measure gets to 1 the higher the correlation between attribute groups.

We tested our models with a real-world problem taken from the Time Series Data Library (Hyndman and Akram, 2007) and contains data of sunspot numbers between the years 1700-1979. The input attributes are x_{T-12} to x_{T-1}

(the data for previous 12 years) and the output (target) attribute is x_T, i.e. one-year-ahead. The experimental results for LID3, Fuzzy Naive Bayes, Semi-Naive Bayes and ε-SVR (Gunn, 1998) are compared in figure 7. We can see the results are quite comparable. In these graphs, for an error free prediction all points will fall on the line defined by $y = x$. Roughly, from the illustration, we can see that SVR and non-merged LDT have better performance, because predicted values distributed closer to $y = x$ than other two models.

4.3 Hybrid Bayesian Estimation Tree

Based on previous two linguistic models, a hybrid model was proposed in (Qin and Lawry, 2005a). Given a decision tree T is learnt from a training database $\mathcal{D}$. According to the Bayesian theorem: A data element $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$ can be classified by:

$$P(C_k|\mathbf{x}, T) \propto P(\mathbf{x}|C_k, T)P(C_k|T) \tag{42}$$

We can then divide the attributes into 2 disjoint groups denoted by $\mathbf{x}_T = \{x_1, \cdots, x_m\}$ and $\mathbf{x}_B = \{x_{m+1}, \cdots, x_n\}$, respectively. $\mathbf{x}_T$ is the vector of the variables that are contained in the given tree T and the remaining variables are contained in $\mathbf{x}_B$. Assuming conditional independence between $\mathbf{x}_T$ and $\mathbf{x}_B$ we obtain:

$$P(\mathbf{x}|C_k, T) = P(\mathbf{x}_T|C_k, T)P(\mathbf{x}_B|C_k, T) \tag{43}$$

Because $\mathbf{x}_B$ is independent of the given decision tree T and if we assume the variables in $\mathbf{x}_B$ are independent of each other given a particular class, we can obtain:

$$P(\mathbf{x}_B|C_k, T) = P(\mathbf{x}_B|C_k) = \prod_{j \in \mathbf{x}_B} P(x_j|C_k) \tag{44}$$

Now consider $\mathbf{x}_T$. According to Bayes theorem,

$$P(\mathbf{x}_T|C_k, T) = \frac{P(C_k|\mathbf{x}_T, T)P(\mathbf{x}_T|T)}{P(C_k|T)} \tag{45}$$

Combining equation 43, 44 and 45:

$$P(\mathbf{x}|C_k, T) = \frac{P(C_k|\mathbf{x}_T, T)P(\mathbf{x}_T|T)}{P(C_k|T)} \prod_{j \in \mathbf{x}_B} P(x_l|C_k) \tag{46}$$

Combining equation 42 and 46

$$P(C_k|\mathbf{x}, T) \propto P(C_k|\mathbf{x}_T, T)P(\mathbf{x}_T|T) \prod_{j \in \mathbf{x}_B} P(x_j|C_k) \tag{47}$$

Further, since $P(\mathbf{x}_T|T)$ is independent from C_k, we have that:

$$P(C_k|\mathbf{x}, T) \propto P(C_k|\mathbf{x}_T, T) \prod_{j \in \mathbf{x}_B} P(x_j|C_k) \tag{48}$$

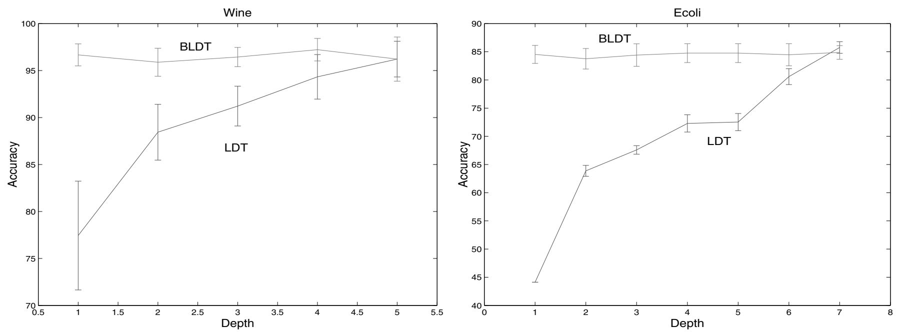


Fig. 8. Results for single LDT with Bayesian estimation: average accuracy with standard deviation on each dataset against the depth of the tree.

where $P(x_j|C_k)$ is evaluated according to eq. 32 and $P(C_k|\mathbf{x}_T, T)$ is just the class probabilities evaluated from the decision tree T according to equation 9.

We tested this new model with a set of UCI (UCI, 2007) data sets. Figure 8 is a simple result. More results are available in (Qin and Lawry, 2005a). From figures 8, we can see that the BLDT model generally performs better at shallow depths than LDT model. However, with the increasing of the tree depth, the performance of the BLDT model remains constant or decreases, while the accuracy curves for LDT increase. The basic idea of using Bayesian estimation given a LDT is to use the LDT as one estimator and the rest of the attributes as other independent estimators. Consider the two extreme cases for eq. 48. If all the attributes are used in building the tree (i.e. $\mathbf{x}_T = \mathbf{x}$), the probability estimations are from the tree only, that is:

$$P(C_k|\mathbf{x}, T) \propto P(C_k|\mathbf{x}_T, T)$$

If none of the attributes are used in developing the tree (i.e. $\mathbf{x} = \mathbf{x}_B$), the probability estimation will become:

$$P(C_k|\mathbf{x}, T) \propto \prod_{j \in \mathbf{x}_B} P(x_j|C_k)$$

which is simply a Naive Bayes classifier.

4.4 Bayesian Estimation From a Set of Trees

Given a training dataset, a small-sized tree (usually the depth is less than 3) can be learnt based on the method we discussed in section 3. We then learn another tree with the same size based on the remaining attributes, i.e., the attributes which have not been used in previous trees. In this manner, a set of trees can successively be built from training set. We denote this set of trees by $\mathcal{T} = \langle T_1, \ldots, T_W \rangle$ and where the set of attributes $\mathbf{x}_{T_w}$ for $w = 1, \ldots, W$ for a partition of $\{x_1, \ldots, x_n\}$ (see fig. 9 for a schematic illustration). For a given unclassified data element $\mathbf{x}$, we can partition it into W groups of disjoint set of attributes $\langle \mathbf{x}_{T_1}, \ldots, \mathbf{x}_{T_W} \rangle$. If we assume:

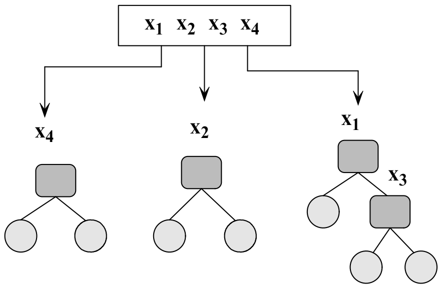


Fig. 9. An schematic illustration of Bayesian estimation from a set of linguistic decision trees.

$$P(C_t|\mathbf{x}) = P(C_t|\mathbf{x}_{T_1}, \ldots, \mathbf{x}_{T_W}) \approx P(C_t|T_1, \ldots, T_W) \tag{49}$$

Then, according to the Bayes theorem:

$$P(C_t|\mathcal{T}) = P(C_t|T_1, \ldots, T_W) = \frac{P(T_1, \ldots, T_W|C_t)P(C_t)}{P(T_1, \ldots, T_W)} \tag{50}$$

Assuming that the trees are generated independently then it is reasonable to assume that the groups of attributes are conditionally independent of each other. Hence,

$$P(T_1, \ldots, T_W|C_t) = \prod_{w=1}^{W} P(T_w|C_t) \tag{51}$$

For a particular tree T_w for $w = 1, \ldots, W$, we have

$$P(T_w|C_t) = \frac{P(C_t|T_w)P(T_w)}{P(C_t)} \tag{52}$$

So that,

$$\prod_{w=1}^{W} P(T_w|C_t) = \frac{\prod_{w=1}^{W} P(C_t|T_w) \prod_{i=1}^{W} P(T_w)}{P(C_t)^W} \tag{53}$$

Combining eq. 50, 51 and 53, we obtain

$$P(C_t|\mathcal{T}) \propto \frac{\prod_{w=1}^{W} P(C_t|T_w) \prod_{w=1}^{W} P(T_w)}{P(C_t)^{W-1}} \tag{54}$$

Since $\prod_{w=1}^{W} P(T_w)$ is independent from C_t, we finally obtain:

$$P(C_t|\mathcal{T}) \propto \frac{\prod_{w=1}^{W} P(C_t|T_w)}{P(C_t)^{W-1}} \tag{55}$$

where $P(C_t|T_w)$ is evaluated according to eq. 9.

5 Linguistic Rule Induction

The use of high-level knowledge representation in data modelling allows for enhanced transparency in the sense that the inferred models can be understood by practioners who are not necessarily experts in the formal representation framework employed. Rule based systems inherently tend to be more transparent than other models such as neural networks. A set of concise understandable rules can provide a better understanding of how the classification or prediction is made. Generally, there are two general types of algorithms for rule induction, *top down* and *bottom up* algorithms. Top-down approaches start from the most general rule and specialize it gradually. Bottom-up methods star from a basic fact given in training database and generalize it. In this paper we will focus on a top-down model for generating linguistic rules based on Quinlan's *First-Order Inductive Learning* (FOIL) Algorithm (Quinlan, 1990).

The FOIL algorithm is based on classical binary logic where typically attributes are assumed to be discrete. Numerical variables are usually discretized by partitioning the numerical domain into a finite number of intervals. However, because of the uncertainty involved in most real-world problems, sharp boundaries between intervals often lead to a loss of robustness and generality. Fuzzy logic has been used to solve the problem of sharp transitions between two intervals. Fuzzy rule induction research has been popular in both fuzzy and machine learning communities as a means to learning robust transparent models. Many algorithms have been proposed including simple fuzzy logic rule induction (Baldwin and Xie, 2004), fuzzy association rule mining (Xie, 2005) and first-order fuzzy rule induction based on FOIL (Drobics *et al.*, 2003,Prade *et al.*, 2003). In this paper, we will focus on an extension to the FOIL algorithm based on label semantics.

5.1 Generalized Appropriateness Measures

Based on definition 5, we can evaluate the appropriateness degree of $\theta \in LE$ is to aggregate the values of m_x across $\lambda(\theta)$. This motivates the following general definition of appropriateness measures.

Definition 11 (*Appropriateness Measures*) *$\forall \theta \in LE$, $\forall x \in \Omega$ the measure of appropriateness degrees of θ as a description of x is given by:*

$$\mu_\theta(x) = \sum_{S \in \lambda(\theta)} m_x(S)$$

Appropriateness degrees (def. 2) introduced at the beginning of this chapter are only a special case of the appropriateness measures where $\theta = L$ for $L \in \mathcal{L}$.

Given a continuous variable x: $\mathcal{L} = \{small,\ medium,\ large\}$, $\mathcal{F} = \{\{small\},\ \{small,\ medium\},\ \{medium\},\ \{medium, large\},\ \{large\}\}$. Suppose we are told that "x is **not large** but it is **small or medium**". This constraint can be interpreted as the logical expression

$$\theta = \neg large \wedge (small \vee medium)$$

According to definition 5, the possible label sets of the given logical expression θ are calculated as follows:

$$\lambda(\neg large) = \{\{small\}, \{small, medium\}, \{medium\}\}$$

$$\lambda(small) = \{\{small\}, \{small, medium\}\}$$

$$\lambda(medium) = \{\{small, medium\}, \{medium\}, \{medium, large\}\}$$

So that we can obtain:

$\lambda(\theta) = \lambda(\neg large \wedge (small \vee medium)) = \{\{small\},\ \{small,\ medium\},\ \{medium\}\} \wedge (\{\{small\}, \{small, medium\}\} \vee \{\{small, medium\},\ \{medium\},\ \{medium,\ large\}\}) = \{\{small\},\ \{small,\ medium\},\ \{medium\}\}$

If a prior distribution on focal elements of variable x are given as follows:

$\{small\} : 0.1, \{small, med.\} : 0.3, \{med.\} : 0.1, \{med., large\} : 0.5, \{large\} : 0.0$

The appropriateness measure for $\theta = \neg large \wedge (small \vee medium)$ is:

$$\begin{aligned}\mu_\theta(x) &= \sum_{S \in \lambda(\theta)} m_x(S) \\ &= m_x(\{small\}) + m_x(\{small, medium\}) + m_x(\{medium\}) \\ &= 0.1 + 0.3 + 0.1 = 0.5\end{aligned}$$

5.2 Linguistic Rules in Label Semantics

In sections 2 and 3, a basic introduction of label semantics is given and how it can be used for data modelling is discussed. In this section, we will describe a linguistic rule induction model based on label semantics. Now, we begin by clarifying the definition of a linguistic rule. Based on def. 4, a linguistic rule is a rule can be represented as a multi-dimensional logical expressions of fuzzy labels.

Definition 12 (*Multi-dimensional Logical Expressions of Labels*) *$MLE^{(n)}$ is the set of all multi-dimensional label expressions that can be generated from the logical label expression LE_j: $j = 1, \ldots, n$ and is defined recursively by:*

(i) If $\theta \in LE_j$ for $j = 1, \ldots, n$ then $\theta \in MLE^{(n)}$
(ii) If $\theta, \varphi \in MLE^{(n)}$ then $\neg\theta$, $\theta \wedge \varphi$, $\theta \vee \varphi$, $\theta \rightarrow \varphi \in MLE^{(n)}$

Any n-dimensional logical expression θ identifies a subset of $2^{\mathcal{L}_1} \times \ldots \times 2^{\mathcal{L}_n}$, denoted $\lambda^{(n)}(\theta)$, constraining the cross product of logical descriptions on each variable: $D_{x_1} \times \ldots \times D_{x_1}$. In such a way the imprecise constraint θ on n variables can be interpret as the precise constraint $D_{x_1} \times \ldots \times D_{x_1} \in \lambda^{(n)}(\theta)$

Given a particular data, how can we evaluated if a linguistic rule is appropriate for describing it? Based on the one-dimensional case, we now extend the concepts of appropriateness degrees to the multi-dimensional case as follows:

Definition 13 (*Multi-dimensional Appropriateness Degrees*) *Given a set of n-dimensional label expressions $MLE^{(n)}$:*

$$\forall\ \theta \in MLE^{(n)}, \forall x_j \in \Omega_j : j = 1, \cdots, n$$

$$\begin{aligned}\mu_\theta^n(\mathbf{x}) &= \mu_\theta^n(x_1, \cdots, x_n) = \sum_{\langle F_1, \cdots, F_n\rangle \in \lambda^{(n)}(\theta)} (F_1, \cdots, F_n) \\ &= \sum_{\langle F_1, \cdots, F_n\rangle \in \lambda^{(n)}(\theta)} \prod_{j=1}^{n} m_{x_j}(F_j)\end{aligned}$$

The appropriateness degrees in one-dimension are for evaluating a single label for describing a single data element, while in multi-dimensional cases they are for evaluating a linguistic rule for describing a data vector.

Consider a modelling problem with two variables x_1 and x_2 for which $\mathcal{L}_1 = \{$*small* (s), *medium* (med), *large*$(lg)\}$ and $\mathcal{L}_2 = \{$*low*(lo), *moderate* (mod), *high*$(h)\}$. Also suppose the focal elements for $\mathcal{L}_1$ and $\mathcal{L}_2$ are:

$$\mathcal{F}_1 = \{\{s\}, \{s, med\}, \{med\}, \{med, lg\}, \{lg\}\}$$

$$\mathcal{F}_2 = \{\{lo\}, \{lo, mod\}, \{mod\}, \{mod, h\}, \{h\}\}$$

According to the multi-dimensional generalization of definition 5 we have that

$$\begin{aligned}\lambda^{(2)}((med \wedge \neg s) \wedge \neg lo) &= \lambda^{(2)}(med \wedge \neg s) \cap \lambda^{(2)}(\neg lo) \\ &= \lambda(med \wedge \neg s) \times \lambda(\neg lo)\end{aligned}$$

Now, the set of possible label sets is obtained according to the λ-function:

$$\lambda(med \wedge \neg s) = \{\{med\}, \{med, lg\}\}$$

$$\lambda(\neg lo) = \{\{mod\}, \{mod, h\}, \{h\}\}$$

Hence, based on def. 5 we can obtain:

$$\lambda^{(2)}((med \wedge \neg s) \wedge \neg lo) = \{\langle\{med\}, \{mod\}\rangle, \langle\{med\}, \{mod, h\}\rangle,$$

$$\langle\{med\}, \{h\}\rangle, \langle\{med, lg\}, \{mod\}\rangle, \langle\{med, lg\}, \{mod, h\}\rangle, \langle\{med, lg\}, \{h\}\rangle\}$$

The above calculation on random set interpretation of the given rule based on λ-function is illustrated in fig. 10: given focal set $\mathcal{F}_1$ and $\mathcal{F}_2$, we can construct a 2-dimensional space where the focal elements have corresponding focal cells. Representation of the multi-dimensional λ-function of the logical expression of the given rule are represented by grey cells.

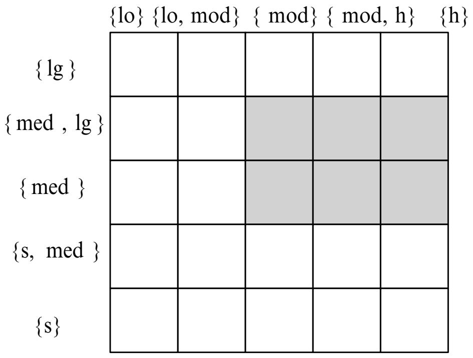


Fig. 10. Representation of the multi-dimensional λ-function of the logical expression $\theta = (med \wedge \neg\ s) \wedge \neg lo$ showing the focal cells $\mathcal{F}_1 \times \mathcal{F}_2$.

Given $\mathbf{x} = \langle x_1, x_2\rangle = \langle x_1 = \{med\} : 0.6, \{med, lg\} : 0.4\rangle$, $\langle x_2 = \{lo, mod\} :$ 0.8, $\{mod\} : 0.2\rangle$, we obtain:

$$\mu_\theta(\mathbf{x}) = (m(\{med\}) + m(\{med, lg\})) \times (m(\{mod\}) + m(\{mod, h\}) + m(\{h\}))$$

$$= (0.6 + 0.4) \times (0.2 + 0 + 0) = 0.2$$

And according to def. 5:

$$\mu^n_{\neg\theta}(\mathbf{x}) = 1 - \mu_\theta(\mathbf{x}) = 0.8$$

In another words, we can say that the linguistic expression θ covers the data $\mathbf{x}$ to degree 0.2 and θ can be considered as a linguistic rule. This interpretation of appropriateness is highlighted in next section on rule induction.

5.3 Information Heuristics for LFOIL

In the last section, we have shown how to evaluate the appropriateness of using a linguistic rule to describe a data vector. In this section, a new algorithm for learning a set of linguistic rules is proposed based on the FOIL algorithm (Quinlan, 1990), it is referred to as *Linguistic FOIL* (LFOIL). Generally, the heuristics for a rule learning model are for assessing the usefulness of a literal as the next component of the rule. The heuristics used for LFOIL are similar but modified from the FOIL algorithm (Quinlan, 1990) so as to incorporate linguistic expressions based on labels semantics. Consider a classification rule of the form:

$$R_i = \theta \rightarrow C_k \quad where \quad \theta \in MLE^{(n)}$$

Given a data set $\mathcal{D}$ and a particular class C_k, the data belonging to class C_k are referred to as *positive examples* and the rest of them are *negative examples.* For the given rule R_i, the coverage of positive data is evaluated by

$$T_i^+ = \sum_{l \in \mathcal{D}_k} \mu_\theta(\mathbf{x}_l) \tag{56}$$

and the coverage of negative examples is given by

$$T_i^- = \sum_{l \in (\mathcal{D}-\mathcal{D}_k)} \mu_\theta(\mathbf{x}_l) \tag{57}$$

where $\mathcal{D}_k$ is the subset of the database which is consisted by the data belonging to class C_k. The information for the original rule R_i can by evaluated by

$$I(R_i) = -\log_2\left(\frac{T_i^+}{T_i^+ + T_i^-}\right) \tag{58}$$

Suppose we then propose to another label expression φ to the body of R_i to generate a new rule

$$R_{i+1} = \varphi \wedge \theta \rightarrow C_k$$

where $\varphi, \theta \in MLE^{(n)}$. By adding the new literal φ, the positive and negative coverage becomes:

$$T_{i+1}^+ = \sum_{l \in \mathcal{D}_k} \mu_{\theta\wedge\varphi}(\mathbf{x}_l) \tag{59}$$

$$T_{i+1}^- = \sum_{l \in (\mathcal{D}-\mathcal{D}_k)} \mu_{\theta\wedge\varphi}(\mathbf{x}_l) \tag{60}$$

Therefore, the information becomes,

$$I(R_{i+1}) = -\log_2\left(\frac{T_{i+1}^+}{T_{i+1}^+ + T_{i+1}^-}\right) \tag{61}$$

Then we can evaluate the information gain from adding expression φ by:

$$G(\varphi) = T_{i+1}^{+}(I(R_i) - I(R_{i+1})) \tag{62}$$

We can see that the measure of information gain consists of two components. T_{i+1}^{+} is the coverage of positive data by the new rule R_{i+1} and $(I(R_i) - I(R_{i+1}))$ is the increase of information. The probability of C_k given a linguistic rule R_i is evaluated by:

$$P(C_k|R_i) = \frac{\sum_{l \in \mathcal{D}_k} \mu_\theta(\mathbf{x}_l)}{\sum_{l \in \mathcal{D}} \mu_\theta(\mathbf{x}_l)} = \frac{T_i^{+}}{T_i^{+} + T_i^{-}} \tag{63}$$

when $P(C_k|R_{i+1}) > P(C_k|R_i)$ (i.e., by appending a new literal, more positive examples are covered), we can obtain that $(I(R_i) - I(R_{i+1})) > 0$. By choosing a literal φ with maximum G value, we can form the new rule which covers more positive examples and thus increasing the accuracy of the rule.

5.4 Linguistic FOIL

We define a prior knowledge base $KB \subseteq MLE^{(n)}$ and a probability threshold $PT \in [0, 1]$. KB consists of fuzzy label expressions based on labels defined on each attribute. For example, given fuzzy labels $\{small_1 \; large_1\}$ to describe attribute 1 and $\{small_2 \; large_2\}$ to describe attribute 2. A possible knowledge base for the given two variables is: $KB = \{small_1, \neg small_1, large_1, \neg large_1, small_2, \neg small_2, large_2, \neg large_2\}$.

The idea for FOIL is as follows: from a general rule, we specify it by adding new literals in order to cover more positive and less negative examples according to the heuristics introduced in last section. After developing one rule, the positive examples covered by this rule are deleted from the original database. We then need to find a new rule based on this reduced database until all positive examples are covered. In this paper, because of the fuzzy linguistic nature of the expressions employed, typically data will be only partially covered by a given rule. For this reason we need a probability threshold PT as part of the decision process concerning rule coverage.

A pseudo-code of LFOIL are consists of two parts which are described follows:

Generating a Rule

- Let rule $R_i = \theta_1 \wedge \cdots \wedge \theta_d \rightarrow C_k$ be the rule at step i, we then find the next literal $\theta_{d+1} \in KB - \{\theta_1, \cdots, \theta_d\}$ for which $G(\theta_{d+1})$ is maximal.

- Replace rule R_i with $R_{i+1} = \theta_1 \wedge \cdots \wedge \theta_d \wedge \theta_{d+1} \rightarrow C_k$

- If $P(C_k|\theta_1 \wedge \cdots \wedge \theta_{i+1}) \geq PT$ then terminate else repeat.

Generating a Rule Base

Let $\Delta_i = \{\varphi_1 \to C_k, \cdots, \varphi_t \to C_k\}$ be the rule base at step i where $\varphi \in MLE$. We evaluate the coverage of Δ_i as follows:

$$CV(\Delta_i) = \frac{\sum_{l \in \mathcal{D}_k} \mu_{\varphi_1 \vee \cdots \vee \varphi_t}(\mathbf{x}_l)}{|\mathcal{D}_k|} \tag{64}$$

We define a coverage function $\delta : \Omega_1 \times \cdots \times \Omega_n \to [0,1]$ according to:

$$\begin{aligned}\delta(\mathbf{x}|\Delta_i) &= \mu_{\neg\Delta_i}(\mathbf{x}) = \mu_{\neg(\varphi_1 \vee \cdots \vee \varphi_t)}(\mathbf{x}) \\ &= 1 - \mu_{(\varphi_1 \vee \cdots \vee \varphi_t)}(\mathbf{x}) = 1 - \sum_{w=1}^{t} \mu_{R_w}(\mathbf{x})\end{aligned} \tag{65}$$

where $\delta(\mathbf{x}|\Delta_i)$ represents the degree to which $\mathbf{x}$ is *not* covered by a given rule base Δ_i. If CV is less than a predefined coverage threshold $CT \in [0,1]$:

$$CV(\Delta_i) < CT$$

then we generate a new rule for class C_k according to the above rule generation algorithm to form a new rule base Δ_{i+1} but where the entropy calculations are amended such that for a rule $R = \theta \to C_k$,

$$T^+ = \sum_{l \in \mathcal{D}_k} \mu_\theta(\mathbf{x}_l) \times \delta(\mathbf{x}_l|\Delta_i) \tag{66}$$

$$T^- = \sum_{l \in (\mathcal{D} - \mathcal{D}_k)} \mu_\theta(\mathbf{x}_l) \tag{67}$$

The algorithm terminates when $CV(RB_{i+1}) \geq CT$ or $CV(RB_{i+1}) - CV(RB_i) < \epsilon$ where $\epsilon \in [0,1]$ is a very small value, i.e., if there are no improvements in covering positive examples, we will stop the algorithm to avoid an infinite-loop calculation.

Given a rule base $\Delta_i = \{\varphi_1 \to C_k, \cdots, \varphi_t \to C_k\}$ and an unclassified data $\mathbf{x}$, we can estimate the probability of C_k, $P(C_k|\mathbf{x})$, as follows: Firstly, we determine the rule $R_{max} = \varphi_j \to C_k$ for which $\mu_{\varphi_j}(\mathbf{x})$ is maximal:

$$\varphi_j = \arg\max_{k \in \Delta_i} \mu_{\varphi_k} \tag{68}$$

Therefore, given the unclassified data $\mathbf{x}$, rule R_{max} is the most appropriate rule from the rule base we learned. For the rule $R_{max} \to C_k$ we evaluate two probabilities p_{max} and q_{max} where:

$$p_{max} = P(C_k|\varphi_j) \tag{69}$$

$$q_{max} = P(C_k|\neg\varphi_j) \tag{70}$$

We then use Jeffrey's rule (Jeffrey, 1965) to evaluate the class probability by:

$$P(C_k|\mathbf{x}) = p_{max} \times \mu_{\varphi_j}(\mathbf{x}) + q_{max} \times (1 - \mu_{\varphi_j}(\mathbf{x})) \tag{71}$$

We tested this rule learning algorithms with some toy problems and some real-world problems. Although it does not give us very good accuracy but we obtained some comparable performance to decision tree but with much better transparency. More details are available in (Qin and Lawry, 2005d).

6 Conclusions and Discussions

In this chapter, label semantics, a higher level knowledge representation language, was used for modeling imprecise concepts and building intelligent data mining systems. In particular, a number of linguistic data mining models have been proposed including: Linguistic Decision Trees (LDT) (for both classification and prediction), Bayesian estimation models (Fuzzy Naive Bayes, Semi-Naive Bayes, Bayesian Estimation Trees) and Linguistic Rule Induction (Linguistic FOIL).

Through previous empirical studies, we have shown that in terms of accuracy the linguistic decision tree model tends to perform significantly better than both C4.5 and Naive Bayes and has equivalent performance to that of the Back-Propagation neural networks (Qin and Lawry, 2005b). However, it is also the case that this model has much better transparency than other algorithms. Linguistic decision trees are suitable for both classification and prediction. Some benchmark prediction problems have been tested with the LDT model and we found that it has comparable performance to a number of state-of-art prediction algorithms such as support vector regression systems. Furthermore, a methodology for classification with linguistic constraints has been proposed within the label semantics framework.

In order to reduce complexity and enhance transparency, a forward merging algorithm has been proposed to merge the branches which give sufficiently similar probability estimations. With merging, the partitioning of the data space is re-constructed and more appropriate granules can be obtained. Experimental studies show that merging reduces the tree size significantly without a significant loss of accuracy. In order to obtain a better estimation, a new hybrid model combining the LDT model and Fuzzy Naive Bayes has been investigated. The experimental studies show that this hybrid model has comparable performance to LID3 but with much smaller trees. Finally, a FOIL based rule learning system has been introduced within label semantics framework. In this approach, the appropriateness of using a rule to describe a data element is represented by multi-dimensional appropriateness measures. Based on the FOIL algorithm, we proposed a new linguistic rule induction algorithm according to which we can obtain concise linguistic rules reflecting the underlying nature of the system.

It is widely recognized that most natural concepts have non-sharp boundaries. These concepts are vague or fuzzy, and one will usually only be willing to agree to a certain degree that an object belongs to a concept. Likewise, in machine learning and data mining, the patterns we are interested in are often vague and imprecise. To model this, in this chapter, we have discretized numerical attributes with fuzzy labels by which we can describe real values. Hence, we can use linguistic models to study the underlying relationships hidden in the data.

One of the distinctive advantages of linguistic models is that they allow for information fusion. In this chapter, we discussed methods for classification with linguistic constraints and classification for fuzzy data. Other information fusion methods are discussed in (Lawry, 2004). How to efficiently use background knowledge is an important challenge in machine learning. For example, Wang (Wang, 2004) argues that Bayesian learning has limitations in combining the prior knowledge and new evidence. We also need to consider the inconsistency between the background knowledge and new evidence. We believe that it will become a popular research topic in approximate reasoning.

Acknowledgements

The authors thank Prof Lotfi Zadeh for some insightful comments on this research. The first author also thanks Masoud Nikravesh, Marcus Thint, Ben Azvine and Trevor Martin for their interests in this research and support. The writing of this chapter is partly funded BT/BISC fellowship.

References

J.F. Baldwin, T.P. Martin and B.W. Pilsworth (1995), *Fril-Fuzzy and Evidential Reasoning in Artificial Intelligence.* John Wiley & Sons Inc, 1995.

J. F. Baldwin, J. Lawry and T.P. Martin (1997), Mass assignment fuzzy ID3 with applications. *Proceedings of the Unicom Workshop on Fuzzy Logic: Applications and Future Directions*, London pp. 278-294, 1997.

J. F. Baldwin and D. Xie (2004), Simple fuzzy logic rules based on fuzzy decision tree for classification and prediction problem, *Intelligent Information Processing II*, Z. Shi and Q. He (Ed.), Springer, 2004.

C. Blake and C.J. Merz (2007). UCI machine learning repository. `http://www.ics.uci.edu/~mlearn/MLRepository.html`, 2007.

M. Drobics, U. Bodenhofer and E. P. Klement (2003), FS-FOIL: an inductive learning method for extracting interpretable fuzzy descriptions, *International Journal of Approximate Reasoning*, 32: pp. 131-152, 2003.

S. R. Gunn (1998), Support vector machines for classification and regression. Technical Report of Dept. of Electronics and Computer Science, University of Southampton, 1998.

E. Hullermeier (2005), Fuzzy methods in machine learning and data mining: status and prospects, to appear in *Fuzzy Sets and Systems*, 2005.

R. Hyndman and M Akram (2005), Time series Data Library. Monash University, 2007.

C. Z. Janikow (1998), Fuzzy decision trees: issues and methods. *IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics*, Vol. 28, No. 1, 1998.

R. C. Jeffrey (1965), *The Logic of Decision*, Gordon & Breach Inc., New York, 1965.

J. Lawry, J. Shanahan, and A. Ralescu (2003), *Modelling with Words: Learning, fusion, and reasoning within a formal linguistic representation framework*. LNAI 2873, Springer-Verlag, 2003.

J. Lawry (2001), Label semantics: A formal framework for modelling with words. *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, LNAI 2143: pp. 374-384, Springer-Verlag, 2001.

J. Lawry (2004), A framework for linguistic modelling, *Artificial Intelligence*, 155: pp. 1-39, 2004.

J Lawry (2006), *Modelling and Reasoning with Vague Concepts*, Springer, 2006.

C. Olaru and L. Wehenkel (2003), A complete fuzzy decision tree technique. *Fuzzy Sets and Systems*. 138: pp.221-254, 2003.

Y. Peng, P. A. Flach (2001), Soft discretization to enhance the continuous decision trees. *ECML/PKDD Workshop: IDDM.*

H. Prade, G. Richard, and M. Serrurier (2003), Enriching relational learning with fuzzy predicates, N. Lavrac, et. al (Eds.): *Proceedings of PKDD*, LNAI 2838, pp. 399-410.

Z. Qin and J. Lawry (2004), A tree-structured model classification model based on label semantics, *Proceedings of the 10th International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU-04)*, pp. 261-268, Perugia, Italy.

Z. Qin and J. Lawry (2005a), Hybrid Bayesian estimation trees based on label semantics, L. Godo (Ed.), *Proceedings of Eighth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, Lecture Notes in Artificial Intelligence 3571, pp. 896-907, Springer.

Z. Qin and J. Lawry (2005b), Decision tree learning with fuzzy labels, *Information Sciences*, Vol. 172/1-2: pp. 91-129.

Z. Qin and J. Lawry (2005c), Prediction trees using linguistic modelling, *the Proceedings of International Fuzzy Association World Congress-05*, Beijing, China, September.

Z. Qin and J. Lawry (2005d), Linguistic rule induction based on a random set semantics, *the Proceedings of International Fuzzy Association World Congress-05*, Beijing, China.

Z. Qin and J. Lawry (2007), Fuzziness and performance: an empirical study with linguistic decision trees. To appear in IFSA-2007, Cuncun, Mexico.

J. R. Quinlan (1986), Induction of decision trees, *Machine Learning*, Vol 1: pp. 81-106.

J. R. Quinlan (1993), *C4.5: Programs for Machine Learning*, San Mateo: Morgan Kaufmann.

J. R. Quinlan (1990), Learning logical definitions from relations, *Machine Learning*, 5: 239-266.

N. J. Randon and J. Lawry (2006), Classification and query evaluation using modelling with words, Information Sciences, Special Issue - Computing with Words: Models and Applications, Vol. 176: pp 438-464.

Pei Wang (2004), The limitation of Bayesianism, Artificial Intelligence 158(1): pp. 97-106.

D. Xie (2005), Fuzzy associated rules discovered on effective reduced database algorithm, *Proceedings of IEEE-FUZZ*, pp. 779-784, Reno, USA, 2005.

L. A. Zadeh (1965), Fuzzy sets, *Information and Control*, Vol 8: pp. 338-353.

L. A. Zadeh (1996), Fuzzy logic = computing with words, *IEEE Transaction on Fuzzy Systems*. Vol. 4, No. 2: pp. 103-111.

L. A. Zadeh, Toward a perception-based theory of probabilistic reasoning with imprecise probabilities, Journal of Statistical Planning and Inference, Vol. 105: pp. 233264.

L.A. Zadeh (2003), Foreword for modelling with words, *Modelling with Words*, LNAI 2873, Ed., J. Lawry, J. Shanahan, and A.Ralescu, Springer.

L.A. Zadeh (2005), Toward a generalized theory of uncertainty (GTU) an outline, Information Sciences, Vol. 172/1-2, pp. 1-40.

Swarm Intelligence Algorithms for Data Clustering

Ajith Abraham[1], Swagatam Das[2], and Sandip Roy[3]

[1] Center of Excellence for Quantifiable Quality of Service (Q2S), Norwegian University of Science and Technology, Trondheim, Norway
ajith.abraham@ieee.org

[2] Department of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata 700032, India.

[3] Department of Computer Science and Engineering, Asansol Engineering College, Asansol-713304, India.

Summary. Clustering aims at representing large datasets by a fewer number of prototypes or clusters. It brings simplicity in modeling data and thus plays a central role in the process of knowledge discovery and data mining. Data mining tasks, in these days, require fast and accurate partitioning of huge datasets, which may come with a variety of attributes or features. This, in turn, imposes severe computational requirements on the relevant clustering techniques. A family of bio-inspired algorithms, well-known as Swarm Intelligence (SI) has recently emerged that meets these requirements and has successfully been applied to a number of real world clustering problems. This chapter explores the role of SI in clustering different kinds of datasets. It finally describes a new SI technique for partitioning any dataset into an optimal number of groups through one run of optimization. Computer simulations undertaken in this research have also been provided to demonstrate the effectiveness of the proposed algorithm.

1 Introduction

Clustering means the act of partitioning an unlabeled dataset into groups of similar objects. Each group, called a 'cluster', consists of objects that are similar between themselves and dissimilar to objects of other groups. In the past few decades, cluster analysis has played a central role in a variety of fields ranging from engineering (machine learning, artificial intelligence, pattern recognition, mechanical engineering, electrical engineering), computer sciences (web mining, spatial database analysis, textual document collection, image segmentation), life and medical sciences (genetics, biology, microbiology, paleontology, psychiatry, pathology), to earth sciences (geography. geology, remote sensing), social sciences (sociology, psychology, archeology, education), and economics (marketing, business) (Evangelou *et al.*, 2001, Lillesand and Keifer, 1994, Rao, 1971, Duda and Hart, 1973, Fukunaga, 1990, Everitt, 1993).

From a machine learning perspective, clusters correspond to the hidden patterns in data, the search for clusters is a kind of unsupervised learning, and the resulting system represents a *data concept*. The problem of data clustering has been approached from diverse fields of knowledge like statistics (multivariate analysis) (Forgy, 1965), graph theory (Zahn, 1971), expectation maximization algorithms (Mitchell, 1997), artificial neural networks (Mao and Jain, 1995, Pal *et al.*, 1993, Kohonen, 1995), evolutionary computing (Falkenauer, 1998, Paterlini and Minerva, 2003) and so on. Researchers all over the globe are coming up with new algorithms, on a regular basis, to meet the increasing complexity of vast real-world datasets. A comprehensive review of the state-of-the-art clustering methods can be found in (Xu and Wunsch, 2005) and (Rokach and Maimon, 2005).

Data mining is a powerful new technology, which aims at the extraction of hidden predictive information from large databases. Data mining tools predict future trends and behaviors, allowing businesses to make proactive, knowledge-driven decisions. The process of knowledge discovery from databases necessitates fast and automatic clustering of very large datasets with several attributes of different types (Mitra *et al.*, 2002). This poses a severe challenge before the classical clustering techniques. Recently a family of nature inspired algorithms, known as *Swarm Intelligence* (SI), has attracted several researchers from the field of pattern recognition and clustering. Clustering techniques based on the SI tools have reportedly outperformed many classical methods of partitioning a complex real world dataset.

Swarm Intelligence is a relatively new interdisciplinary field of research, which has gained huge popularity in these days. Algorithms belonging to the domain, draw inspiration from the collective intelligence emerging from the behavior of a group of social insects (like bees, termites and wasps). When acting as a community, these insects even with very limited individual capability can jointly (cooperatively) perform many complex tasks necessary for their survival. Problems like finding and storing foods, selecting and picking up materials for future usage require a detailed planning, and are solved by insect colonies without any kind of supervisor or controller. An example of particularly successful research direction in swarm intelligence is Ant Colony Optimization (ACO) (Dorigo *et al.*, 1996, Dorigo and Gambardella, 1997), which focuses on discrete optimization problems, and has been applied successfully to a large number of NP hard discrete optimization problems including the traveling salesman, the quadratic assignment, scheduling, vehicle routing, etc., as well as to routing in telecommunication networks. Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995) is another very popular SI algorithm for global optimization over continuous search spaces. Since its advent in 1995, PSO has attracted the attention of several researchers all over the world resulting into a huge number of variants of the basic algorithm as well as many parameter automation strategies.

In this Chapter, we explore the applicability of these bio-inspired approaches to the development of self-organizing, evolving, adaptive and autonomous clustering techniques, which will meet the requirements of next-generation data mining systems, such as diversity, scalability, robustness, and resilience. The next section of the chapter provides an overview of the SI paradigm with a special emphasis on two SI algorithms well-known as Particle Swarm Optimization (PSO) and Ant Colony Systems (ACS). Section 3 outlines the data clustering problem and briefly reviews the present state of the art in this field. Section 4 describes the use of the SI algorithms in both crisp and fuzzy clustering of real world datasets. A new automatic clustering algorithm, based on PSO, has been outlined in this Section. The algorithm requires no previous knowledge of the dataset to be partitioned, and can determine the optimal number of classes dynamically. The new method has been compared with two well-known, classical fuzzy clustering algorithms. The Chapter is concluded in Section 5 with possible directions for future research.

2 An Introduction to Swarm Intelligence

The behavior of a single ant, bee, termite and wasp often is too simple, but their collective and social behavior is of paramount significance. A look at National Geographic TV Channel reveals that advanced mammals including lions also enjoy social lives, perhaps for their self-existence at old age and in particular when they are wounded. The collective and social behavior of living creatures motivated researchers to undertake the study of today what is known as *Swarm Intelligence*. Historically, the phrase Swarm Intelligence (SI) was coined by Beny and Wang in late 1980s (Beni and Wang, 1989) in the context of cellular robotics. A group of researchers in different parts of the world started working almost at the same time to study the versatile behavior of different living creatures and especially the social insects. The efforts to mimic such behaviors through computer simulation finally resulted into the fascinating field of SI. SI systems are typically made up of a population of simple agents (an entity capable of performing/executing certain operations) interacting locally with one another and with their environment. Although there is normally no centralized control structure dictating how individual agents should behave, local interactions between such agents often lead to the emergence of global behavior. Many biological creatures such as fish schools and bird flocks clearly display structural order, with the behavior of the organisms so integrated that even though they may change shape and direction, they appear to move as a single coherent entity (Couzin *et al.*, 2002). The main properties of the collective behavior can be pointed out as follows and is summarized in Figure 1.

Homogeneity: every bird in flock has the same behavioral model. The flock moves without a leader, even though temporary leaders seem to appear.

Locality: its nearest flock-mates only influence the motion of each bird. Vision is considered to be the most important senses for flock organization.
Collision Avoidance: avoid colliding with nearby flock mates.
Velocity Matching: attempt to match velocity with nearby flock mates.
Flock Centering: attempt to stay close to nearby flock mates

Individuals attempt to maintain a minimum distance between themselves and others at all times. This rule is given the highest priority and corresponds to a frequently observed behavior of animals in nature (Krause and Ruxton, 2002). If individuals are not performing an avoidance maneuver they tend to be attracted towards other individuals (to avoid being isolated) and to align themselves with neighbors (Partridge and Pitcher, 1980, Partridge, 1982).

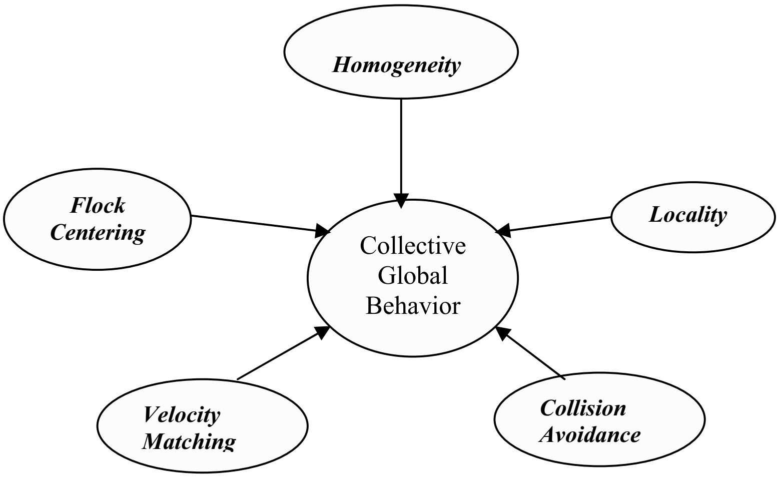


Fig. 1. Main traits of collective behavior

Couzin *et al.* identified four collective dynamical behaviors (Couzin *et al.*, 2002) as illustrated in Figure 2:

Swarm: an aggregate with cohesion, but a low level of polarization (parallel alignment) among members
Torus: individuals perpetually rotate around an empty core (milling). The direction of rotation is random.
Dynamic parallel group: the individuals are polarized and move as a coherent group, but individuals can move throughout the group and density and group form can fluctuate (Partridge and Pitcher, 1980, Major and Dill, 1978).
Highly parallel group: much more static in terms of exchange of spatial positions within the group than the dynamic parallel group and the variation in density and form is minimal.

As mentioned in (Grosan *et al.*, 2006) at a high-level, a swarm can be viewed as a group of agents cooperating to achieve some purposeful behavior and achieve some goal (Abraham *et al.*, 2006). This collective intelligence seems to emerge from what are often large groups:

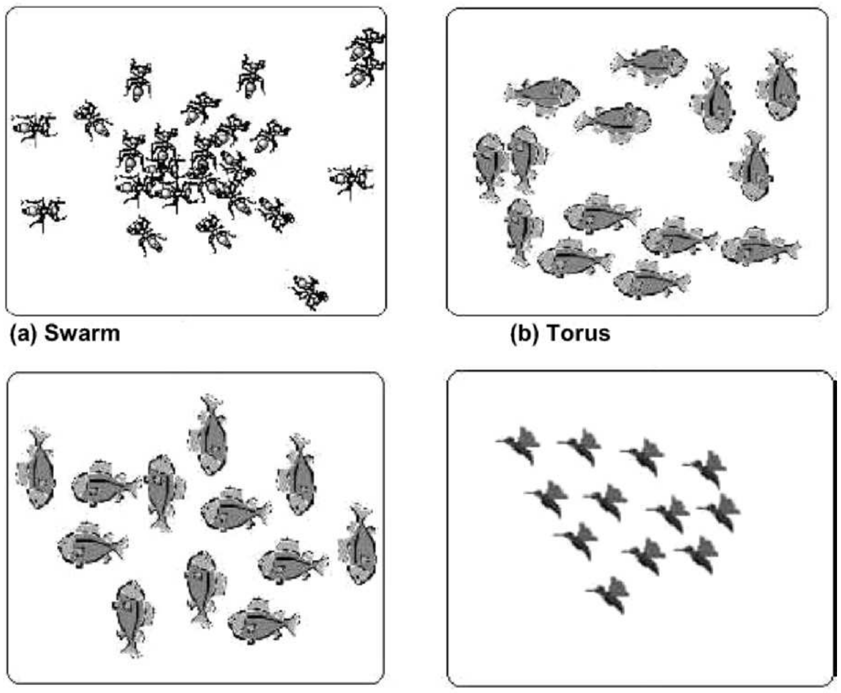


Fig. 2. Different models of collective behavior (Grosan *et al.*, 2006)

According to Milonas, five basic principles define the SI paradigm (Milonas, 1994). First is the the proximity principle: the swarm should be able to carry out simple space and time computations. Second is the quality principle: the swarm should be able to respond to quality factors in the environment. Third is the principle of diverse response: the swarm should not commit its activities along excessively narrow channels. Fourth is the principle of stability: the swarm should not change its mode of behavior every time the environment changes. Fifth is the principle of adaptability: the swarm must be able to change behavior mote when it is worth the computational price. Note that principles four and five are the opposite sides of the same coin. Below we discuss in details two algorithms from SI domain, which have gained wide popularity in a relatively short span of time.

2.1 The Ant Colony Systems

The basic idea of a real ant system is illustrated in Figure 4. In the left picture, the ants move in a straight line to the food. The middle picture illustrates the situation soon after an obstacle is inserted between the nest and the food. To avoid the obstacle, initially each ant chooses to turn left or right at random. Let us assume that ants move at the same speed depositing pheromone in the trail uniformly. However, the ants that, by chance, choose to turn left will reach the food sooner, whereas the ants that go around the obstacle turning right will follow a longer path, and so will take longer time to circumvent the obstacle. As a result, pheromone accumulates faster in the shorter path around the obstacle. Since ants prefer to follow trails with larger amounts of pheromone, eventually all the ants converge to the shorter path around the obstacle, as shown in Figure 3.

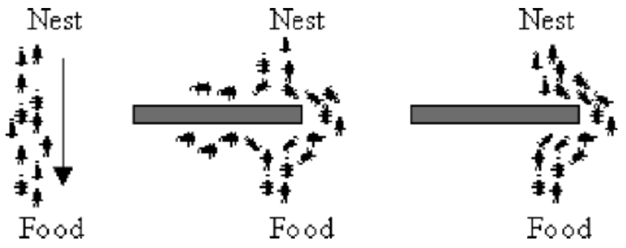


Fig. 3. Illustrating the behavior of real ant movements.

An artificial Ant Colony System (ACS) is an agent-based system, which simulates the natural behavior of ants and develops mechanisms of cooperation and learning. ACS was proposed by Dorigo *et al.* (Dorigo and Gambardella, 1997) as a new heuristic to solve combinatorial optimization problems. This new heuristic, called Ant Colony Optimization (ACO) has been found to be both robust and versatile in handling a wide range of combinatorial optimization problems.

The main idea of ACO is to model a problem as the search for a minimum cost path in a graph. Artificial ants as if walk on this graph, looking for cheaper paths. Each ant has a rather simple behavior capable of finding relatively costlier paths. Cheaper paths are found as the emergent result of the global cooperation among ants in the colony. The behavior of artificial ants is inspired from real ants: they lay pheromone trails (obviously in a mathematical form) on the graph edges and choose their path with respect to probabilities that depend on pheromone trails. These pheromone trails progressively decrease by evaporation. In addition, artificial ants have some extra features not seen in their counterpart in real ants. In particular, they live in a discrete world (a graph) and their moves consist of transitions from nodes to nodes.

Below we illustrate the use of ACO in finding the optimal tour in the classical Traveling Salesman Problem (TSP). Given a set of n cities and a set of distances between them, the problem is to determine a minimum traversal of the cities and return to the home-station at the end. It is indeed important to note that the traversal should in no way include a city more than once. Let $r \quad (C_x, C_y)$be a measure of cost for traversal from city C_x to C_y. Naturally, the total cost of traversing n cities indexed by $i_1, i_2, i_{3,} \ldots,\ i_n$ in order is given by the following expression:

$$Cost(i_1, i_2, ..., i_n) = \sum_{j=1}^{n-1} r(Ci_j, Ci_{j+1}) + r(Ci_n, Ci_1) \tag{1}$$

The ACO algorithm is employed to find an optimal order of traversal of the cities. Let τ be a mathematical entity modeling the pheromone and $\eta_{ij} =$ 1/r (i , j) is a local heuristic. Also let allowed$_k$(t) be the set of cities that are yet to be visited by ant k located in city$i$. Then according to the classical ant system (Everitt, 1993) the probability that ant k in city i visits city j is given by:

$$p_{ij}^k(t) = \begin{matrix} \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum\limits_{h \in allowed_k(t)} [\tau_{ih}(t)]^\alpha [\eta_{ih}]^\beta} & if\, h \in allowed_k(t) \\ 0 & otherwise \end{matrix} \tag{2}$$

In Equation 2 shorter edges with greater amount of pheromone are favored by multiplying the pheromone on edge $(i, j\ )$ by the corresponding heuristic value η*(i, j).* Parameters α (¿ 0) and β (¿ 0) determine the relative importance of pheromone versus cost. Now in ant system, pheromone trails are updated as follows. Let D_k be the length of the tour performed by ant k, $\Delta\tau_k$ (i , j)= 1/D$_k$ if $(i, j) \quad \in$ tour done by ant $k$and = 0 otherwise and finally let ρ $\in$ [0,1] be a pheromone decay parameter which takes care of the occasional evaporation of the pheromone from the visited edges. Then once all ants have built their tours, pheromone is updated on all the ages as,

$$\tau(i, j) = (1 - \rho).\tau(i, j) + \sum_{k=1}^{m} \Delta\tau_k(i, j) \tag{3}$$

From Equation (3), we can guess that pheromone updating attempts to accumulate greater amount of pheromone to shorter tours (which corresponds to high value of the second term in (3) so as to compensate for any loss of pheromone due to the first term). This conceptually resembles a reinforcement-learning scheme, where better solutions receive a higher reinforcement.

The ACO differs from the classical ant system in the sense that here the pheromone trails are updated in two ways. Firstly, when ants construct a tour they locally change the amount of pheromone on the visited edges by a local updating rule. Now if we let γ to be a decay parameter and $\Delta\tau$(i, j) = τ_0 such that τ_0 is the initial pheromone level, then the local rule may be stated as,

$$\tau(i,j) = (1-\gamma).\tau(i,j) + \gamma.\Delta\tau(i,j) \tag{4}$$

Secondly, after all the ants have built their individual tours, a global updating rule is applied to modify the pheromone level on the edges that belong to the best ant tour found so far. If κ be the usual pheromone evaporation constant, D_{gb} be the length of the globally best tour from the beginning of the trial and

$\Delta\tau^{/}$(i , j) = 1/ D_{gb} only when the edge (i, j) belongs to global-best-tour and zero otherwise, then we may express the global rule as follows:

$$\tau(i,j) = (1-\kappa).\tau(i,j) + \kappa.\Delta\tau^{/}(i,j) \tag{5}$$

The main steps of ACO algorithm are presented in Algorithm 1.

Algorithm 1: Procedure ACO

```
1: Initialize pheromone trails;
2: repeat {at this stage each loop is called an iteration}
3:    Each ant is positioned on a starting node
4:    repeat {at this level each loop is called a step}
5:       Each ant applies a state transition rule like rule (2) to incrementally build
         a solution and a local pheromone-updating rule like rule (4);
6:    until all ants have built a complete solution
7:    global pheromone-updating rule like rule (5) is applied.
8: until terminating condition is reached
```

2.2 The Particle Swarm Optimization (PSO)

The concept of Particle Swarms, although initially introduced for simulating human social behaviors, has become very popular these days as an efficient search and optimization technique. The Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995,Kennedy *et al.*, 2001), as it is called now, does not require any gradient information of the function to be optimized, uses only primitive mathematical operators and is conceptually very simple.

In PSO, a population of conceptual 'particles' is initialized with random positions X_i and velocities V_i, and a function, f, is evaluated, using the particle's positional coordinates as input values. In an n-dimensional search space, X_i = (x_{i1}, x_{i2}, x_{i3},...,x_{in}) and V_i= (v_{i1}, v_{i2}, v_{i3},...,v_{in}). Positions and velocities are adjusted, and the function is evaluated with the new coordinates at each time-step. The basic update equations for the d-th dimension of the i-th particle in PSO may be given as

$$\begin{aligned}
V_{id}(t+1) &= \omega.V_{id}(t) + C_1.\varphi_1.(P_{lid} - X_{id}(t)) + C_2.\varphi_2.(P_{gd} - X_{id}(t)) \\
X_{id}(t+1) &= X_{id}(t) + V_{id}(t+1)
\end{aligned} \tag{6}$$

The variables ϕ_1 and ϕ_2 are random positive numbers, drawn from a uniform distribution and defined by an upper limit $\phi_{max,}$which is a parameter of the system. C_1 and C_2 are called acceleration constants whereas ω is called inertia weight. P_{li} is the local best solution found so far by the i-th particle, while P_g represents the positional coordinates of the fittest particle found so far in the entire community. Once the iterations are terminated, most of the particles are expected to converge to a small radius surrounding the global optima of the search space. The velocity updating scheme has been illustrated in Figure 4 with a humanoid particle.

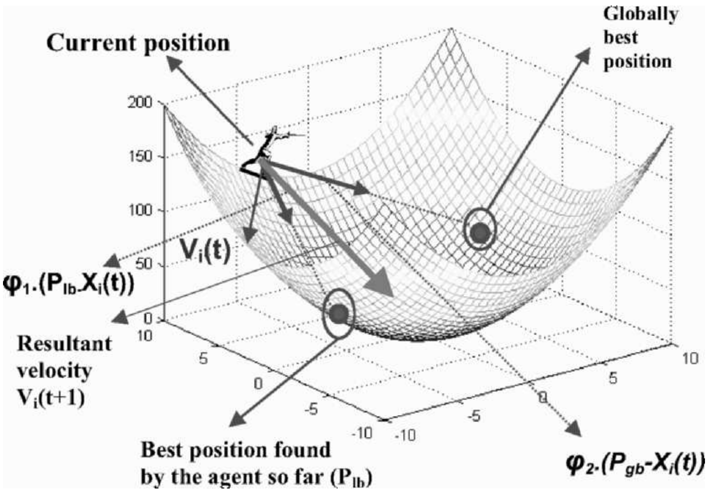

Current position
Globally best position
200
150
100
50
0
$V_i(t)$
$\varphi_1.(P_{lb}$-$X_i(t))$
Resultant velocity $V_i(t+1)$
10
5
0
-5
-10
-10
-5
0
5
10
Best position found by the agent so far (P_{lb})
$\varphi_2.(P_{gb}$-$X_i(t))$

Fig. 4. Illustrating the velocity updating scheme of basic PSO

A pseudo code for the PSO algorithm is presented in Algorithm 2.

3 Data Clustering – An Overview

In this section, we first provide a brief and formal description of the clustering problem. We then discuss a few major classical clustering techniques.

3.1 Problem Definition

A *pattern* is a physical or abstract structure of objects. It is distinguished from others by a collective set of attributes called *features*, which together represent

Algorithm 2: The PSO Algorithm

```
Input: Randomly initialized position and velocity of the particles: $\mathbf{X}_i(0)$ and $\mathbf{V}_i(0)$
Output: Position of the approximate global optima $\mathbf{X}^*$
1: while terminating condition is not reached do
2:    for $i = 1$ to $numberofparticles$ do
3:       Evaluate the fitness: $=f(\mathbf{X}_i(t))$;
4:       Update $\mathbf{P}(t)$and$\mathbf{g}(t)$;
5:       Adapt velocity of the particle using Equation 3;
6:       Update the position of the particle;
7:    end for
8: end while
```

a pattern (Konar, 2005). Let P = {P$_1$, P$_2$... P$_n$} be a set of n patterns or data points, each having d features. These patterns can also be represented by a profile data matrix $\mathbf{X}_{n\times d}$ having n d-dimensional row vectors. The i-th row vector $\mathbf{X}_i$ characterizes the i-th object from the set P and each element X$_{i,j}$ in $\mathbf{X}_i$ corresponds to the j-th real value feature ($j = 1, 2, ..., d$) of the i-th pattern (i =1,2,...., n). Given such an $\mathbf{X}_{n\times d,}a$ partitional clustering algorithm tries to find a partition C = {C$_1$, C$_2$,......, C$_K$} of K classes, such that the similarity of the patterns in the same cluster is maximum and patterns from different clusters differ as far as possible. The partitions should maintain the following properties:

1. Each cluster should have at least one pattern assigned i.e. $C_i \neq \Phi \forall i \in \{1, 2, ..., K\}$.
2. Two different clusters should have no pattern in common. i.e. $C_i \cap C_j = \Phi, \forall i \neq j$ and $i, j \in \{1, 2, ..., K\}$. This property is required for crisp (hard) clustering. In Fuzzy clustering this property doesn't exist.
3. Each pattern should definitely be attached to a cluster i.e. $\bigcup_{i=1}^{K} C_i = P$.

Since the given dataset can be partitioned in a number of ways maintaining all of the above properties, a fitness function (some measure of the adequacy of the partitioning) must be defined. The problem then turns out to be one of finding a partition $\mathbf{C}^*$ of optimal or near-optimal adequacy as compared to all other feasible solutions $\mathbf{C}$ = { C^1, C^2,........, C$^{N(n,K)}$} where,

$$N(n, K) = \frac{1}{K!} \sum_{i=1}^{K} (-1)^i \binom{K}{i}^i (K - i)^i \tag{7}$$

is the number of feasible partitions. This is same as,

$$\begin{matrix} Optimize f(X_{\mathrm{n\times d}}, C) \\ C \end{matrix} \tag{8}$$

where C is a single partition from the set **C** and f is a statistical-mathematical function that quantifies the goodness of a partition on the basis of the similarity measure of the patterns. Defining an appropriate similarity measure plays fundamental role in clustering (Jain *et al.*, 1999). The most popular way to evaluate similarity between two patterns amounts to the use of *distance measure.* The most widely used distance measure is the Euclidean distance, which between any two d-dimensional patterns $\mathbf{X}_i$ and $\mathbf{X}_j$ is given by,

$$d(\mathbf{X}_i, \mathbf{X}_j) = \sqrt{\sum_{p=1}^{d} (X_{i,p} - X_{j,p})^2} = \|\mathbf{X}_i - \mathbf{X}_j\| \tag{9}$$

It has been shown in (Brucker, 1978) that the clustering problem is NP-hard when the number of clusters exceeds 3.

3.2 The Classical Clustering Algorithms

Data clustering is broadly based on two approaches: *hierarchical* and *partitional* (Frigui and Krishnapuram, 1999, Leung *et al.*, 2000). Within each of the types, there exists a wealth of subtypes and different algorithms for finding the clusters. In hierarchical clustering, the output is a tree showing a sequence of clustering with each cluster being a partition of the data set (Leung *et al.*, 2000). Hierarchical algorithms can be agglomerative (bottom-up) or divisive (top-down). Agglomerative algorithms begin with each element as a separate cluster and merge them in successively larger clusters. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters. Hierarchical algorithms have two basic advantages (Frigui and Krishnapuram, 1999). Firstly, the number of classes need not be specified a priori and secondly, they are independent of the initial conditions. However, the main drawback of hierarchical clustering techniques is they are static, i.e. data-points assigned to a cluster can not move to another cluster. In addition to that, they may fail to separate overlapping clusters due to lack of information about the global shape or size of the clusters (Jain *et al.*, 1999).

Partitional clustering algorithms, on the other hand, attempt to decompose the data set directly into a set of disjoint clusters. They try to optimize certain criteria. The criterion function may emphasize the local structure of the data, as by assigning clusters to peaks in the probability density function, or the global structure. Typically, the global criteria involve minimizing some measure of dissimilarity in the samples within each cluster, while maximizing the dissimilarity of different clusters. The advantages of the hierarchical algorithms are the disadvantages of the partitional algorithms and vice versa. An extensive survey of various clustering techniques can be found in (Jain *et al.*, 1999). The focus of this chapter is on the partitional clustering algorithms.

Clustering can also be performed in two different modes: crisp and fuzzy. In crisp clustering, the clusters are disjoint and non-overlapping in nature.

Any pattern may belong to one and only one class in this case. In case of fuzzy clustering, a pattern may belong to all the classes with a certain fuzzy membership grade (Jain *et al.*, 1999).

The most widely used iterative K-means algorithm (MacQueen, 1967) for partitional clustering aims at minimizing the ICS (Intra-Cluster Spread) which for K cluster centers can be defined as

$$ICS(C_1, C_2, ..., C_K) = \sum_{i=1}^{K} \sum_{\mathbf{X}_i \in C_i} \|\mathbf{X}_i - \mathbf{m}_i\|^2 \tag{10}$$

The K-means (or hard c-means) algorithm starts with K cluster-centroids (these centroids are initially selected randomly or derived from some a priori information). Each pattern in the data set is then assigned to the closest cluster-centre. Centroids are updated by using the mean of the associated patterns. The process is repeated until some stopping criterion is met.

In the c-medoids algorithm (Kaufman and Rousseeuw, 1990), on the other hand, each cluster is represented by one of the representative objects in the cluster located near the center. Partitioning around medoids (PAM) (Kaufman and Rousseeuw, 1990) starts from an initial set of medoids, and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering. Although PAM works effectively for small data, it does not scale well for large datasets. Clustering large applications based on randomized search (CLARANS) (Ng and Han, 1994), using randomized sampling, is capable of dealing with the associated scalability issue.

The fuzzy c-means (FCM) (Bezdek, 1981) seems to be the most popular algorithm in the field of fuzzy clustering. In the classical FCM algorithm, a *within cluster sum* function J_m is minimized to evolve the proper cluster centers:

$$J_m = \sum_{j=1}^{n} \sum_{i=1}^{c} (u_{ij})^m \|\mathbf{X}_j - \mathbf{V}_i\|^2 \tag{11}$$

where $\mathbf{V}_i$ is the i-th cluster center, $\mathbf{X}_j$ is the j-th d-dimensional data vector and || . || is an inner product-induced norm in d dimensions. Given *c* classes, we can determine their cluster centers $\mathbf{V}_i$ for i=1 to c by means of the following expression:

$$\mathbf{V}_i = \frac{\sum\limits_{j=1}^{n} (u_{ij})^m \mathbf{X}_j}{\sum\limits_{j=1}^{n} (u_{ij})^m} \tag{12}$$

Here m (m¿1) is any real number that influences the membership grade. Now differentiating the performance criterion with respect to $\mathbf{V}_i$ (treating u_{ij} as constants) and with respect to u_{ij} (treating $\mathbf{V}_i$ as constants) and setting them to zero the following relation can be obtained:

$$u_{ij} = \left[\sum_{k=1}^{c} \left(\frac{\|\mathbf{X}_j - \mathbf{V}_i\|^2}{\|\mathbf{X} - \mathbf{V}_i\|^2} \right)^{1/(m-1)} \right]^{-1} \tag{13}$$

Several modifications of the classical FCM algorithm can be found in (Hall *et al.*, 1999,Gath and Geva, 1989,Bensaid *et al.*, 1996,Clark *et al.*, 1994,Ahmed *et al.*, 2002,Wang *et al.*, 2004).

3.3 Relevance of SI Algorithms in Clustering

From the discussion of the previous section, we see that the SI algorithms are mainly stochastic search and optimization techniques, guided by the principles of collective behaviour and self organization of insect swarms. They are efficient, adaptive and robust search methods producing near optimal solutions and have a large amount of implicit parallelism. On the other hand, data clustering may be well formulated as a difficult global optimization problem; thereby making the application of SI tools more obvious and appropriate.

4 Clustering with the SI Algorithms

In this section we first review the present state of the art clustering algorithms based on SI tools, especially the ACO and PSO. We then outline a new algorithm which employs the PSO model to automatically determine the number of clusters in a previously unhandled dataset. Computer simulations undertaken for this study have also been included to demonstrate the elegance of the new dynamic clustering technique.

4.1 The Ant Colony Based Clustering Algorithms

Ant colonies provide a means to formulate some powerful nature-inspired heuristics for solving the clustering problems. Among other social movements, researchers have simulated the way, ants work collaboratively in the task of grouping dead bodies so, as to keep the nest clean (Bonabeau *et al.*, 1999). It can be observed that, with time the ants tend to cluster all dead bodies in a specific region of the environment, thus forming piles of corpses.

Larval sorting and corpse cleaning by ant was first modeled by Deneubourg *et al.* for accomplishing certain tasks in robotics (Deneubourg *et al.*, 1991). This inspired the Ant-based clustering algorithm (Handl *et al.*, 2003). Lumer and Faieta modified the algorithm using a dissimilarity-based evaluation of the local density, in order to make it suitable for data clustering (Lumer and Faieta, 1994). This introduced standard Ant Clustering Algorithm (ACA). It has subsequently been used for numerical data analysis (Lumer and Faieta, 1994), data-mining (Lumer and Faieta, 1995), graph-partitioning (Kuntz and Snyers, 1994, Kuntz and Snyers, 1999, Kuntz *et al.*, 1998) and text-mining (Handl and Meyer, 2002, Hoe *et al.*, 2002, Ramos and Merelo, 2002). Many authors (Handl and Meyer, 2002, Ramos *et al.*, 2002) proposed a number of modifications to improve the convergence rate and to get optimal number of clusters. Monmarche *et al.* hybridized the Ant-based clustering algorithm with K-means algorithm (Monmarche *et al.*, 1999) and compared it to traditional K-means on various data sets, using the classification error for evaluation purposes. However, the results obtained with this method are not applicable to ordinary ant-based clustering since it differs significantly from the latter.

Like a standard ACO, ant-based clustering is a distributed process that employs positive feedback. Ants are modeled by simple agents that randomly move in their environment. The environment is considered to be a low dimensional space, more generally a two-dimensional plane with square grid. Initially, each data object that represents a multi-dimensional pattern is randomly distributed over the 2-D space. Data items that are scattered within this environment can be picked up, transported and dropped by the agents in a probabilistic way. The picking and dropping operation are influenced by the similarity and density of the data items within the ant's local neighborhood. Generally, the size of the neighborhood is 3×3. Probability of picking up data items is more when the object are either isolated or surrounded by dissimilar items. They trend to drop them in the vicinity of similar ones. In this way, a clustering of the elements on the grid is obtained.

The ants search for the feature space either through random walk or with jumping using a short term memory. Each ant picks up or drops objects according to the following local probability density measure:

$$f(\mathbf{X}_i) = \max\{0, \frac{1}{s^2} \sum\nolimits_{\mathbf{X}_j \in N_{s\times s}(r)} [1 - \frac{d(\mathbf{X}_i, \mathbf{X}_j)}{\alpha(1 + \frac{\nu - 1}{\nu_{\max}})} \tag{14}$$

In the above expression, $N_{s\times s}(r)$ denotes the local area of perception surrounding the site of radius r, which the ant occupies in the two-dimensional grid. The threshold αg cales the dissimilarity within each pair of objects, and the moving speed v controls the step-size of the ant searching in the space within one time unit. If an ant is not carrying an object and finds an object $\mathbf{X}_i$ in its neighborhood, it picks up this object with a probability that is inversely proportional to the number of similar objects in the neighborhood. It may be expressed as:

$$P_{pick-up}(\mathbf{X}_i) = [\frac{k_p}{k_p + f(\mathbf{X}_i)}]^2 \tag{15}$$

If however, the ant is carrying an object x and perceives a neighbor's cell in which there are other objects, then the ant drops off the object it is carrying with a probability that is directly proportional to the object's similarity with the perceived ones. This is given by:

$$P_{drop}(\mathbf{X}_i) = \begin{matrix} 2.f(\mathbf{X}_i) & iff(\mathbf{X}_i) < k_d \\ 1 & iff(\mathbf{X}_i) \geq k_d \end{matrix}$$

The parameters k_p and k_d are the picking and dropping constants (Gath and Geva, 1989) respectively. Function $f(\mathbf{X}_i)$ provides an estimate of the density and similarity of elements in the neighborhood of object $\mathbf{X}_i$. The standard ACA pseudo-code is summarized in Algorithm 3.

Algorithm 3: Procedure ACA

```
1: Place every item $\mathbf{X}_i$ on a random cell of the grid;
2: Place every ant k on a random cell of the grid unoccupied by ants;
3: iteration_count ← 1;
4: while iteration_count < maximum_iteration do
5:    for $i = 1$ to $no\_of\_ants$ do
6:       if unladen ant and cell occupied by item $\mathbf{X}_i$ then
7:          compute $f(\mathbf{X}_i)$ and $P_{pick-up}(\mathbf{X}_i)$;
8:       else
9:          if ant carrying item xi and cell empty then
10:            compute $f(\mathbf{X}_i)$ and $P_{drop}(\mathbf{X}_i)$;
11:            drop item $\mathbf{X}_i$ with probability $P_{drop}(\mathbf{X}_i)$;
12:         end if
13:      end if
14:      move to a randomly selected, neighboring and unoccupied cell ;
15:   end for
16:   t ← t + 1
17: end while
18: print location of items;
```

Kanade and Hall (Kanade and Hall, 2003) presented a hybridization of the ant systems with the classical FCM algorithm to determine the number of clusters in a given dataset automatically. In their fuzzy ant algorithm, at first the ant based clustering is used to create raw clusters and then these clusters are refined using the FCM algorithm. Initially the ants move the individual data objects to form heaps. The centroids of these heaps are taken as the initial cluster centers and the FCM algorithm is used to refine these clusters. In the second stage the objects obtained from the FCM algorithm are hardened according to the maximum membership criteria to form new heaps. These new heaps are then sometimes moved and merged by the ants. The final clusters formed are refined by using the FCM algorithm.

A number of modifications have been introduced to the basic ant based clustering scheme that improve the quality of the clustering, the speed of convergence and, in particular, the spatial separation between clusters on the grid, which is essential for the scheme of cluster retrieval. A detailed description of the variants and results on the qualitative performance gains afforded by these extensions are provided in (Tsang and Kwong, 2006).

4.2 The PSO Based Clustering Algorithms

Research efforts have made it possible to view data clustering as an optimization problem. This view offers us a chance to apply PSO algorithm for evolving a set of candidate cluster centroids and thus determining a near optimal partitioning of the dataset at hand. An important advantage of the PSO is its ability to cope with local optima by maintaining, recombining and comparing several candidate solutions simultaneously. In contrast, local search heuristics, such as the simulated annealing algorithm (Selim and Alsultan, 1991) only refine a single candidate solution and are notoriously weak in coping with local optima. Deterministic local search, which is used in algorithms like the K-means, always converges to the nearest local optimum from the starting position of the search.

PSO-based clustering algorithm was first introduced by Omran *et al.* in (Omran *et al.*, 2002). The results of Omran *et al.* (Omran *et al.*, 2002,Omran *et al.*, 2005a) showed that PSO based method outperformed K-means, FCM and a few other state-of-the-art clustering algorithms. In their method, Omran *et al.* used a quantization error based fitness measure for judging the performance of a clustering algorithm. The quantization error is defined as:

$$J_e = \frac{\sum_{i=1}^{K} \sum_{\forall \mathbf{X}_j \in C_i} d(\mathbf{X}_j, \mathbf{V}_i)/n_i}{K} \tag{16}$$

where C_i is the i-th cluster center and n_i is the number of data points belonging to the i-th cluster. Each particle in the PSO algorithm represents a possible set of K cluster centroids as:

$$\vec{Z}_i(t) = \begin{array}{|c|c|c|c|}\hline \vec{V}_{i,1} & \vec{V}_{i,2} & .... & \vec{V}_{i,K} \\ \hline \end{array}$$

where $\mathbf{V}_{i,p}$ refers to the p-th cluster centroid vector of the i-th particle. The quality of each particle is measured by the following fitness function:

$$f(\mathbf{Z}_i, M_i) = w_1 \bar{d}_{\max}(M_i, \mathbf{X}_i) + w_2(R_{\max} - d_{\min}(\mathbf{Z}_i)) + w_3 J_e \tag{17}$$

In the above expression, R$_{max}$ is the maximum feature value in the dataset and $\mathbf{M}_i$ is the matrix representing the assignment of the patterns to the clusters of the i-th particle. Each element m$_{i,k,p}$indicates whether the pattern $\mathbf{X}_p$belongs to cluster C$_k$ of i-th particle. The user-defined constants w_1, w_2, and w_3 are used to weigh the contributions from different sub-objectives. In addition,

$$\bar{d}_{\max} = \max_{k \in 1,2,\ldots.,K} \{ \sum_{\forall \mathbf{X}_p \in C_{i,K}} d(\mathbf{X}_p, \mathbf{V}_{i,k})/n_{i,k} \} \tag{18}$$

and,

$$d_{\min}(\mathbf{Z}_i) = \min_{\forall p,q,p \neq q} \{ d(\mathbf{V}_{i,p}, \mathbf{V}_{i,q}) \} \tag{19}$$

is the minimum Euclidean distance between any pair of clusters. In the above, $n_{i,k}$is the number of patterns that belong to cluster $\mathbf{C}i,k$ of particle i. he fitness function is a multi-objective optimization problem, which minimizes the intra-cluster distance, maximizes inter-cluster separation, and reduces the quantization error. The PSO clustering algorithm is summarized in Algorithm 4.

Algorithm 4: The PSO Clustering Algorithm

```
 1: Initialize each particle with K random cluster centers.
 2: for iteration_count = 1 to maximum_iterations do
 3:    for all particle i do
 4:       for all pattern $\mathbf{X}_p$ in the dataset do
 5:          calculate Euclidean distance of $\mathbf{X}_p$ with all cluster centroids
 6:          assign $\mathbf{X}_p$ to the cluster that have nearest centroid to $\mathbf{X}_p$
 7:       end for
 8:       calculate the fitness function $f(\mathbf{Z}_i, M_i)$
 9:    end for
10:    find the personal best and global best position of each particle.
11:    Update the cluster centroids according to velocity updating and coordinate
       updating formula of PSO.
12: end for
```

Van der Merwe and Engelbrecht hybridized this approach with the k-means algorithm for clustering general dataets (van der Merwe and Engelbrecht, 2003). A single particle of the swarm is initialized with the result of the k-means algorithm. The rest of the swarm is initialized randomly. In 2003, Xiao *et al* used a new approach based on the synergism of the PSO and the Self Organizing Maps (SOM) (Xiao *et al.*, 2003) for clustering gene expression data. They got promising results by applying the hybrid SOM-PSO algorithm over the gene expression data of Yeast and Rat Hepatocytes. Paterlini and Krink (Paterlini and Krink, 2006) have compared the performance of K-means, GA (Holland, 1975, Goldberg, 1975), PSO and Differential Evolution (DE) (Storn and Price, 1997) for a representative point evaluation approach to partitional clustering. The results show that PSO and DE outperformed the K-means algorithm.

Cui et al. (Cui and Potok, 2005) proposed a PSO based hybrid algorithm for classifying the text documents. They applied the PSO, K-means and a hybrid PSO clustering algorithm on four different text document datasets. The results illustrate that the hybrid PSO algorithm can generate more compact clustering results over a short span of time than the K-means algorithm.

4.3 An Automatic Clustering Algorithm Based on PSO

Tremendous research effort has gone in the past few years to evolve the clusters in complex datasets through evolutionary computing techniques. However, little work has been taken up to determine the optimal number of clusters at the same time. Most of the existing clustering techniques, based on evolutionary algorithms, accept the number of classes K as an input instead of determining the same on the run. Nevertheless, in many practical situations, the appropriate number of groups in a new dataset may be unknown or impossible to determine even approximately. For example, while clustering a set of documents arising from the query to a search engine, the number of classes K changes for each set of documents that result from an interaction with the search engine. Also if the dataset is described by high-dimensional feature vectors (which is very often the case), it may be practically impossible to visualize the data for tracking its number of clusters.

Finding an optimal number of clusters in a large dataset is usually a challenging task. The problem has been investigated by several researches (Halkidi *et al.*, 2001, Theodoridis and Koutroubas, 1999) but the outcome is still unsatisfactory (Rosenberger and Chehdi, 2000). Lee and Antonsson (Lee and Antonsson, 2000) used an Evolutionary Strategy (ES) (Schwefel, 1995) based method to dynamically cluster a dataset. The proposed ES implemented variable-length individuals to search for both centroids and optimal number of clusters. An approach to classify a dataset dynamically using Evolutionary Programming (EP) (Fogel *et al.*, 1966) can be found in Sarkar (Sarkar *et al.*, 1997) where two fitness functions are optimized simultaneously: one gives the optimal number of clusters, whereas the other leads to a proper identification of each cluster's centroid. Bandopadhyay *et al.* (Bandyopadhyay and Maulik, 2000) devised a variable string-length genetic algorithm (VGA) to tackle the dynamic clustering problem using a single fitness function. Very recently, Omran *et al.* came up with an automatic hard clustering scheme (Omran *et al.*, 2005c). The algorithm starts by partitioning the dataset into a relatively large number of clusters to reduce the effect of the initialization. Using binary PSO (Kennedy and Eberhart, 1997), an optimal number of clusters is selected. Finally, the centroids of the chosen clusters are refined through the K-means algorithm. The authors applied the algorithm for segmentation of natural, synthetic and multi-spectral images.

In this section we discuss a new fuzzy clustering algorithm (Das *et al.*, 2006), which can automatically determine the number of clusters in a given dataset. The algorithm is based on a modified PSO algorithm with improved convergence properties.

The Modification of the Classical PSO

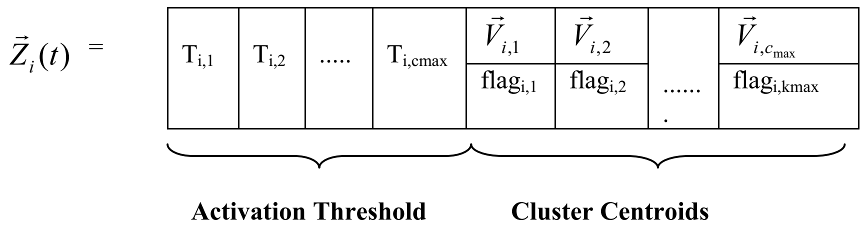
The canonical PSO has been subjected to empirical and theoretical investigations by several researchers (Eberhart and Shi, 2001, Clerc and Kennedy, 2002). In many occasions, the convergence is premature, especially if the swarm uses a small inertia weight ω or constriction coefficient (Clerc and Kennedy, 2002). As the global best found early in the searching process may be a poor local minima, we propose a multi-elitist strategy for searching the global best of the PSO. We call the new variant of PSO the MEPSO. The idea draws inspiration from the works reported in (Deb *et al.*, 2002). We define a growth rate β for each particle. When the fitness value of a particle of t-th iteration is higher than that of a particle of (t-1)-th iteration, the β will be increased. After the local best of all particles are decided in each generation, we move the local best, which has higher fitness value than the global best into the candidate area. Then the global best will be replaced by the local best with the highest growth rate β. Therefore, the fitness value of the new global best is always higher than the old global best. The pseudo code about MEPSO is described in Algorithm 5.

Algorithm 5: The MEPSO Algorithm

```
1: for $t = 1$ to $t_{max}$ do
2:    if $t < t_{max}$ then
3:       for $j = 1$ to $N$ do {swarm size is N}
4:          if the fitness value of $particle_j$ in $t$-th time-step > that of $particle_j$ in
            $(t-1)$-th time-step then
5:             $\beta_j = \beta_j + 1$;
6:          end if
7:          Update Local $best_j$.
8:          if the fitness of Local $best_j$ > that of Global best now then
9:             Choose Local $best_j$ put into candidate area.
10:         end if
11:      end for
12:      Calculate $\beta$ of every candidate, and record the candidate of $\beta_{max}$ .
13:      Update the Global best to become the candidate of $\beta_{max}$.
14:   else
15:      Update the Global best to become the particle of highest fitness value.
16:   end if
17: end for
```

Particle Representation

In the proposed method, for n data points, each p-dimensional, and for a user-specified maximum number of clusters c_{max}, a particle is a vector of real numbers of dimension $c_{max} + c_{max} \times p$. The first c_{max}entries are positive floating-point numbers in (0, 1), each of which controls whether the corresponding cluster is to be activated (i.e. to be really used for classifying the data) or not. The remaining entries are reserved for c_{max} cluster centers, each p-dimensional. A single particle can be shown as:



Every probable cluster center $m_{i,j}$ has p features and a binary $flag_{i,j}$ associated with it. The cluster center is active (i.e., selected for classification) if $flag_{i,j} = 1$ and inactive if $flag_{i,j} = 0$. Each flag is set or reset according to the value of the activation threshold $T_{i,j}$. Note that these flags are latent information associated with the cluster centers and do not take part in the PSO-type mutation of the particle. The rule for selecting the clusters specified by one particle is:

$$IfT_{i,j} > 0.5Thenflag_i, j = 1Elseflag_{i,j} = 0 \qquad (20)$$

Note that the flags in an offspring are to be changed only through the T_{ij}'s (according to the above rule). When a particle jumps to a new position, according to (8), the T values are first obtained which then are used to select (via equation (6)) the m values. If due to mutation some threshold T in a particle exceeds 1 or becomes negative, it is fixed to 1 or zero, respectively. However, if it is found that no flag could be set to one in a particle (all activation thresholds are smaller than 0.5), we randomly select 2 thresholds and re-initialize them to a random value between 0.5 and 1.0. Thus the minimum number of possible clusters is 2.

Fitness Function

The quality of a partition can be judged by an appropriate cluster validity index. Cluster validity indices correspond to the statistical-mathematical functions used to evaluate the results of a clustering algorithm on a quantitative basis. Generally, a cluster validity index serves two purposes. First, it can be used to determine the number of clusters, and secondly, it finds out the corresponding best partition. One traditional approach for determining the optimum number of classes is to run the algorithm repeatedly with different number of classes as input and then to select the partitioning of the data resulting in the best validity measure (Halkidi and Vazirgiannis, 2001). Ideally, a validity index should take care of the following aspects of the partitioning:

1. ***Cohesion***: Patterns in one cluster should be as similar to each other as possible. The fitness variance of the patterns in a cluster is an indication of the cluster's cohesion or compactness.
2. ***Separation*:** Clusters should be well separated. The distance among the cluster centers (may be their Euclidean distance) gives an indication of cluster separation.

In the present work we have based our fitness function on the Xie-Benni index. This index, due to (Xie and Beni, 1991), is given by:

$$XB_m = \frac{\sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^2 \left\| \mathbf{X}_j - \mathbf{V}_i \right\|^2}{n \times \min_{i \neq j} \left\| \mathbf{V}_i - \mathbf{V}_j \right\|^2} \tag{21}$$

Using XB_m the optimal number of clusters can be obtained by minimizing the index value. The fitness function may thus be written as:

$$f = \frac{1}{XB_i(c) + eps} \tag{22}$$

where XB$_i$ is the Xie-Benni index of the i-th particle and eps is a very small constant (we used 0.0002). So maximization of this function means minimization of the XB index.

We have employed another famous validity index known as the partition entropy in order to judge the accuracy of the final clustering results obtained by MEPSO and its competitor algorithms in case of the image pixel classification. The partition entropy (Bezdek, 1981) function is given by,

$$V_{pe} = \frac{- \sum_{j=1}^{n} \sum_{i=1}^{c} [u_{ij} \log u_{ij}]}{n} \tag{23}$$

The idea of the validity function is that the partition with less fuzziness means better performance. Consequently, the best clustering is achieved when the value V$_{pe}$ is minimal.

4.4 Avoiding Erroneous particles with Empty Clusters or Unreasonable Fitness Evaluation

There is a possibility that in our scheme, during computation of the XB index, a division by zero may be encountered. This may occur when one of the selected cluster centers is outside the boundary of distributions of the data set. To avoid this problem we first check to see if any cluster has fewer than 2 data points in it. If so, the cluster center positions of this special chromosome are re-initialized by an average computation. We put n/c data points for every individual cluster center, such that a data point goes with a center that is nearest to it.

4.5 Combining All Together

The clustering method described here, is a two-pass process at each iteration or time step. The first pass amounts to calculating the active clusters as well as the membership functions for each particle in the spectral domain. In the second pass, the membership information of each pixel is mapped to the spatial domain, and the spatial function is computed from that. The MEPSO iteration proceeds with the new membership that is incorporated with the spatial function. The algorithm is stopped when the maximum number of time-steps t_{max} is exceeded. After the convergence, de-fuzzification is applied to assign each data item to a specific cluster for which the membership is maximal.

4.6 A Few Simulation Results

The MEPSO-clustering algorithm has been tested over a number of synthetic and real world datasets as well as on some image pixel classification problems. The performance of the method has been compared with the classical FCM algorithm and a recently developed fuzzy clustering algorithm based on GA. The later algorithm is referred in literature as Fuzzy clustering with Variable length Genetic Algorithm (FVGA) the details of which can be found in (Pakhira *et al.*, 2005). In the present chapter, we first provide the simulation results obtained over four well-chosen synthetic datasets (Bandyopadhyay and Maulik, 2000) and two real world datasets. The real world datasets used are the glass and the Wisconsin breast cancer data set, both of which have been taken from the UCI public data repository (Blake *et al.*, 1998). The glass data were sampled from six different type of glass: building windows float processed (70 objects), building windows non float processed (76 objects), vehicle windows float processed (17 objects), containers (13 objects), tableware (9 objects), headlamps (29 objects) with nine features each. The Wisconsin breast cancer database contains 9 relevant features: clump thickness, cell size uniformity, cell shape uniformity, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli and mitoses. The dataset has two classes. The objective is to classify each data vector into benign (239 objects) or malignant tumors (444 objects).

Performance of the MEPSO based algorithm on four synthetic datasets has been shown in Figures 5 through 8. In Table 1, we provide the mean value and standard deviation of the Xie Beni index evaluated over final clustering results, the number of classes evaluated and the number of misclassified items with respect to the nominal partitions of the benchmark data, as known to us. For each data set, each run continues until the number of function evaluations (FEs) reaches 50,000. Twenty independent runs (with different seeds for the random number generator) have been taken for each algorithm. The results have been stated in terms of the mean *best-of-run* values and standard deviations over these 20 runs in each case. Only for the FCM, correct number of classes has been provided as input. Both FVGA and MEPSO determine the number of classes automatically on the run.

From Tables 1 and 2, one may see that our approach outperforms the state-of-the-art FVGA and the classical FCM over a variety of datasets in a statistically significant manner. Not only does the method find the optimal number of clusters, it also manages to find better clustering of the data points in terms of the two major cluster validity indices used in the literature.

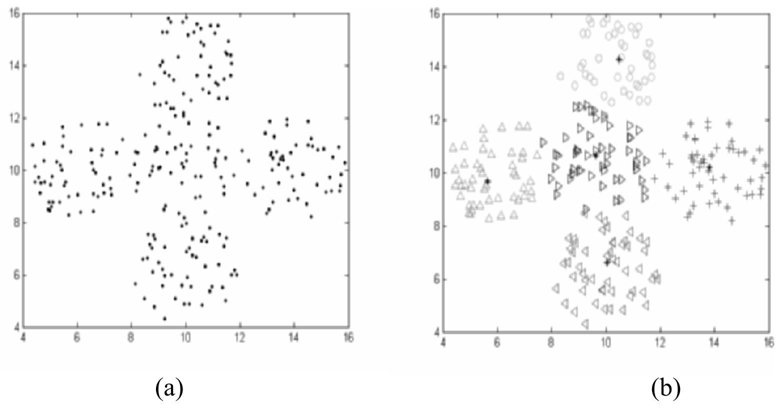


Fig. 5. (a) The unlabeled synthetic dataset 1 (b) Automatic Clustering with the MEPSO

4.7 Image Segmentation through Clustering

Image segmentation may be defined as the process of dividing an image into disjoint homogeneous regions. These homogeneous regions usually contain similar objects of interest or part of them. The extent of homogeneity of the segmented regions can be measured using some image property (e.g. pixel intensity (Jain *et al.*, 1999)). Segmentation forms a fundamental step towards several complex computer-vision and image analysis applications including digital mammography, remote sensing and land cover study. Image segmentation can be treated as a clustering problem where the features describing each pixel correspond to a pattern, and each image region (i.e., segment)

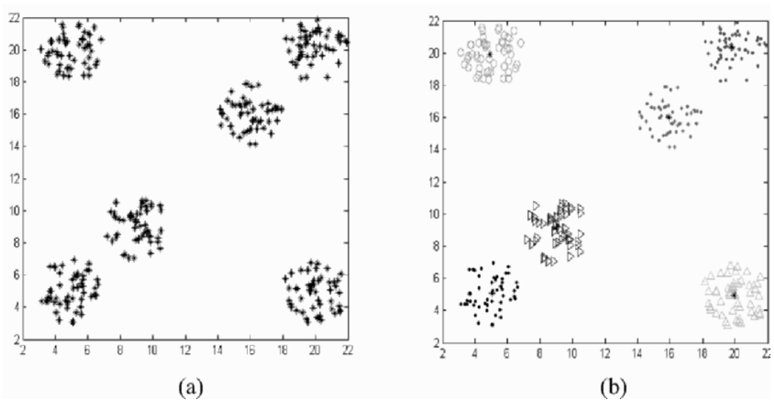


Fig. 6. (a) The unlabeled synthetic dataset 1 (b) Automatic Clustering with the MEPSO

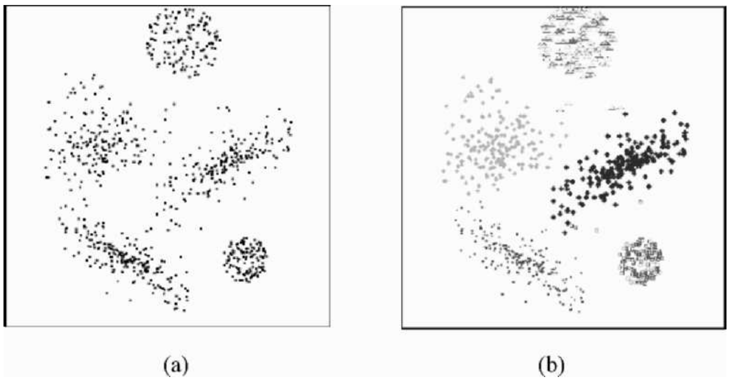


Fig. 7. (a) The unlabeled synthetic dataset 1 (b) Automatic Clustering with the MEPSO

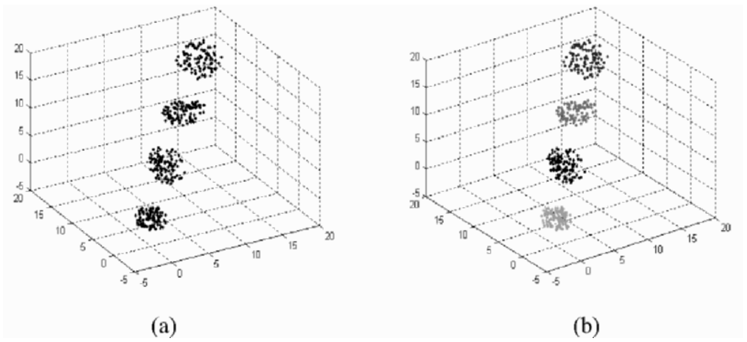


Fig. 8. (a) The unlabeled synthetic dataset 1 (b) Automatic Clustering with the MEPSO

Table 1. Final solution (mean and standard deviation over 20 independent runs) after each algorithm was terminated after running for 50,000 function evaluations (FE) with DB Measure based fitness function.

Problem	**Algorithm**	**Average no. of clusters found**	**Final DB measure**	**Mean No. of misclassified Items**
Synthetic Data 1	MEPSO	**5.05±0.0931**	**3.0432±0.021**	**5.25±0.096**
	FVGA	8.15±0.0024	4.3432±0.232	15.75±0.154
	FCM	NA	5.3424±0.343	19.50±1.342
Synthetic Data 2	MEPSO	**6.45±0.0563**	**1.4082±0.006**	**4.50±0.023**
	FVGA	6.95±0.021	1.5754±0.073	10.25±0.373
	FCM	NA	1.6328±0.002	26.50±0.433
Synthetic Data 3	MEPSO	**5.25±0.0241**	**0.9224±0.334**	**9.15±0.034**
	FVGA	5.75±0.0562	1.2821±0.009	15.50±0.048
	FCM	NA	2.9482±0.028	17.25±0.275
Synthetic Data 4	MEPSO	**4.00±0.00**	**1.0092±0.083**	**1.50±0.035**
	FVGA	4.75±0.0193	1.5152±0.073	4.55±0.05
	FCM	NA	1.8371±0.034	8.95±0.15
Glass	MEPSO	**6.05±0.0248**	**1.0802±0.083**	**8.35±0.662**
	FVGA	5.95±0.0193	1.5152±0.073	14.35±0.26
	FCM	NA	1.8371±0.034	18.65±0.85
Breast Cancer	MEPSO	2.05±0.0563	0.5003±0.006	**25.00±0.09**
	FVGA	2.50±0.0621	0.5754±0.073	26.50±0.80
	FCM	NA	0.6328±0.002	30.23±0.46

corresponds to a cluster (Jain *et al.*, 1999). Therefore, many clustering algorithms have widely been used to solve the segmentation problem (e.g., K-means (Tou and Gonzalez, 1974), Fuzzy C-means (Trivedi and Bezdek, 1986), ISODATA (Ball and Hall, 1967), Snob (Wallace and Boulton, 1968) and recently the PSO and DE based clustering techniques (Omran *et al.*, 2005a, Omran *et al.*, 2005b)).

Here we illustrate the automatic soft segmentation of a number of grey scale images by using our MEPSO based clustering algorithm. An important characteristic of an image is the high degree of correlation among the neighboring pixels. In other words, these neighboring pixels possess similar feature values, and the probability that they belong to the same cluster is great. This spatial relationship (Ahmed *et al.*, 2002) is important in clustering, but it is not utilized in a standard FCM algorithm. To exploit the spatial information, a spatial function is defined as:

$$h_{ij} = \sum_{k \in \delta(\mathbf{X}_j)} u_{ik} \tag{24}$$

where $\delta(\mathbf{X}_j)$represents a square window centered on pixel (i.e. data point) $\mathbf{X}_j$ in the spatial domain. A 5×5 window was used throughout this work. Just like the membership function, the spatial function h_{ij} represents the probability that pixel $\mathbf{X}_j$belongs to i-th cluster. The spatial function of a pixel for a cluster is large if the majority of its neighborhood belongs to the same clusters. We incorporate the spatial function into membership function as follows:

$$u'_{ij} = \frac{u^r_{ij} h^t_{ij}}{\sum\limits_{k=1}^{c} u^r_{kj} h^t_{kj}} \tag{25}$$

Here in all the cases we have used r = 1, t = 1after considerable trial and errors.

Although we tested our algorithm over a large number of images with varying range of complexity, here we show the experimental results for three images only, due to economy of space. Figures 4.7 to 4.7 show the three original images and their segmented counterparts obtained using the FVGA algorithm and the MEPSO based method. In these figures the segmented portions of an image have been marked with the grey level intensity of the respective cluster centers. In Table 2, we report the mean value the DB measure and partition entropy calculated over the 'best-of-run' solutions in each case. One may note that the MEPSO meets or beats the competitor algorithm in all the cases. Table 3 reports the mean time taken by each algorithm to terminate on the image data. Finally, Table 4 contains the mean and standard deviations of the number of classes obtained by the two automatic clustering algorithms.

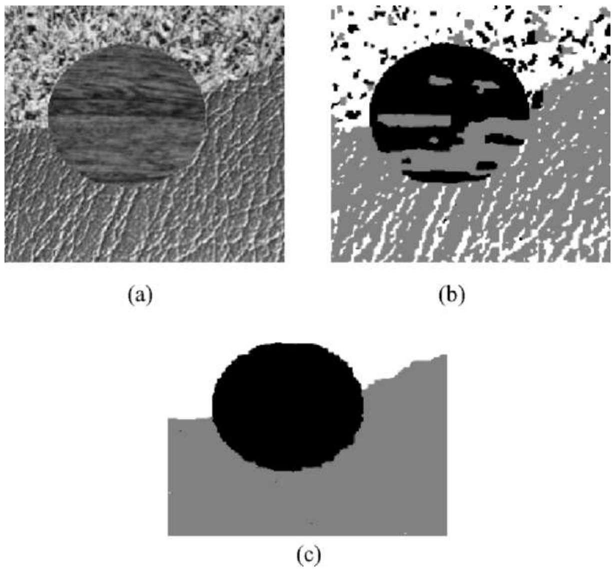


Fig. 9. (a) The original Texture image. (b) Segmentation by FVGA (c= 3) (c) Segmentation by MEPSO based method (c = 3)

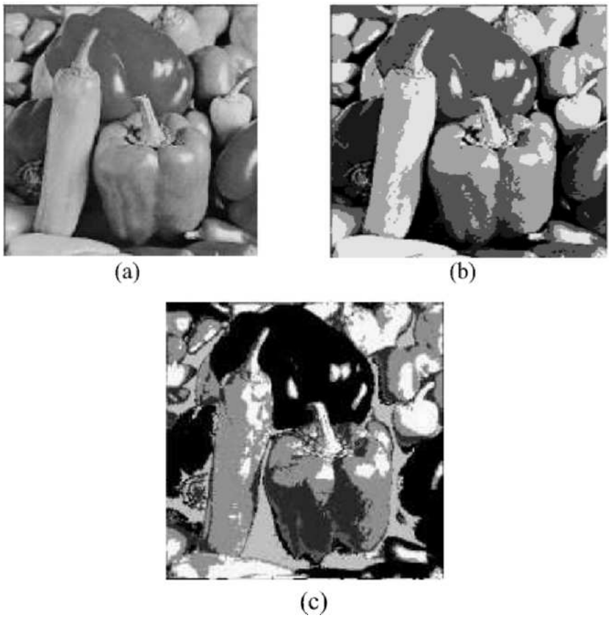


Fig. 10. (a) The original Pepper image. (b) Segmentation by FVGA (c= 7) (c) Segmentation by MEPSO based method (c = 7)

Table 2. Automatic clustering result for three real life grayscale images (over 20 runs; each run continued up to 50,000 FE)

Image	Validity Index	Mean and Std Dev of the validity indices over the final clustering results of 20 independent runs		
		AFDE	FVGA	FCM
Texture	Xie-Beni	**0.7283 (0.0001)**	0.7902 (0.0948)	0.7937 (0.0013)
	Partition Entropy	**2.6631 (0.7018)**	2.1193 (0.8826)	2.1085 (0.0043)
MRI Image of Brain	Xie-Beni	**0.2261 (0.0017)**	0.2919 (0.0583)	0.3002 (0.0452)
	Partition Entropy	**0.1837 (0.0017)**	0.1922 (0.0096)	0.1939 (0.0921)
Pepper Image	Xie-Beni	**0.05612 (0.0092)**	0.09673 (0.0043)	0.09819 (0.0001)
	Partition Entropy	**0.8872 (0.0137)**	1.1391 (0.0292)	1.1398 (0.0884)

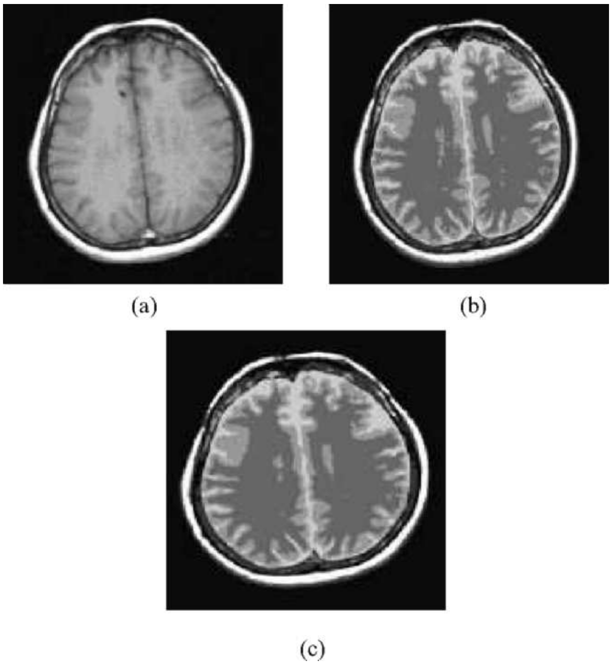


Fig. 11. (a) The original MRI image. (b) Segmentation by FVGA (c= 5) (c) Segmentation by MEPSO (c = 5)

Table 3. Comparison among the mean execution time taken by the different algorithms

Image	Optimal No. of Clusters	Mean and Std Dev of the number of classes estimated by the competitor algorithms	
		FVGA	MEPSO
Texture	3	3.75±0.211	3.05±0.132
MRI	5	5.05±0.428	5.25±0.212
Pepper	7	8.15±0.772	6.95±0.982

Table 4. Automatic clustering results for the three real-life grayscale images (over 20 runs; each runs continued for 50,000 FE)

Image	Mean and Std Dev of the execution time (in seconds) taken by the competitor algorithms	
	FVGA	MEPSO
Texture	32.05±0.076	47.25±0.162
MRI	24.15±0.016	34.65±0.029
Pepper	49.20±0.201	67.85±0.817

5 Conclusion and Future Directions

In this Chapter, we introduced some of the preliminary concepts of Swarm Intelligence (SI) with an emphasis on particle swarm optimization and ant colony optimization algorithms. We then described the basic data clustering terminologies and also illustrated some of the past and ongoing works, which apply different SI tools to pattern clustering problems. We proposed a novel fuzzy clustering algorithm, which is based on a deviant variety of the PSO. The proposed algorithm can automatically compute the optimal number of clusters in any dataset and thus requires minimal user intervention. Comparison with a state of the art GA based clustering strategy, reveals the superiority of the MEPSO-clustering algorithm both in terms of accuracy and speed.

Despite being an age old problem, clustering remains an active field of interdisciplinary research till date. No single algorithm is known, which can group all real world datasets efficiently and without error. To judge the quality of a clustering, we need some specially designed statistical-mathematical function called the clustering validity index. But a literature survey reveals that, most of these validity indices are designed empirically and there is no universally good index that can work equally well over any dataset. Since, majority of the PSO or ACO based clustering schemes rely on a validity index to judge the fitness of several possible partitioning of the data, research effort should be spent for defining a reasonably good index function and validating the same mathematically.

Feature extraction is an important preprocessing step for data clustering. Often we have a great number of features (especially for a high dimensional dataset like a collection of text documents) which are not all relevant for a given operation. Hence, future research may focus on integrating the automatic feature-subset selection scheme with the SI based clustering algorithm. The two-step process is expected to automatically project the data to a low dimensional feature subspace, determine the number of clusters and find out the appropriate cluster centers with the most relevant features at a faster pace.

Gene expression refers to a process through which the coded information of a gene is converted into structures operating in the cell. It provides the physical evidence that a gene has been "turned on" or activated for protein synthesis (Lewin, 1995). Proper selection, analysis and interpretation of the gene expression data can lead us to the answers of many important problems in experimental biology. Promising results have been reported in (Xiao *et al.*, 2003) regarding the application of PSO for clustering the expression levels of gene subsets. The research effort to integrate SI tools in the mechanism of gene expression clustering may in near future open up a new horizon in the field of bioinformatic data mining.

Hierarchical clustering plays an important role in fields like information retrieval and web mining. The self-assembly behavior of the real ants may be exploited to build up new hierarchical tree-structured partitioning of a data set according to the similarities between those data items. A description of the little but promising work already been undertaken in this direction can be found in (Azzag *et al.*, 2006). But a more extensive and systematic research effort is necessary to make the ant based hierarchical models superior to existing algorithms like Birch (Zhang *et al.*, 1997).

References

A. Abraham, C. Grosan and V. Ramos (2006) (Eds.), Swarm Intelligence and Data Mining, Studies in Computational Intelligence, Springer Verlag, Germany, pages 270, ISBN: 3-540-34955-3.

Ahmed MN, Yaman SM, Mohamed N, (2002), Farag AA and Moriarty TA, Modified fuzzy c-means algorithm for bias field estimation and segmentation of MRI data. IEEE Trans Med Imaging, 21, pp. 193–199.

Azzag H, Guinot C and Venturini G, Data and text mining with hierarchical clustering ants, in *Swarm Intelligence in Data Mining*, Abraham A, (2006), Grosan C and Ramos V (Eds), Springer, pp. 153-186.

Ball G and Hall D, (1967), A Clustering Technique for Summarizing Multivariate Data, Behavioral Science 12, pp. 153-155.

Bandyopadhyay S and Maulik U, (2000), Genetic clustering for automatic evolution of clusters and application to image classification, Pattern Recognition, 35, pp. 1197-1208.

Beni G and Wang U, (1989), Swarm intelligence in cellular robotic systems. In *NATO Advanced Workshop on Robots and Biological Systems*, Il Ciocco, Tuscany, Italy.

Bensaid AM, Hall LO, Bezdek JC.and Clarke LP, (1996), Partially supervised clustering for image segmentation. Pattern Recognition, vol. 29, pp. 859-871.

Bezdek JC, (1981), Pattern recognition with fuzzy objective function algorithms. New York: Plenum.

Blake C, Keough E and Merz CJ, (1998), UCI repository of machine learning database http://www.ics.uci.edu/∼mlearn/MLrepository.html.

Bonabeau E, Dorigo M and Theraulaz G, (1999), *Swarm Intelligence: From Natural to Artificial Systems.* Oxford University Press, New York.

Brucker P, (1978), On the complexity of clustering problems. Beckmenn M and Kunzi HP(Eds.), *Optimization and Operations Research*, Lecture Notes in Economics and Mathematical Systems, Berlin, Springer, vol.157, pp. 45-54.

Clark MC, Hall LO, Goldgof DB, Clarke LP, (1994), Velthuizen RP and Silbiger MS , MRI segmentation using fuzzy clustering techniques. IEEE Eng Med Biol, 13, pp.730–742.

Clerc M and Kennedy J. (2002), The particle swarm - explosion, stability, and convergence in a multidimensional complex space, In IEEE Transactions on Evolutionary Computation, 6(1):58-73.

Couzin ID, Krause J, James R, Ruxton GD, Franks NR, (2002), Collective Memory and Spatial Sorting in Animal Groups, Journal of Theoretical Biology, 218, pp. 1-11.

Cui X and Potok TE, (2005), Document Clustering Analysis Based on Hybrid PSO+Kmeans Algorithm, Journal of Computer Sciences (Special Issue), ISSN 1549-3636, pp. 27-33.

Das S, Konar A and Abraham A, (2006), Spatial Information based Image Segmentation with a Modified Particle Swarm Optimization, in proceedings of Sixth International Conference on Intelligent System Design and Applications (ISDA 06) Jinan, Shangdong, China, IEEE Computer Society Press.

Deb K, Pratap A, Agarwal S, and Meyarivan T (2002), A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. on Evolutionary Computation, Vol.6, No.2.

Deneubourg JL, Goss S, Franks N, Sendova-Franks A, (1991), Detrain C and Chetien L , The dynamics of collective sorting: Robot-like ants and ant-like robots. In Meyer JA and Wilson SW (Eds.) *Proceedings of the First International Conference on Simulation of Adaptive Behaviour: From Animals to Animats 1*, pp. 356–363. MIT Press, Cambridge, MA.

Dorigo M and Gambardella LM, (1997), Ant colony system: A cooperative learning approach to the traveling salesman problem, IEEE Trans. Evolutionary Computing, vol. 1, pp. 53–66.

Dorigo M, Maniezzo V and Colorni A, (1996), The ant system: Optimization by a colony of cooperating agents, IEEE Trans. Systems Man and Cybernetics – Part B, vol. 26.

Duda RO and Hart PE, (1973), *Pattern Classification and Scene Analysis.* John Wiley and Sons, USA.

Eberhart RC and Shi Y, (2001), Particle swarm optimization: Developments, applications and resources, In Proceedings of IEEE International Conference on Evolutionary Computation, vol. 1, pp. 81-86.

Evangelou IE, Hadjimitsis DG, Lazakidou AA, (2001), Clayton C, Data Mining and Knowledge Discovery in Complex Image Data using Artificial Neural Networks, Workshop on Complex Reasoning an Geographical Data, Cyprus.

Everitt BS, (1993), *Cluster Analysis.* Halsted Press, Third Edition.

Falkenauer E, (1998), *Genetic Algorithms and Grouping Problems*, John Wiley and Son, Chichester.

Fogel LJ, Owens AJ and Walsh MJ, (1966), Artificial Intelligence through Simulated Evolution. New York: Wiley.

Forgy EW, (1965), Cluster Analysis of Multivariate Data: Efficiency versus Interpretability of classification, Biometrics, 21.

Frigui H and Krishnapuram R, (1999), A Robust Competitive Clustering Algorithm with Applications in Computer Vision, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (5), pp. 450-465.

Fukunaga K, (1990), Introduction to Statistical Pattern Recognition. Academic Press.

Gath I and Geva A, (1989), Unsupervised optimal fuzzy clustering. IEEE Transactions on PAMI, 11, pp. 773-781.

Goldberg DE, (1975), Genetic *Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA.

Grosan C, Abraham A and Monica C, Swarm Intelligence in Data Mining, in *Swarm Intelligence in Data Mining*, Abraham A, (2006), Grosan C and Ramos V (Eds), Springer, pp. 1-16.

Halkidi M and Vazirgiannis M, (2001), Clustering Validity Assessment: Finding the Optimal Partitioning of a Data Set. Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM 01), San Jose, California, USA, pp. 187-194.

Halkidi M, Batistakis Y and Vazirgiannis M, (2001), On Clustering Validation Techniques. Journal of Intelligent Information Systems (JIIS), 17(2-3), pp. 107-145.

Handl J and Meyer B, (2002), Improved ant-based clustering and sorting in a document retrieval interface. In Proceedings of the Seventh International Conference on Parallel Problem Solving from Nature (PPSN VII), volume 2439 of LNCS, pp. 913–923. Springer-Verlag, Berlin, Germany.

Handl J, Knowles J and Dorigo M, (2003), Ant-based clustering: a comparative study of its relative performance with respect to k-means, average link and 1D-som. Technical Report TR/IRIDIA/2003-24. IRIDIA, Universite Libre de Bruxelles, Belgium.

Hoe K, Lai W, and Tai T, (2002), Homogenous ants for web document similarity modeling and categorization. In *Proceedings of the Third International Workshop on Ant Algorithms (ANTS 2002)*, volume 2463 of *LNCS*, pp. 256–261. Springer-Verlag, Berlin, Germany.

Holland JH, (1975), *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor.

Jain AK, Murty MN and Flynn PJ, (1999), Data clustering: a review, ACM Computing Surveys, vol. 31, no.3, pp. 264—323.

Kanade PM and Hall LO, (2003), Fuzzy Ants as a Clustering Concept. In Proceedings of the 22nd International Conference of the North American Fuzzy Information Processing Society (NAFIPS03), pp. 227-232.

Kaufman, L and Rousseeuw, PJ, (1990), *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley & Sons, New York.

Kennedy J and Eberhart R, (1995), Particle swarm optimization, In Proceedings of IEEE International conference on Neural Networks, pp. 1942-1948.

Kennedy J and Eberhart RC, (1997), A discrete binary version of the particle swarm algorithm, Proceedings of the 1997 Conf. on Systems, Man, and Cybernetics, IEEE Service Center, Piscataway, NJ, pp. 4104-4109.

Kennedy J, Eberhart R and Shi Y, (2001), *Swarm Intelligence*, Morgan Kaufmann Academic Press.

Kohonen T, (1995), *Self-Organizing Maps*, Springer Series in Information Sciences, Vol 30, Springer-Verlag.

Konar A, (2005), Computational Intelligence: Principles, Techniques and Applications, Springer.

Krause J and Ruxton GD, (2002), Living in Groups. Oxford: Oxford University Press.

Kuntz P and Snyers D, (1994), Emergent colonization and graph partitioning. In *Proceedings of the Third International Conference on Simulation of Adaptive Behaviour: From Animals to Animats 3*, pp. 494– 500. MIT Press, Cambridge, MA.

Kuntz P and Snyers D, (1999), New results on an ant-based heuristic for highlighting the organization of large graphs. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pp. 1451–1458. IEEE Press, Piscataway, NJ.

Kuntz P, Snyers D and Layzell P, (1998), A stochastic heuristic for visualising graph clusters in a bi-dimensional space prior to partitioning. *Journal of Heuristics*, 5(3), pp. 327–351.

Lee C-Y and Antonsson EK, (2000), Self-adapting vertices for mask layout synthesis Modeling and Simulation of Microsystems Conference (San Diego, March

27–29) eds. M Laudon and B Romanowicz. pp. 83–86.

Leung Y, Zhang J and Xu Z, (2000), Clustering by Space-Space Filtering, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (12), pp. 1396-1410.

Lewin B, (1995), *Genes VII.* Oxford University Press, New York, NY.

Lillesand T and Keifer R, (1994), *Remote Sensing and Image Interpretation*, John Wiley & Sons, USA.

Lumer E and Faieta B, (1994), Diversity and Adaptation in Populations of Clustering Ants. In Proceedings Third International Conference on Simulation of Adaptive Behavior: from animals to animates 3, Cambridge, Massachusetts MIT press, pp. 499-508.

Lumer E and Faieta B, (1995), Exploratory database analysis via self-organization, Unpublished manuscript.

MacQueen J, (1967), Some methods for classification and analysis of multivariate observations, Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, pp. 281-297.

Major PF, Dill LM, (1978), The three-dimensional structure of airborne bird flocks. Behavioral Ecology and Sociobiology, 4, pp. 111-122.

Mao J and Jain AK, (1995), Artificial neural networks for feature extraction and multivariate data projection. IEEE Trans. Neural Networks.vol. 6, 296–317.

Milonas MM, (1994), Swarms, phase transitions, and collective intelligence, In Langton CG Ed., Artificial Life III, Addison Wesley, Reading, MA.

Mitchell T, (1997), *Machine Learning.* McGraw-Hill, Inc., New York, NY.

Mitra S, Pal SK and Mitra P, (2002), Data mining in soft computing framework: A survey, IEEE Transactions on Neural Networks, Vol. 13, pp. 3-14.

Monmarche N, Slimane M and Venturini G, (1999), Ant Class: discovery of clusters in numeric data by a hybridization of an ant colony with the k means algorithm. Internal Report No. 213, E3i, Laboratoire d'Informatique, Universite de Tours.

Ng R and Han J, (1994), Efficient and effective clustering method for spatial data mining. In: Proc. 1994 International Conf. Very Large Data Bases (VLDB'94). Santiago, Chile, September pp. 144–155.

Omran M, Engelbrecht AP and Salman A, (2005), Particle Swarm Optimization Method for Image Clustering. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(3), pp. 297–322.

Omran M, Engelbrecht AP and Salman A, (2005), Differential Evolution Methods for Unsupervised Image Classification, Proceedings of Seventh Congress on Evolutionary Computation (CEC-2005). IEEE Press.

Omran M, Salman A and Engelbrecht AP, (2002), Image Classification using Particle Swarm Optimization. In *Conference on Simulated Evolution and Learning*, volume 1, pp. 370–374.

Omran M, Salman A and Engelbrecht AP, (2005), Dynamic Clustering using Particle Swarm Optimization with Application in Unsupervised Image Classification. Fifth World Enformatika Conference (ICCI 2005), Prague, Czech Republic.

Pakhira MK, Bandyopadhyay S and Maulik, U, (2005), A Study of Some Fuzzy Cluster Validity Indices, Genetic clustering And Application to Pixel Classification, Fuzzy Sets and Systems 155, pp. 191–214.

Pal NR, Bezdek JC and Tsao ECK, (1993), Generalized clustering networks and Kohonen's self-organizing scheme. IEEE Trans. Neural Networks, vol 4, 549–557.

Partridge BL, (1982), The structure and function of fish schools. Science American, 245, pp. 90-99.

Partridge BL, Pitcher TJ, (1980), The sensory basis of fish schools: relative role of lateral line and vision. Journal of Comparative Physiology, 135, pp. 315-325.

Paterlini S and Krink T, (2006), Differential Evolution and Particle Swarm Optimization in Partitional Clustering. *Computational Statistics and Data Analysis*, vol. 50, pp. 1220– 1247.

Paterlini S and Minerva T, (2003), Evolutionary Approaches for Cluster Analysis. In Bonarini A, Masulli F and Pasi G (eds.) *Soft Computing Applications*. Springer-Verlag, Berlin. 167-178.

Ramos V and Merelo JJ, (2002), Self-organized stigmergic document maps: Environments as a mechanism for context learning. In *Proceedings of the First Spanish Conference on Evolutionary and Bio-Inspired Algorithms (AEB 2002)*, pp. 284–293. Centro Univ. M'erida, M'erida, Spain.

Ramos V, Muge F and Pina P, (2002), Self-Organized Data and Image Retrieval as a Consequence of Inter-Dynamic Synergistic Relationships in Artificial Ant Colonies. Soft Computing Systems: Design, Management and Applications. 87, pp. 500–509.

Rao MR, (1971), Cluster Analysis and Mathematical Programming,. Journal of the American Statistical Association, Vol. 22, pp 622-626.

Rokach, L., Maimon, O. (2005), Clustering Methods, Data Mining and Knowledge Discovery Handbook, Springer, pp. 321-352.

Rosenberger C and Chehdi K, (2000), Unsupervised clustering method with optimal estimation of the number of clusters: Application to image segmentation, in Proc. IEEE International Conference on Pattern Recognition (ICPR), vol. 1, Barcelona, pp. 1656-1659.

Sarkar M, Yegnanarayana B and Khemani D, (1997), A clustering algorithm using an evolutionary programming-based approach, Pattern Recognition Letters, 18, pp. 975–986.

Schwefel H-P, (1995), *Evolution and Optimum Seeking*. New York, NY: Wiley, 1st edition.

Selim SZ and Alsultan K, (1991), A simulated annealing algorithm for the clustering problem. Pattern recognition, 24(7), pp. 1003-1008.

Storn R and Price K, (1997), Differential evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, Journal of Global Optimization, 11(4), pp. 341–359.

Theodoridis S and Koutroubas K, (1999), Pattern recognition, Academic Press.

Tou JT and Gonzalez RC, (1974), Pattern Recognition Principles. London, Addison-Wesley.

Trivedi MM and Bezdek JC, (1986), Low-level segmentation of aerial images with fuzzy clustering, IEEE Trans.on Systems, Man and Cybernetics, Volume 16.

Tsang W and Kwong S, Ant Colony Clustering and Feature Extraction for Anomaly Intrusion Detection, in *Swarm Intelligence in Data Mining*, Abraham A, (2006), Grosan C and Ramos V (Eds), Springer, pp. 101-121.

van der Merwe DW and Engelbrecht AP, (2003), Data clustering using particle swarm optimization. In: Proceedings of the 2003 IEEE Congress on Evolutionary Computation, pp. 215-220, Piscataway, NJ: IEEE Service Center.

Wallace CS and Boulton DM, (1968), An Information Measure for Classification, Computer Journal, Vol. 11, No. 2, 1968, pp. 185-194.

Wang X, Wang Y and Wang L, (2004), Improving fuzzy c-means clustering based on feature-weight learning. Pattern Recognition Letters, vol. 25, pp. 1123–32.

Xiao X, Dow ER, Eberhart RC, Miled ZB and Oppelt RJ, (2003), Gene Clustering Using Self-Organizing Maps and Particle Swarm Optimization, Proc of the 17th International Symposium on Parallel and Distributed Processing (PDPS '03), IEEE Computer Society, Washington DC.

Xie, X and Beni G, (1991), Validity measure for fuzzy clustering. IEEE Trans. Pattern Anal. Machine Learning, Vol. 3, pp. 841–846.

Xu, R., Wunsch, D. (2005), Survey of Clustering Algorithms, IEEE Transactions on Neural Networks, Vol. 16(3): 645-678.

Zahn CT, (1971), Graph-theoretical methods for detecting and describing gestalt clusters, IEEE Transactions on Computers C-20, 68–86.

Zhang T, Ramakrishnan R and Livny M, (1997), BIRCH: A New Data Clustering Algorithm and Its Applications, Data Mining and Knowledge Discovery, vol. 1, no. 2, pp. 141-182.

Hall LO, Özyurt IB and Bezdek JC, (1999), Clustering with a genetically optimized approach, IEEE Trans. Evolutionary Computing 3 (2) pp. 103–112.

A Diffusion Framework for Dimensionality Reduction

Alon Schclar[1]

School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel
shekler@post.tau.ac.il

Summary. Many fields of research deal with high-dimensional data sets. Hyper-spectral images in remote sensing and in hyper-spectral microscopy, transactions in banking monitoring systems are just a few examples for this type of sets. Revealing the geometric structure of these data-sets as a preliminary step facilitates their efficient processing. Often, only a small number of parameters govern the structure of the data-set. This number is the true dimension of the data-set and is the motivation to reduce the dimensionality of the set. Dimensionality reduction algorithms try to discover the true dimension of a data set.

In this chapter, we describe a natural framework based on diffusion processes for the multi-scale analysis of high-dimensional data-sets (Coifman and Lafon, 2006). This scheme enables us to describe the geometric structures of such sets by utilizing the Newtonian paradigm according to which a global description of a system can be derived by the aggregation of local transitions. Specifically, a Markov process is used to describe a random walk on the data set. The spectral properties of the Markov matrix that is associated with this process are used to embed the data-set in a low-dimensional space. This scheme also facilitates the parametrization of a data-set when the high dimensional data-set is not accessible and only a pair-wise similarity matrix is at hand.

1 Introduction

In the following, we describe the reason why diffusion processes are suitable for the analysis of data-sets. We also give a brief introduction to Markov processes. A more comprehensive introductory to random processes can be found in (Sheldon, 1983).

1.1 Why diffusion ?

Let $\{x_k\}_{k=1}^{m}$ be a data-set of points. Suppose we wish to *randomly walk* between these points starting arbitrarily from one of the points. At each time t we choose our next point in the path according to a given probability. We denote the probability to move from point x_i to point x_j by p_{ij} and refer to a single move as a *transition*. We look into the the simple case where p_{ij} is inversely proportional to the Euclidean distance between x_i and x_j and where the data-set can be divided into two well separated clusters. This means that the probability to travel within a cluster is much higher than that of traveling between the clusters. Consequently, after a large number of steps we will most probably end up in the cluster we began our walk.

Contrary to this example, in most cases the structure of the data-set is unknown and we need to reveal it. This can be done by calculating for every point x_i the probability we will end up there after traveling between the points for a *long time.* (formally, we look into the asymptotic behavior of the random walk). Points that are clustered together will have similar probabilities.

1.2 Markov processes

The random walk we described above is a special case of a *Markov process.* The general Markov framework describes a random walk between a set of states $\{s_k\}_{k=1}^{m}$. The transition probabilities are given by a $m \times m$ matrix $P = (p_{ij})$ where the i-th row contains the transition probabilities from state s_i to all the other states. In our model we choose the sum of each row to be equal to one. Furthermore, we use a *memoryless* process i.e. the probability to move at time $t+1$ from the current state s_i to the next state s_j depends only on s_i and the transition probability p_{ij}. If we denote the probability to be in state s_i at time t by $p_t(s_i)$, then the *memoryless* property can be described as:

$$p_{t+1}(s_j \,| s_i) = p_t(s_i) \cdot p_{ij}$$

A Markov process can also be modeled by a weighted graph $G = (V, E)$ in which the nodes correspond to the states, the edges to transitions and the edge weights to the transition probabilities. We will use this model to describe the diffusion maps algorithm.

2 The diffusion maps algorithm

As mentioned above, the diffusion maps algorithm is used to analyze a given a set of data points

$$\Gamma = \{x_i\}_{i=1}^{m}, \; x_i \in \mathbb{R}^n \tag{1}$$

It includes the following steps:

1. Construction of an undirected graph G on Γ with a weight function w_ε that corresponds to the *local* point-wise similarity between the points in Γ[1].

[1] This step is skipped in case the set Γ is not accessible and we are only given w_ε.

2. Construction of a random walk on the graph G via a Markov transition matrix P that is derived from w_ε.
3. Eigen-decomposition of P.

The graph weights are chosen to be proportional to the similarity between the points i.e. the similar a pair of points is, the higher is the edge weight connecting the pair. This similarity depends on the application that requires the analysis of Γ and on the nature of the data-set at hand. In many situations, however, each data point is a collection of numerical measurements and thus can be thought of as a point in an Euclidean space. In this case, similarity can be measured in terms of closeness in this space and it is chosen to be inverse-proportional to the Euclidian distance between x_i and x_j. This construction captures the local geometry of the data-set and it reflects the quantities of interest.

A diffusion process is used to propagate the local geometry in order to discover the global geometry of the data-set. Specifically, we build a diffusion operator whose eigen-decomposition enables the embedding of Γ into a space S of substantially lower dimension. The Euclidean distance between a pair of points in the dimension-reduced space defines a *diffusion metric* that measures the proximity of points in terms of their connectivity in the original space. Specifically, the Euclidean distance between a pair of points, in S, is equal to the random walk distance between the corresponding pair of points in the original space. The embedding of the data points into the low-dimension space provides coordinates on the data set that reorganize the points according to this metric.

The eigenvalues and eigenfunctions of P define a natural embedding of the data through the diffusion map and the study of the eigenvalues allows us to use the eigenfunctions for dimensionality reduction.

2.1 Building the graph G and the weight function w_ε

Let Γ be a set of points in $\mathbb{R}^n$ as defined in (1). We construct the graph $G(V,E)$, $|V| = m$, $|E| \ll m^2$, on Γ in order to study the intrinsic geometry of this set. A weight function $w_\varepsilon(x_i, x_j)$ which measures the pairwise similarity between the points is introduced.

For all $x_i, x_j \in \Gamma$, the weight function is chosen to obey the following properties:

- symmetry: $w_\varepsilon(x_i, x_j) = w_\varepsilon(x_j, x_i)$
- non-negativity: $w_\varepsilon(x_i, x_j) \geq 0$
- fast decay: given a scale parameter $\varepsilon > 0$, $w_\varepsilon(x_i, x_j) \to 0$ when $\|x_i - x_j\| \gg \varepsilon$ and $w_\varepsilon(x_i, x_j) \to 1$ when $\|x_i - x_j\| \ll \varepsilon$. .

Note that the parameter ε defines a notion of neighborhood i.e. for every point x_i, the weights $w_\varepsilon(x_i, x_j)$ are numerically significant only if x_j is ε-close to

x_i. In this sense, w_ε captures the local geometry of Γ by providing a first-order pairwise similarity measure for ε-neighborhood of every point x_i i.e it defines the nearest neighbor structures in the graph. This corresponds to the assumption that the only relevant information lies in local neighborhoods. Consequently, the matrix that represents w_ε is sparse.

One of the common choices for w_ε is

$$w_\varepsilon(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{\varepsilon}\right). \tag{2}$$

Indeed, x_i and x_j will be numerically significant if they are sufficiently close. In the case of a data set approximately lying on a submanifold, this choice corresponds to an approximation of the heat kernel on the submanifold (see (Belkin and Niyogi, 2003)). Furthermore, this result was extended in (Coifman and Lafon, 2006) where it was shown that any weight of the form $f(\|x_i - x_j\|)$ where f decays sufficiently fast at infinity allows to approximate the heat kernel . Thus, other application dependant weight functions can be used. Accordingly, this choice should take into account any prior knowledge on the data. For example, if the the data points are binary, the Hamming distance can be used instead of the Euclidean distance.

The choice of ε in (2) is extremely important since it defines the scale of the neighborhood. General guidelines as well as some data-driven heuristics for choosing ε are described later in this chapter.

2.2 Construction of the normalized graph Laplacian

The non-negativity property of w_ε allows us to normalize it into a Markov transition matrix P where the states of the corresponding Markov process are the data points. This enables to analyze Γ via a random walk. The construction of P is known as the *normalized graph Laplacian* (Chung, 1997).

Formally, $P = \{p(x_i, x_j)\}_{i,j=1,\ldots,m}$ is constructed as follows:

$$p(x_i, x_j) = \frac{w_\varepsilon(x_i, x_j)}{d(x_i)} \tag{3}$$

where

$$d(x_i) = \sum_{j=1}^{m} w_\varepsilon(x_i, x_j) \tag{4}$$

is the degree of x_i. P is a Markov matrix since the sum of each row in P is 1 and $p(x_i, x_j) \geq 0$. Thus, $p(x_i, x_j)$ can be viewed as the probability to move from x_i to x_j in a *single* time step. Raising this quantity to a power t advances the walk in time i.e. this probability is propagated to nodes in the neighborhood of x_i and x_j and the result is the probability to move from x_i to x_j in t time steps. We denote this probability by $p_t(x_i, x_j)$. These probabilities measure the connectivity of the points within the graph. The parameter t controls the scale of the neighborhood in addition to the scale control provided by ε.

2.3 Eigen-decomposition

As mentioned above the asymptotic behavior of the random walk entails the geometrical structure of the data set Γ. The close relation between the asymptotic behavior of P, i.e. the properties of its eigen-decomposition and the clusters that are inherent in the data, was explored in (Chung, 1997, Fowlkes *et al.*, 2004). As suggested in the example above, this behavior can be used to find clusters in the data set (Weiss, 1999, Shi and Malik, 2000), by using the first non-constant eigenvector to separate a data set into two clusters. One can separate to more than two clusters by using additional eigenvectors (Lafon and Lee, 2006, Meila and Shi, 2001, Yu and Shi, 2003).

Let $\{\mu_k\}_{k=1}^{m}$ and $\{\nu_k\}_{k=1}^{m}$ be the left and the right biorthogonal eigenvectors of P, respectively and let $\{\lambda_k\}_{k=1}^{m}$ be their corresponding eigenvalues where $|\lambda_1| \geq |\lambda_2| \geq ... \geq |\lambda_m|$.

Proposition 1. *If G is connected than the asymptotic behavior of the Markov process is dominated by a stationary distribution which is given by*

$$\lim_{t\to\infty} p_t(x_i, x_j) = \mu_1(x_j)$$

where μ_1 is the left eigenvector that corresponds to the highest eigenvalue.

Proof. Essentially, we need to prove that the stationary distribution exists and is equal to μ_1. First, it is easy to see that the state space of the Markov chain at hand is finite. This is since our data set is finite. Second, according to the Perron–Frobenius theorem, it is enough to show that the chain is irreducible and aperiodic.

Irreducibility: Let x_i and x_j be two data points. We denote by l the length of the path between them. The connectivity of the graph implies that l is finite and thus $p_l(x_i, x_j) > 0$. Thus, the chain is irreducible.

Aperiodicity: Although we require the weight function to be non-negative, most weight functions are actually positive and thus we get $w_\varepsilon(x_i, x_j) > 0$ which implies aperiodicity since $p(x_i, x_j) > 0$ as well. □

Proposition 2. *I. The eigenvector μ_1 can be derived from*

$$\mu_1(x_i) = \frac{d(x_i)}{\sum_{j=1}^{m} d(x_j)}.$$

II. From a pre-asymptotic point of view, for a finite time t we have

$$p_t(x_i, x_j) = \sum_{k=1}^{m} \lambda_k^t \nu_k(x_i)\, \mu_k(x_j)\,. \tag{5}$$

Proof. In order to prove this proposition construct a *symmetric* matrix A that is conjugate (see (Chung, 1997)) to P. The entries of A are obtained from the entries of P by

$$a(x_i, x_j) = \sqrt{\frac{d(x_i)}{d(x_j)}} p(x_i, x_j) = \frac{w_\varepsilon(x_i, x_j)}{\sqrt{d(x_i) d(x_j)}}.$$

We denote the eigenvectors of A by $\xi_1, \ldots, \xi_m$ and have

$$a(x_i, x_j) = \sum_{k=1}^{m} \lambda_k \xi_k(x_i) \xi_k(x_j) \tag{6}$$

It can be verified that A and P share the same eigenvalues and that their eigenvectors are connected by:

$$\mu_k(x_j) = \xi_k(x_j) \xi_1(x_j) \tag{7}$$

$$\nu_k(x_i) = \xi_k(x_j) / \xi_1(x_j) \tag{8}$$

for $i, j = 1, \ldots m$.

Furthermore, it is easy to check that $\xi_1(x_i) = \frac{\sqrt{d(x_i)}}{\sqrt{\sum_{j=1}^{m} d(x_j)}}$ which implies that $\nu_1(x_i) = 1$ and proves I:

$$\mu_1(x_i) = \frac{d(x_i)}{\sum_{j=1}^{m} d(x_j)}.$$

Combining (6), (7), (8) with

$$\mu_1(x_i) \nu_k(x_i) = \mu_k(x_i) \tag{9}$$

and raising A to the power t proves II

$$p_t(x_i, x_j) = \sum_{k=1}^{m} \lambda_k^t \nu_k(x_i) \mu_k(x_j)$$

with the following biorthogonality relation

$$\sum_{k=1}^{m} \nu_i(x_k) \mu_j(x_k) = \delta_{ij} \tag{10}$$

for $i, j = 1, \ldots m$, where δ_{ij} is the Kronecker symbol (Stewart, 2002). □

The matrix A plays an important role in the actual calculation of $\{\mu_k\}_{k=1}^{m}$, $\{\nu_k\}_{k=1}^{m}$ and $\{\lambda_k\}_{k=1}^{m}$. Algorithms for the eigen-decomposition of symmetric matrices are more accurate than those for non-symmetric matrices. Using (7) and (8) along with the fact that the eigenvalues of A and P are the same enables to perform the eigen-decomposition on A instead of P.

An appropriate choice of ε achieves a fast decay of $\{\lambda_k\}$. Thus, in order to achieve a relative accuracy $\delta > 0$, only a small number $\eta(\delta)$ of terms are required in the sum in (5). This enables us to introduce a *diffusion metric* based upon the following *diffusion distance* ((Coifman and Lafon, 2006))

$$D_t^2(x_i, x_j) = \sum_{k=1}^{m} \frac{(p_t(x_i, x_k) - p_t(x_j, x_k))^2}{\mu_1(x_k)}. \tag{11}$$

This formulation is derived from the known random walk distance in Potential Theory:

$$D_t^2(x_i, x_j) = p_t(x_i, x_i) + p_t(x_j, x_j) - 2p_t(x_i, x_j)$$

where the factor 2 is due to the fact that G is undirected.

The diffusion distance averages all the paths from x_i to x_j. Doing so measures the interaction of x_i and x_j with the rest of the graph by taking into account the connectivity of the points in the graph. This metric is more robust to noise and topological short-circuits than the geodesic distance or the shortest-path distance since it involves an integration along all paths of length t between x_i and x_j. Furthermore, classical graph theory notions such as mixing time and clusterness (Diaconis and Stroock, 1991) are also incorporated into this metric.

Finally, the low dimensional embedding is facilitated by the following proposition:

Proposition 3. *The diffusion distance can be expressed in terms of the right eigenvectors of P:*

$$D_t^2(x_i, x_j) = \sum_{k=1}^{m} \lambda_k^{2t} (\nu_k(x_i) - \nu_k(x_j))^2.$$

Proof. Combining (9) with (10) we get

$$\sum_{k=1}^{m} \frac{\mu_i(x_k)\,\mu_j(x_k)}{\mu_1(x_k)} = \delta_{ij}. \tag{12}$$

for $i, j = 1, \ldots m$. This implies that $\{\mu_i\}_{i=1}^m$ is an orthonormal system in the metric space $L_2(\Gamma, 1/\mu_1)$. By fixing x_k, we can view (5) as the decomposition of $p_t(x_i, \cdot)$ in this system where the decomposition coefficients are given by $\{\lambda_k^t \nu_k(x_i)\}_{k=1}^m$. This allows to rewrite (11) as

$$D_t^2(x_i, x_j) = \|p_t(x_i, \cdot) - p_t(x_j, \cdot)\|_{L_2(\Gamma, 1/\mu_1)}$$

which yields

$$D_t^2(x_i, x_j) = \sum_{k=1}^{m} \lambda_k^{2t} (\nu_k(x_i) - \nu_k(x_j))^2 \square \tag{13}$$

Notice that the sum in (13) can start from 2 since $\nu_1(x_i) = 1$ as it was seen in proposition 1, and therefore does not help to differentiate between two distances. It follows that in order to compute the diffusion distance, one can simply use the right eigenvectors of P. Moreover, this facilitates the embedding of the original points in a Euclidean space $\mathbb{R}^{\eta(\delta)-1}$ by:

$$\Xi_t : x_i \rightarrow \left(\lambda_2^t \nu_2(x_i), \lambda_3^t \nu_3(x_i), \ldots, \lambda_{\eta(\delta)}^t \nu_{\eta(\delta)}(x_i)\right).$$

This also provides coordinates on the set Γ. Essentially, $\eta(\delta) \ll n$ due to the fast decay of the eigenvalues decay of P and it depends only on the primary intrinsic variability of the data as captured by the random walk and not on the original dimensionality of the data. Furthermore, this data-driven method enables the parametrization of any set of points – abstract or not – provided the similarity matrix of the points w_ε is available.

Figure 1 illustrates the results of the diffusion maps algorithm applied on a data-set of digit images. The data-set consisted of five hundreds 30×20 binary images of the digit nine. Each vector in this data-set was in $\mathbb{R}^{600}$. The data-set was embedded into $\mathbb{R}^2$ and recovered two governing features of the digit: the *rotation angle* of the digit, which is given by the x-axis, and the *height* of the digit, which is given by the y-axis.

An example of data clustering using diffusion maps is given in Fig. 2. The diffusion maps algorithm was applied on a data set consisting of five hundreds 30×20 binary images of the digit nine and five hundreds 30×20 binary images of the digit zero. The embeddings of the digit nine are illustrated by circles while the embeddings of the digit zero are illustrated by plus signs.

3 Choosing ε

The size of the local neighborhood of each point is determined by ε. A large ε defines a wide neighborhood and thus producing a coarse analysis of the data since most neighborhoods will contain a large number of points. In contrast, for a small ε many neighborhoods will contain a single point. Clearly, an appropriate ε should be between these two cases and should stem from the data. Let $D = \{d_{ij}\}_{i,j=1,\ldots,m}$ be the pairwise Euclidean distance matrix between the points in Γ. Two heuristics are proposed:

The median heuristic: $\varepsilon = \mathit{median}\,\{d_{ij}\}_{i,j=1,\ldots,m}$.

The median of D provides an estimate to the average pairwise distance that is robust to outliers.

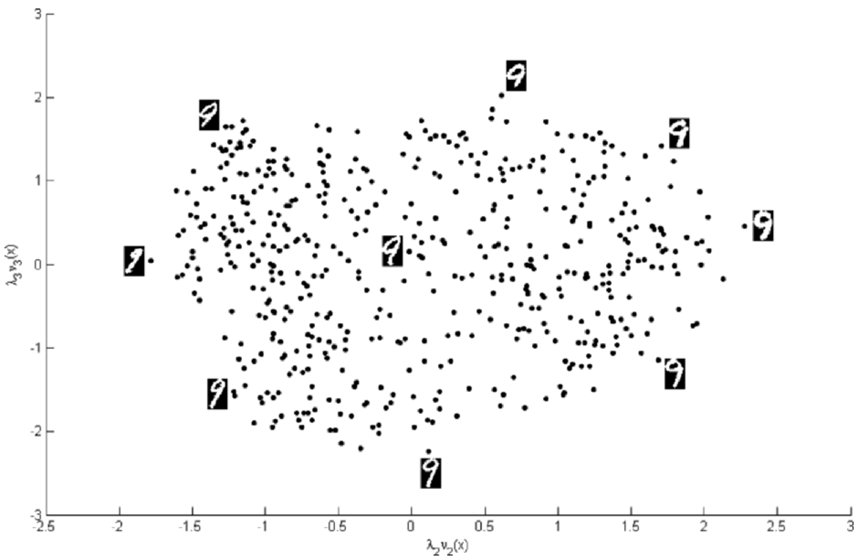


Fig. 1. Embedding of five hundreds 30×20 binary images of the digit nine (each is in $\mathbb{R}^{600}$) into $\mathbb{R}^2$. The *rotation angle* of the digit is given by the x-axis and the *height* of the digit is given by the y-axis

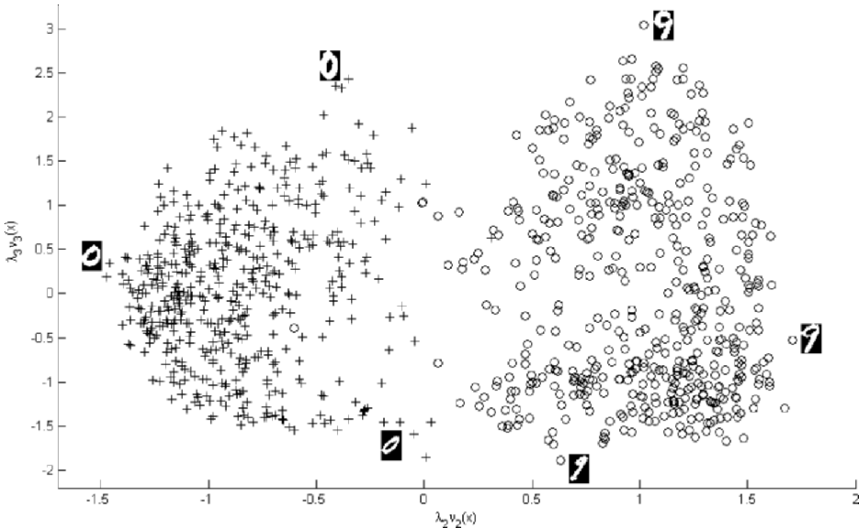


Fig. 2. Embedding of five hundreds 30×20 binary images of the digit nine and five hundreds 30×20 binary images of the digit zero into $\mathbb{R}^2$. The embeddings of the digit zero are illustrated by plus signs while the embeddings of the digit nine are illustrated by circles

The max-min heuristic: $\varepsilon = \alpha \cdot \max_i \min_j D$

In this case, $\min_j D$ denotes a column vector consisting of the distance of each point to its closest neighbor. Taking $\alpha \geq 1$, verifies that each neighborhood contains at least one neighbor.

4 Conclusion

In this chapter we saw that diffusion processes prove to be an effective tool for data analysis. Given a data set, a Markov process is constructed where the states are composed of the data points and the transition probabilities are determined according to a pairwise similarity measure. The similarities are only locally determined and they are propagated via the diffusion process. Using the spectral properties of the Markov matrix that is associated with the Markov process, we are able to reduce the dimensionality of the original data set and embed it into a space of substantially lower dimension. The diffusion distance that was introduced in this chapter measures the level of connectivity between a pair of points in the original space. After the dimensionality reduction, the Euclidean distance between a pair of points in the lower dimensional space approximates the diffusion distance between these points in the original space.

5 Further reading

A more comprehensive description of the diffusion maps scheme can be found in (Coifman and Lafon, 2006, Coifman *et al.*, 2005). A few of the successful application of the diffusion maps algorithm include lip reading (Lafon *et al.*, 2006), text classification (Lafon and Lee, 2006), segmentation of multi-contrast MRI images (Shtainhart *et al.*, 2007). The Diffusion bases algorithm – a dual algorithm to the diffusion maps algorithm – was recently introduced in (Schclar and Averbuch, 2007). Laplacian eigenmaps – a dimensionality reduction algorithm that is closely related to the diffusion maps – is described in (Belkin and Niyogi, 2003). Several papers explored the connections and applications of the graph Laplacian to machine learning (for example, (Kondor and Lafferty, 2002)). A generalization of classical wavelets which is based on diffusion processes, allowing multiscale analysis of general structures, such as manifolds, graphs and point clouds in Euclidean space, is introduced in (Coifman and Maggioni, 2006).

References

M. Belkin and P. Niyogi. (2003), Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396.

F. R. K. Chung. (1997), *Spectral Graph Theory.* AMS Regional Conference Series in Mathematics, 92.

R. R. Coifman, S. Lafon, A. Lee, M. Maggioni, B. Nadler, F. Warner, and S. Zucker. (2005), Geometric diffusions as a tool for harmonics analysis and structure definition of data: Diffusion maps. In *Proceedings of the National Academy of Sciences*, volume 102, pages 7432–7437.

R. R. Coifman and S. Lafon. (2006), Diffusion maps. *Applied and Computational Harmonic Analysis: special issue on Diffusion Maps and Wavelets*, 21:5–30.

R. R. Coifman and M. Maggioni R. R. Coifman and M. Maggioni. (2006) Diffusion wavelets. *Applied and Computational Harmonic Analysis: special issue on Diffusion Maps and Wavelets*, 21(1):53–94.

P. Diaconis and D. Stroock. (1991), Geometric bounds for eigenvalues of markov chains. *The Annals of Applied Probability*, 1(1):36–61.

C. Fowlkes, S. Belongie, F. Chung, and J. Malik. (2004), Spectral grouping using the nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):214–225.

R. I. Kondor and J. D. Lafferty. (2002), Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the 19th International Conference on Machine Learning (ICML 02)*, pages 315–322.

S. Lafon Y. Keller and R. R. Coifman. (2006), Data fusion and multi-cue data matching by diffusion maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1784–1797.

S. Lafon and A. Lee. (2006), Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1393–1403.

M. Meila and J. Shi. (2001), A random walk's view of spectral segmentation. In *Proceedings of the International Workshop on Artifical Intelligence and Statistics.*

A. Schclar and A. Averbuch. (2007), Hyper-spectral segmentation via diffusion bases. *Technical report, Tel Aviv University.*

S. M. Sheldon. (1983), *Stochastic Processes.* John Wiley & Sons.

J. Shi and J. Malik. (2000), Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905.

A. Shtainhart, A. Schclar, and A. Averbuch. (2006), Neuronal tissues sub-nuclei segmentation using multi-contrast mri. *Technical report, Tel Aviv University.*

J. Stewart. (2002), *Calculus.* Brooks Cole, 5th edition.

Y. Weiss. (1999), Segmentation using eigenvectors: A unifying view. In *ICCV (2)*, pages 975–982.

S. X. Yu and J. Shi. (2003), Multiclass spectral clustering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 313–319.

Data Mining and Agent Technology: a fruitful symbiosis

Christos Dimou[1], Andreas L. Symeonidis[1,2], and Pericles A. Mitkas[1,2]

[1] Electrical and Computer Engineering Dept.
Aristotle University of Thessaloniki, 54 124, Thessaloniki, Greece
[2] Intelligent Systems and Software Engineering Laboratory,
Informatics and Telematics Institute/CERTH, 57 001, Thessaloniki, Greece
cdimou@issel.ee.auth.gr, asymeon@iti.gr, mitkas@eng.auth.gr

Summary. Multi-agent systems (MAS) have grown quite popular in a wide spectrum of applications where argumentation, communication, scaling and adaptability are requested. And though the need for well-established engineering approaches for building and evaluating such intelligent systems has emerged, currently no widely accepted methodology exists, mainly due to lack of consensus on relevant definitions and scope of applicability. Even existing well-tested evaluation methodologies applied in traditional software engineering, prove inadequate to address the unpredictable emerging factors of the behavior of intelligent components. The following chapter aims to present such a unified and integrated methodology for a specific category of MAS. It takes all constraints and issues into account and denotes the way knowledge extracted with the use of Data mining (DM) techniques can be used for the formulation initially, and the improvement, in the long run, of agent reasoning and MAS performance. The coupling of DM and Agent Technology (AT) principles, proposed within the context of this chapter is therefore expected to provide to the reader an efficient gateway for developing and evaluating highly reconfigurable software approaches that incorporate domain knowledge and provide sophisticated Decision Making capabilities. The main objectives of this chapter could be summarized into the following: a) introduce Agent Technology (AT) as a successful paradigm for building Data Mining (DM)-enriched applications, b) provide a methodology for (re)evaluating the performance of such DM-enriched Multi-Agent Systems (MAS), c) Introduce Agent Academy II, an Agent-Oriented Software Engineering framework for building MAS that incorporate knowledge model extracted by the use of (classical and novel) DM techniques and d) denote the benefits of the proposed approach through a real-world demonstrator. This chapter provides a link between DM and AT and explains how these technologies can efficiently cooperate with each other. The exploitation of useful knowledge extracted by the use of DM may considerably improve agent infrastructures, while also increasing reusability and minimizing customization costs. The synergy between DM and AT is ultimately expected to provide MAS with higher levels of autonomy, adaptability and accuracy and, hence, intelligence.

1 Introduction

Large amounts of data are being produced and made available online every day, pushing user needs towards a more knowledge-demanding direction. Today's applications are therefore required to extract knowledge from large, often distributed, repositories of text, multimedia or hybrid content. The nature of this quest makes it impossible to use traditional deterministic computing techniques. Instead, various soft computing techniques are employed to meet the challenge for more sophisticated solutions in knowledge discovery. Most notably, Data Mining (DM) is thought of as one of the state-of-the-art paradigms. DM produces useful patterns and associations from large data repositories that can later be used as *knowledge nuggets*, within the context of any application.

Individual facets of knowledge discovery, introduced by DM techniques, often need to be orchestrated, integrated and presented to end users in a unified way. Moreover, knowledge has to be exploited and embodied in autonomous software for learning purposes and, hence, a more increased performance (Figure 1). Agent Technology (AT) proves to be a promising paradigm that is suitable for modelling and implementing the unification of DM tasks, as well as for providing autonomous entity models that dynamically incorporate and use existing knowledge. Indeed, a plethora of multi-agent systems (MAS) and other agent-related solutions for knowledge-based systems can be found in the literature, and more specifically in the area of agent-based DM, as it is explained in detail in Section 3 of this chapter.

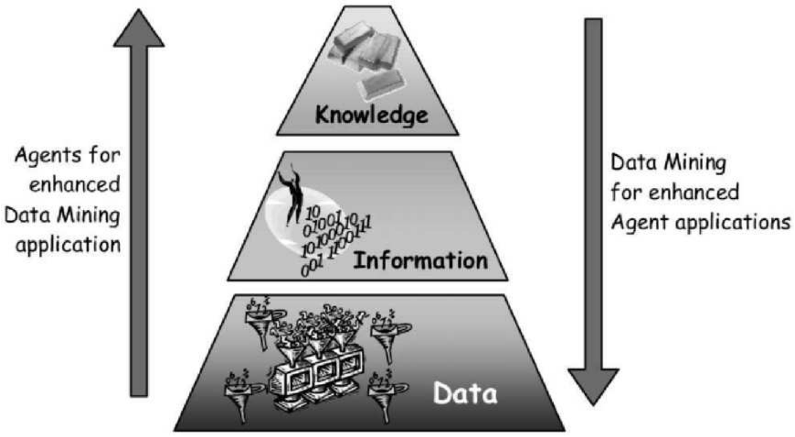

Knowledge
Information
Data
Agents for enhanced Data Mining application
Data Mining for enhanced Agent applications

Fig. 1. Mining for intelligence

A numerous related agent development methodologies deal with most of the steps of the development lifecycle. However, there is a remarkable lack of generalized evaluation methodologies for the systems in question. The evaluation of performance is a fundamental step of any development methodology, which provides developers with countable, qualitative and verifiable attributes in an effort to better comprehend the nature of the system at hand. Additionally, generalized and standardized evaluation procedures allow third parties to safely verify the acclaimed properties of deployed systems or newly discovered scientific results.

Existing evaluation approaches address either the DM algorithmic issues or the overall system performance. Both approaches come short in the case of AT and DM integration, due to the complex and dynamic nature of the produced systems. In the case of DM evaluation, focus is given on the statistical performance of individual techniques, in terms of precision and recall, ignoring the actual impact of the extracted knowledge to the application level. In the case of overall system evaluation, existing methods fail to deal satisfactorily with emergent agent behaviors that may not be known at design time.

In this chapter, we present an integrated software engineering approach for developing DM-enriched MAS. Having *Agent Academy II* as the basic designing, development and agent training framework for MAS that employ DM, we provide a generalized methodology for evaluating the performance of a developed system. A set of consice methodological steps is presented, focusing on three fundamental evaluation aspects, namely the selection of a) metrics, b) measurement method, and c) aggregation methods. The proposed methodology is designed to assist developers as an off-the-shelf tool that can be integrated in the overall system development methodology.

The remainder of this chapter is organized as follows: in Section 2 an overview of the basic primitives of AT and DM is provided; Section 3 reviews the related literature in DM and MAS integration and evaluation; Section 4 presents the core development and evaluation methodology, by outlining the appropriate theoretical and software tools; in Section 5, Agent Academy II, a development framework for DM-enriched MAS is presented; finally, Section 6 summarizes and discusses related concluding remarks.

2 Agent Technology and Data Mining

AT and DM have been incorporated and integrated in numerous research efforts. However, the disparity of applications and the notable diversity of the nature of these technologies motivate us to provide thorough definitions of relevant terms and present our point of view over their symbiosis.

2.1 Agents

The term "software agent" has been coined since the early years of Artificial Intelligence, in order to denote any software module that exhibits intelligent behavior. This vague definition, in combination with the unfeasible visions of early AI, has resulted into unsatisfactory applications and has decreased agent computing popularity for many years. It is only until recently that interest in software agents has revived within the context of complex systems' engineering. Agents appear as a handy modelling concept for autonomous, decentralized entities, cooperating towards a common goal or competing on limited resources. Moreover, the advent of many popular, highly distributed internet applications has reinforced agents' position as a promising paradigm for addressing the emerging engineering problems (Weiss (2001)) (Jennings (2001)) (Foster et al.(2004)) (Greenwald and Stone (2001)).

In practice, however, no single universally agreed-upon definition of a software agent exists. This problem occurs due to the horizontal nature of agents. Agents can be either abstract tools for modelling complex systems or actual implemented software modules that may perform any task. The vast disparity of application domains on which agents have been applied reinforces the definition difficulties. In some sets of applications agents may need to exhibit decision making behaviors, whereas in other cases agents may be assigned routine, predefined tasks. Various abstract definitions for agents have been proposed, focusing on one or more agent characteristics with respect to one or more application domains. Woolridge and Jennings, for example, define a software agent with respect to situatedness, autonomy and goal-orientation.

In general, an agent is a software entity that exhibits some or all of the following characteristics:

1. *Autonomy*: Considered as one of the most fundamental features of agency, autonomy implies the degree of control that an agents poses on its own execution thread. Autonomy is usually a strong requirement in many application domains and therefore agents often employ relevant techniques for task-wise decision making in their effort to accomplish their goals.
2. *Interactivity*: Agents are seldom stand alone. They most often rely in information rich and eventful environments with other agents, services or human users. It is therefore required for agents to possess corresponding sensors for perceiving information and actuators for changing the environment. Moreover, interactivity emerges as a result of a more complex intrinsic behavioral processing of the environmental input. Therefore, agents may either respond to occurring events (*reactiveness*) or take initiative and act in order to accomplish predefined goals (*proactiveness*).
3. *Adaptability*: In dynamically changing environments, agents need to be able to change their internal states and consequent actions, to better match the ever-changing conditions.
4. *Sociability*: A product of interactivity, socialability is based on relative human social skills, such as the ability to determine trusted parties or form coalitions in unknown environments.
5. *Cooperativity*: In applications such as distributed problem solving, agents are often required to collaborate with each other in order to reach a common goal that otherwise would be impossible or impractical to reach.

Cooperation may be coordinated by dedicated agents or, in more open environments, emerge via agent communication.
6. *Competitiveness*: Agents are often programmed to allocate certain resources in well-defined environments. Like in similar real-world scenaria, the resources are limited and therefore agents need to compete against each other in order to allocate such resources. Agents employ strategies and action plans for prevailing in such competitive environments.
7. *Mobility*: Transition between different environments is a desired property of agents in applications such as data gathering, web crawling etc.
8. *Character*: Human-centered characteristics can be summarized as a character that is embodied in agents, most often in interface, personal, or assisting agents.
9. *Learning*: Learning is an all-encompassing term that utilizes some or all of the above properties, so that agents observe the impact of theirs or other agents' actions on the environment and predict the optimal behavior, that is activated in similar situations in the future.

Alternative definition approaches attempt to classify agents with respect to their application domain. Such domains may include *inter alia* searching and filtering of information, monitoring conditions, alerting, proxing, coordinating network tasks and managing resources.

A robust classification of agents is provided by (Nwana (1996)). According to this approach, agents can be classified with respect to the following dimensions:

1. *Mobility*, that differentiates agents into *static* and *mobile*
2. The *logic paradigm* they employ, which classifies them as either *deliberative* or *reactive*
3. The *fundamental characteristic* that describe the agent (autonomy, cooperativity, learning). Based on these axes, agents can be classified as (Figure 2):
   - collaborative agents
   - collaborative learning agents
   - interface agents
   - smart agents

In this work, we have adopted Nwana's classification scheme, since it covers successfully a wide area of agent-related applications, as well as it proves to be robust enough to meet the needs of a large number of researchers in the AT field.

2.2 Multi-Agent Systems (MAS)

The promising properties of agents discussed above can only be fully exploited in complex architectures that are deployed in a systematic way. *Agent Oriented Software Engineering* (AOSE) (Jennings (1999)) (Perini et al. (2001)) tackles

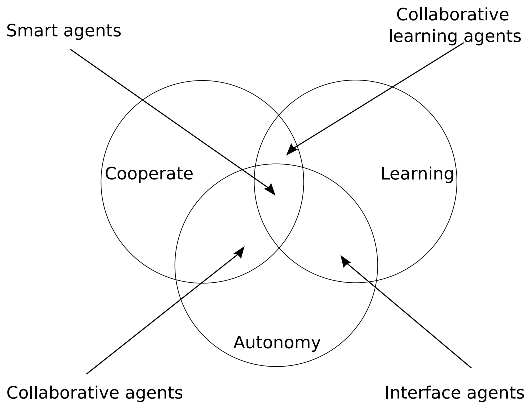


Fig. 2. Nwana's classification of agents

this challenge by providing high level abstraction, modelling and development tools, grouped under the umbrella term of Multi-Agent Systems (MAS). A MAS is as a systematic solution to inherently distributed problems by utilizing autonomous interacting agents and appropriate communication protocols. MAS integrate the above mentioned agent properties and provide new system features that would otherwise be impossible to achieve using monolithic systems. Such characteristics include:

- Increased *performance*, *reliability* and *maintenability*.
- *Modularity*, *flexibility* and *extensibility*.
- Overcome of *limited scope* and *capacity* of single agents in terms of knowledge and tasks.
- Better support for *dynamic*, *unpredictable environments*.
- Introduction of *collective intelligence*, that is the augmentation of intelligence of many simple, not-so-smart interacting entities in contradiction to single, centralized sophisticated modules.

Distributed systems, in the above sense, are modelled as a network of autonomous entities that regulate, control and organize all activities within the distributed environment. In the recent years, a plethora of such distributed systems has emerged. Most notably, the Grid paradigm (Foster et al.(2001)) that envisions a transparent infrastructure of high performance computer resources or knowledge resources that is scattered throughout the globe. Since the conceptualization of the Grid, it has become apparent that MAS may play a pivotal role in the modelling and implementation of such systems (Foster et al.(2004)) (Tang and Zhang (2006)). Other agent-based popular applications include Web Services (Gibbins et al. (2003)) (Muller et al. (2006)) (Negri et al. (2006)), Web Crawling (Jansen et al. (2006)) (Dimou et al.(2006)), environmental monitoring (Athanasiadis and Mitkas (2005)) (Purvis et al. (2003))

and virtual market places (Wurman et al. (1998)) (Wellman and Wurman (1998)).

MAS Characteristics

In general, MAS adhere to the following three primitives. First, MAS must specify appropriate communication and interaction protocols. Despite agents being the building blocks of a problem solving architecture, no individual problem could be effectively solved if no common communication ground, and no action protocol exists.

Secondly, MAS must be open and decentralized. No prior knowledge of, for example, number of participants or static behaviors are always known to the system developer. In a running MAS, new agents may join at any time having only to comform to the communication protocol, being able to act on the way they choose, often in unpredictable manner.

Finally, MAS must consist of possibly heterogeneous agents that are scattered around the environment and act autonomously or in collaboration.

2.3 Data Mining

The need for methods for discovering useful information in large data volumes has been a vivid research topic for many years. Especially nowdays, this need is imperative due to the increasing rate of data production, intensified by the ever-increasing demand for information. DM is a relatively new approach to this problem, often denoted as Knowledge Discovery in Databases (KDD), deals with this exact problem: "the extraction of interesting, non-trivial, implicit, previously unknown and potentially useful information or patterns from data in large databases" (Fayyad et al.(1996)). Though a large number of patterns may arise from the application of DM on datasets, only interesting ones are selected, that is patterns that can be easily understood by humans and suitable for validating a user hypothesis.

Other researchers argue that DM is only "the most important step in the KDD process and involves the application of data analysis and discovery algorithms that, under acceptable computational efficiency limitations, produce a particular enumeration of patterns of data" (Allard and Fraley (1997)). Either approach adopted, in essence DM and KDD address the same problem of extracting useful knowledge and, within the context of this chapter, integrate this knowledge into MAS.

KDD process

KDD is the iterative traversal of the list of steps presented in Table 1

One or more steps of the KDD process may be repeated as many times deemed necessary, in order to come up with desirable outcome.

Table 1. Steps of the KDD process

1. Identify the goal of the KDD process
2. Create a target dataset
3. Clean and preprocess data
4. Reduce and project data
5. Identify the appropriate DM method
6. Select a DM algorithm
7. Apply DM
8. Evaluate DM results
9. Consolidate discovered knowledge

Data Mining Techniques

DM techniques may be applied on large data sets either for validation of a hypothesis or for discovering new patterns. The latter case is further divided into prediction of future trends of the data fluctuation and to a more detailed description and understanding of already extracted patterns. Within the scope of this chapter, focus is given on discovering new patterns and the associated DM techniques include (Fayyad et al.(1996)):

1. *Classification*: the discovery of knowledge model that classifies new data into one of the existing pre-specified classes.
2. *Characterization and Discrimination*: the discovery of a valid description for a part of the dataset.
3. *Clustering*: the identification of finite number of clusters that group data based on their similarities and differences.
4. *Association-Correlation*: the extraction of association rules, which indicate cause-effect relations between the attributes of a dataset.
5. *Outlier analysis*: the identification and exclusion of data that do not abide by the behavior of the rest of the data records.
6. *Trend and evolution analysis*: the discovery of trends and diversions and the study of the evolution of an initial state/hypothesis throughout the course of time.

2.4 Integrating Agent Technology and Data Mining

AT and DM are two separate vessels that each, as derived from the above, has its own scope and applicability. The idea of combining these diverse technologies is emerged by the need to either a) enrich autonomous agents by employing knowledge derived from DM or b) utilize software agents to assist the data extraction process. Both aspects are intriguing and challenging, mainly because of the disparity of AT and DM, since:

1. Despite logic being their common denominator, AT and DM employ two complementary logic paradigms. Agent reasoning is usually based on deductive logic, whereas DM embraces inductive logic.

2. Categorization of MAS with respect to DM-extracted knowledge is complicated due to the wide application range of both technologies.

Logic paradigms

In deductive inference, conclusions are drawn by the combination of a number of premises. Thus, knowledge models are applied to data producing knowledge information (Fernandes(2000)). Under the assumption that these premises are true, deductive logic is truth preserving. In MAS applications, deduction is used by agents in a form of predefined rules and procedures usually defined by domain experts. These rules specify agent actions with respect to the input sensed. Nevertheless, deduction proves inefficient in complex and versatile environments (Arthur (1994)) (Wooldridge (1999)).

Inductive inference, on the other hand, attempts to transform specific data and information into concrete, generalized knowledge models. During the induction process, new rules and correlations are being produced aiming at validating each hypotheses. In contradiction to deduction, induction may lead to invalid conclusions, as it only uses progressive generalizations of specific examples (Kodratoff (1988)).

It becomes evident that the coupling of the above two approaches under the combination of the carrying technologies leads to enhanced and more efficient reasoning systems as proved by (Symeonidis and Mitkas (2005)). Indeed, this combination overcomes the limitations of both paradigms by using deduction for well-known procedures and induction of discovering previously unknown knowledge. The processes of agent training and knowledge diffusion are further explained in the remainder of this section.

Agent modelling

Knowledge extraction capabilities must be present in agent design, as early as in the agent modelling phase. During this process, the intended DM techniques employed shape the nature of the reference engine and provide the knowledge model of the agent with required useful patterns. Every agent exhibiting such reasoning capabilities is required to have the internal structure outlined below.

- Application domain
- Domain ontology
- Agent shell
- Agent type
- Behavior type
- Knowledge Model
- Reasoning engine

When the DM-generated knowledge model is incorporated to the otherwise dummy agent, the outer layers, namely domain ontology, agent shell, agent type and behavior type consist the functional parts of the agent that are influenced in corresponding manners.

The process of dynamically incorporating DM-extracted knowledge models (KM) into agents and MAS is defined as "agent training" while the process of revising in order to improve the knowledge model of agents by reapplying DM techniques is defined as retraining. Finally, "knowledge diffusion" is defined as the outcome of the incorporation of DM extracted knowledge to agents.

Three levels of diffusion

Knowledge diffusion is instantiated in three different ways, with respect to the alternative targets of DM:

1. *DM on the application level of MAS.* DM is applied in order to find useful rules and associations in application data, in order to provide knowledge nuggets to the end user, independently of the internal architecture.
2. *DM on the behavioral level of MAS.* DM is applied on behavioral data, usually log files with past agent actions, in order to predict future agent behaviors.
3. *DM on the evolutionary agent communities.* Evolutionary DM is performed on agent communities in order to study agent societal issues. According to this societal point of view, the goals that need to be satisfied are not atomic but collective and, therefore, knowledge is fused into agents that share common goals.

3 Related work

3.1 MAS and DM

The integration of AT and DM is a subject of a number of research efforts that can be found in the literature. In (Galitsky and Pampapathi(2003)), a combination of inductive and deductive logic for reasoning purposes is proposed for improved customer relationship management. In this work, deduction is used when complete information is available, whereas induction is employed to forecast behaviors of customers when the available information is incomplete. (Fernandes(2000)) provides an implementation of a single inference engine for agents that uses both inductive and deductive reasoning. In this work, logic terms of model data, information and knowledge are incorporated and processed by deductive agents. Finally, an integration of deductive database queries and inductive analysis on these queries and their produced knowledge is presented in (Kero et al. (1995)).

It is a frequent observation in MAS applications that a tradeoff between inference (either inductive or deductive), complexity and development cost arises. However, in more dynamic environments, where both requirements and agent behaviors need constant modification, a systematic approach is compulsory. Symeonidis and Mitkas (Symeonidis and Mitkas (2005)) present a unified methodology for transferring DM extracted knowledge into newly created agents. Knowledge models are generated through DM on the various levels of knowledge diffusion and are dynamically incorporated in agents. The iterative process of retraining through DM on newly acquired data is employed, in order to enhance the efficiency of intelligent agent behavior. The suggested methodology is thoroughly tested on three diverse case studies.

3.2 Evaluation

Although promising, the integration of DM results into MAS functionality arises interesting and some times crucial issues, as far as safety and soundness is concerned. Seeking to extend the work of (Symeonidis and Mitkas (2005)) and provide an evaluation framework for agent efficiency, we present a literature review on intelligent agent evaluation.

Evaluation is a vital step in any complete scientific and engineering methodology. It is defined as "the process of examining a system or a system component to determine the extend to which specified properties are present"[1]. It is the most powerful and sound tool for researchers to assess the quality and applicability of their findings, as well as to set the limits and the optimal environmental or intrinsic system parameters for optimal performance.

Within the Soft Computing paradigm, evaluation is an all encompassing term that may address any component of the system at hand, heavily depending on the developer's aims. It is therefore the researcher's choice to focus on: a) algorithmic issues, b) system performance evaluation, or c) observable intelligence of the implemented system. By addressing one or more of the above, a researcher is able to isolate theoretical shortcoming or implementation malpractices, comprehend the intrinsic characteristics of the system and improve it in the most beneficial manner.

Algorithmic performance and quality evaluation in the context of Soft Computing has been an issue covered by a large corpus of work in the literature, that especially draws from the information retrieval theory primitives. In such cases, the algorithm employed is assessed against its ability to extract the largest percent possible of useful information. Common metrics in this direction include precision, recall, fallout, F-measure and mean average precision. Other Soft Computing approaches use other appropriate metrics or aggregation techniques, depending on the case, including ROC curves, fitness functions and composite figure of merits. Algorithmic evaluation is an important tool that gives a summary of the black box implementation of any algorithm.

[1] The Free Online Dictionary of Computing, September 2003 (http://www.foldoc.com)

In the case of intelligent systems, such as MAS, instead of evaluating individual algorithms, we need to assess the actual impact of the employed techniques with respect to other engineering aspects, such as integration issues, performance issues and impact of the selected techniques to the overall quality of the outcome. Moreover, emergent, unpredictable and intelligent behavior that often is exhibited by such systems complicates the process of defining and realizing evaluation procedures. For instance, a bidding agent may use a DM knowledge extraction algorithm from historical data. Despite the satisfaction of high algorithmic precision, the agent may end up losing all auctions because of inefficiency in timing and/or overall strategy. We, therefore, need to regard the system at hand as an integrated intelligent system that consists of modules, exhibits certain behaviors and aims at the accomplishment of specific goals.

In the literature, two general research approaches towards the direction of engineering aspects evaluation exist: a) bottom-up and b) top-down. The first approach represents the strong AI perspective on the problem, indicating that intelligent systems may exhibit any level of intelligence comparable to human abilities. Zadeh (Zadeh (2002)) argues that evaluating such systems is infeasible today, due to the lack of powerful formal languages for defining intelligence and appropriate intelligent metrics. The second approach represents the weak AI or engineering perspective, according to which intelligent systems are systems of increased complexity that are nevertheless well-defined in specific application domains, designed for solving specific problems. Albus (Albus et al. (2000)) suggests that intelligent performance can be effectively evaluated after a concise decomposition of the problem scope and definitions of relative metrics and measurement procedures. Driven by the urging need to evaluate and compare existing or emergent applications, we adopt the top-down approach.

It should be denoted at this point that no general, complete methodology for evaluating engineering applications exists. Instead, researchers often have to devise their own ad-hoc metrics and experimental procedures. In fact, in some cases, the chosen parameters or input data are chosen so as to produce the best results for the -each time presented- method. Moreover, the findings are often supported by qualitatively arguments only, in favor of the proposed system and no debate with respect to its drawbacks is provided. Consequently, it is impossible for a third party to repeat the evaluation procedure and validate the quality of the proposed solution by concluding to similar results. The need for a generalized evaluation framework is, thus, evident.

The requirements of such a methodology is to address both system performance issues as well as emergent, intelligent atomic and social behavior. To answer this challenge, we must devise a methodology that focuses on the following:

- *Re-usability*: The provided methodology must be domain independent and must be available for reuse under different application scenaria, always taking into account possible inherent or emergent heterogeneity in different implementations.

- *Qualitative comparability*: Different implementations in a specific application domain must be liable to comparison with respect to a set of selected qualitative features.
- *Quantitative assessment*: Different implementations in a specific application domain must also be liable to comparison with respect to a set of selected quantitative criteria. Additionally, there must be the opportunity of defining optimal or desired performance, against which one or more implementations may be compared.

Software engineering evaluation, as a well established field, consists a major source of background theory and tools towards this direction. Complete methodologies with quantitative and qualitative metrics have been developed and used in actual software projects. Although subsets of metrics and methods may be adopted, these approaches do not suffice for evaluating intelligent systems, since standard software evaluation processes focus in product features and do not always take into account emergent and unpredictable system behavior.

Ongoing efforts for generalized metrics and evaluation methodologies exist in application fields, such as robotics and autonomic computing. In robotics, evaluation efforts span from autonomous vehicle navigation (Nelson et al. (2002)) (Hu and Zeigler (2002)) (Zimmerman et al. (2002)) to hybrid human-robot control systems (Scholtz et al. (2002)) (Burke et al.(2002)) (Goodrich et al. (2002)). In autonomic computing, emphasis is given to the quality assessment of the selected self-managing techniques (Huebscher and McCann (2004)). Both fields provide us with usefull metrics and thorough methodological steps. However, neither of the above approaches are complete and mature nor do they provide us with relevant tools for the case of knowledge infusion in autonomous entities.

4 A methodology for designing and evaluating DM-enriched applications

In this section we present a generalized methodology for comprehensive development of DM-enriched Multi-Agent Systems. While a typical designing methodology comprises many parts, we focus mainly on the evaluation part of our methodology, for two reasons. First, a multiplicity of designing methodologies exists in the area of MAS and any of them could be used and tailored to model and implement DM enriched applications. Second, there is a remarkable lack of evaluation methods and tools in the area of MAS. By first outlining the evaluation requirements of our proposed approach, we imply the designing requirements that can be met by some of the existing designing methodologies.

The proposed evaluation methodology serves as an off-the-shelf tool for researchers and developers in this field. Composed of both theoretical analysis and software tools, it provides guidelines and techniques that can be used, adopted or extended for the application domain at hand. We follow the top-down engineering perspective, as proposed by Albus (Albus et al. (2000)) and described in the previous section. The methodology is therefore applicable to existing applications or applications that meet current agent oriented engineering concepts and follow the definitions for agent systems and DM terms provided in previous sections.

In our approach, evaluation derives from the observable behavior of agents within MAS. We therefore consider agents to be black boxes at different levels of granularity, depending on the application scope and the evaluation needs of the designer. In a *fine-grained* level, it is desirable to isolate single agents and observe their efficiency and impact of their actions on their environment. In a *coarse-grained* level, we focus on the overall behavior of an agent society and the outcome of transparent collaborative problem solving, competition or negotiation. In the former case, we consider each participating agent as a black box, whereas in the latter case we view the entire MAS as a black box. In both cases, the methodology isolates and measures certain characteristics of the observable behavior of each black box.

By this approach, we assess the impact of the actual performance of agents and MAS, bypassing the methods of implementation, algorithms and other intrinsic characteristics. This implementation independence of our methodology, makes it a powerful tool for measuring both directly the actual efficacy of a system as a whole and indirectly the performance of the intrinsic methods employed. In other words, the algorithmic decisions are indirectly handled and revised in an iterative manner, if and only if the results of the observable behavior evaluation are not within accepted limits. Moreover, two differently implemented systems can be compared against each other, solely in terms of efficiency, having the underlying mechanisms implicitly compared at the same time.

For establishing an evaluation framework that meets the above characteristics, we define:

- *Horizontal aspects*, the essential methodological steps, that if followed sequentially in an iterative manner, will comprise a complete evaluation methodology. The horizontal aspects of our methodology are:
  - *Definitions and theoretical background* on evaluation terms and relevant techniques.
  - *Theoretical tools* that can help designers chose what to measure, how to measure and how to integrate specific findings.
  - *Software tools* that assist designers in their experimental measurements.

- *Vertical aspects* are specific techniques that may be part of any of the above horizontal aspects and deal with the following three terms (Shih (2000)):
  - *Metrics* that correspond to system features to be measured.
  - *Measurement methods* that define the actual experimental procedure of assigning measurement values to the selected metrics.
  - *Aggregation* of the metric-measurement pairs in single characterizations for the system.

In the remainder of this section, we examine the above mentioned horizontal aspects in turn, analyzing each of their vertical aspects accordingly.

4.1 Definitions and theoretical background

The definitions of relevant terms and the corresponding theoretical background is of vital importance, in order to determine the scope and goals of evaluation. Any developer, before actually initiating his/her experiments, must have full grasp of what can and what cannot be evaluated. We, hereinafter, present relevant definitions and background theory with respect to: a) metrics, b) measurement methods, and c) aggregation.

Metrics

Metrics are standards that define measurable attributes of entities, their units and their scopes. Metrics are the essential building blocks of any evaluation process, since they allow the establishment of specific goals for improvement. A specific metric provides an indication of the degree to which a specific system attribute has met its defined goal. Deviation from the desired range of values indicates that improvement is needed in the related parts or modules of the system. With respect to a complete evaluation methodology, a metric is the answer to the question: "*What should I evaluate?*".

It must be noted that in software engineering (SE) the term metric is often used interchangeably with the term measurement to denote specific measured values for an attribute. In this work, however, we distinguish the two terms, in order to separate the metrics selection process from the actual measurement data collection.

A metric is defined by: a) its relationship to the corresponding features of the evaluated system, and b) the selected scale of measurement. The former consists of actual parameters or attributes of the system. For example, in auction environments, typical attributes would be the starting and ending time of an auction, as well as the winning bid. The latter term refers to the unit of measurement for this attribute. In the above example, a timestamp in milliseconds or an amount in Euros would suffice to describe the scale of the selected attributes. Typical types of scales include nominal and ordinal values, intervals and ratio scales.

In SE more than 300 metrics have been defined and used in various evaluation methodologies. Metrics are organized in categories, including technical, performance, business, productivity and end-user metrics (Fenton(1991)). However, in this work, we focus only on performance metrics, since DM for MAS is a relatively new technology that has not yet reached the maturity of software products. The proposed evaluation methodology, however, is expandable so that other aspects besides performance may be covered as well.

A requirement of our methodology with respect to metrics is to be able to provide a set of appropriate metrics and a comprehensive organization of this set in terms of similarity and cohesion. This taxonomy of metrics should also be extensible so that it is applicable to any application. A user will ideally be able to parse this taxonomy and select the metrics that are desirable for the system at hand.

Measurement

Measurement is defined as "the process of ascertaining the attributes, dimensions, extend, quantity, degree of capacity of some object of observation and representing these in the qualitative or quantitative terms of a data language" (Krippendorff (1986)). Having selected the appropriate metrics, measurement is the next fundamental methodological step that systematically assigns specific values to these metrics. Typical measurement methods consists of experimental design and data collection. A measurement method is the answer to the question "*How should I perform the experimental evaluation?*".

Among the three distinct steps of the proposed methodology, measurement is the most dependent on the implementation details of the application. Indeed, a generalized evaluation methodology cannot indicate precise steps of carrying out a specific measurement procedure. From this point of view, the key requirement of our methodology is to provide a set of measurement methods, so that the user may choose from a variety of options the most appropriate on to tackle the application domain at hand. Our methodology also provides a guideline list for conducting experimental design and data collection, as derived from traditional SE evaluation paradigms.

Aggregation

Aggregation, or composition, is the process of summarizing multiple measurements into a single measurement is such a manner that the output measurement will be characteristic of the system performance. Aggregation groups and combines the collected measurements, possibly by the use of weights of importance, in order to conclude to atomic characterization for the evaluated system. For example, an evaluated system may perform exceptionally well in terms of response time metrics (timeliness), but these responses may be far from correct (accuracy). An aggregation process must weightedly balance contradicting measures and provide an overall view of parts or the whole of the system, within boundaries of acceptable performance. Aggregation is the answer to the question: "*What is the outcome of the evaluation procedure?*".

A plethora of diverse aggregation techniques exist. Most commonly, aggregation is accomplished with the assistance of mathematical representation of multi-dimensional vectors and methods of either comparing sets of such vectors or comparing single vectors against predefined ideal vectors. The field of multi-criteria decision making provides us with a rich literature on this issue. Approaches in this field vary from simple sum, average, triangulation or other weighted functions to multi-dimensional vector comparison and vector distance. Especially in the field of intelligent system evaluation, vectors play a crucial role, in the form of Vector of Performance (VoP) or Vectors of Intelligence (VoI) (Szuba (2002)).

A requirement of our methodology with respect to aggregation is to provide with the user with appropriate aggregation theory and techniques in order to combine any specific measurements into conclusive characterizations of the evaluated system.

4.2 Theoretical Tools

We next present a set of theoretical tools that aim to assist users throughout the designing of the evaluation procedure, by providing sets of options and guidelines for intelligent performance assessment.

Metrics

Motivated by the requirements for metrics presented above, we introduce a theoretical tool for metric categorization in the form of an acyclic directed graph. The graph is organized in layers or *views* of granularity from general to specific, as further explained below. A user may traverse the graph in a top-down manner and, depending on the choices made, he/she shall conclude to a set of suitable metrics. After finalizing the measurement and aggregation methodology, as presented in the following sections, the developer will be able to traverse the graph upwards in order to provide single characterization for the system at each view. The graph is designed to be general, but also provides the option of extensibility for necessary domain specific metrics.

This tool was inspired by numerous related efforts in the traditional SE field, that provide comprehensive taxonomies of metrics (e.g. (de los Angeles Martin and Olsina (2003)). Besides dealing with predictable, deterministic, closed and non-dynamic software, the above approaches also come short because of the flat categorization of metrics at a single level of granularity.

In the proposed approach, we organize a metrics graph into four views, as depicted in Figure 3:

These views include:

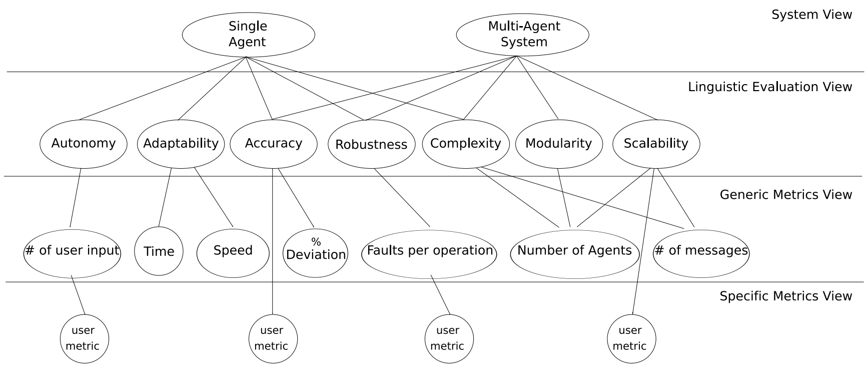


Fig. 3. Metrics graph

1. *System view*: At the top-most level, the class of the application is selected. A user may chose between single-agent, multi-agent society and multi-agent competition, depending on the scope and focus of the evaluation effort.
2. *Linguistic evaluation view*: At this level, a user chooses the appropriate verbal characterizations of system aspects, such as accuracy, timeliness, robustness and scalability. These abstract high level characterizations exclude parts of the underlying metrics, while focusing on the aspects of interest to the evaluator.
3. *Generic metrics view*: This level consists of metrics that are general and independent of the application field, such as response time, number of agents and message exchange frequency. The user may either use directly these metrics or refine them by continuing to the next level.
4. *Specific metrics view*: The final level consists of metrics that are specific to the application field. These metrics are only defined by the user, since they are not known *a priori* to a generalized evaluation methodology. Newly defined metrics must conform to the metric definition and parametrization presented in the previous section. Finally, they must be appended to one of the graph nodes of the above levels with directed arcs.

After selecting the metrics from this graph, the user is requested to define a set of parameters for each metric, including the preferred scale of measurement and other attributes, such as frequency of measurement, time intervals etc.

Measurement methods

Before implementing the actual measurement process, one must define the measurement method. Kitchenham (Kitchenham (1996)) provides a categorization of measurement techniques, with respect to the types of properties employed and the nature of the experimental technique. Inspired by this work, we provide the categorization displayed in the Table 2.

Table 2. Categorization of measurement methods

Experiment Type	Description
Quantitative experiment	An investigation of the quantitative impact of methods/tools organized as a formal experiment
Quantitative case study	An investigation of the quantitative impact of methods/tools organized as a case study
Quantitative survey	An investigation of the quantitative impact of methods/tools organized as a survey
Qualitative screening	A feature-based evaluation done by a single individual who not only determines the features to be assessed and their rating scale but also does the assessment. For initial screening, the evaluations are usually based on literature describing the software method/tools rather than actual use of the methods/tools
Qualitative experiment	A feature-based evaluation done by a group of potential user who are expected to try out the methods/tools on typical tasks before making their evaluations
Qualitative case study	A feature-based evaluation performed by someone who has used the method/tool on a real project
Qualitative survey	A feature-based evaluation done by people who have had experience of using the method/tool, or have studied the method/tool. The difference between a survey and an experiment is that participation in a survey is at the discretion of the subject
Qualitative effects analysis	A subjective assessment of the quantitative effect of methods and tools, based on expert opinion
Benchmarking	A process of running a number of standard tests using alternative tools/methods (usually tools) and assessing the relative performance of the tools against those tests

Having selected the measurement method, one must thoroughly provide an experimental design prototype and a data collection procedure. As stated earlier, our methodology can only provide a set of guidelines that any designer may adjust to their specific application. A typical experimental design procedure must describe thoroughly the objectives of the experiments and ensure that these objectives can be reached using the specified techniques.

The last step of the measurement methodology is to carry out the data collection process. Here, the basic guidelines for the designer to follow are to ensure that the data collection process is well defined and monitor the data collection and watch for deviations from the experiment design.

Aggregation

Following the collection of measurement values and the construction of metric-measurement pairs, the problem of aggregation arises. In the evaluation process, aggregation occurs naturally in order to summarize the experimental findings into a single characterization of the performance, either of single modules, or the system as a whole. In the case of the metrics graph of the proposed methodology, after having the measurements collected, the user must traverse the graph in a bottom-up manner. From the *specific metrics view* and the *general metrics view*, he/she must proceed upwards and, at each view, apply aggregation techniques to provide single characterizations for every parent node.

For example, assume that the simplified subtree, depicted in Figure 4, has been used for measuring the performance of a single agent. It must be noted that in this example, no user specified metrics have been included and therefore the *Specific Metrics View* layer is omitted.

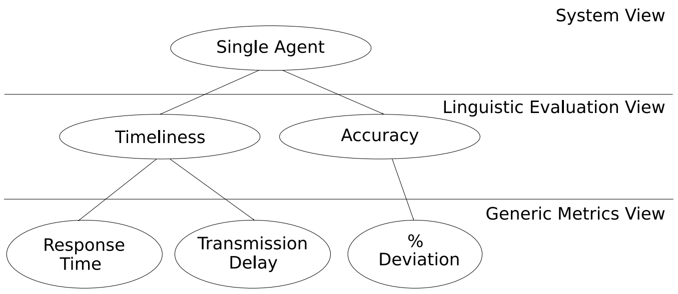


Fig. 4. Example metrics graph for aggregation

We assume that we already have obtained measurement values for the metrics at the general metrics view. In this example, the aggregation process consists of two steps. The first step concludes to a single measurement or characterization for each of the two selected linguistic terms, namely timeliness and accuracy, by weightedly combining the available measurements. The second step combines the linguistic characterization into a single characterization for the single agent system.

It becomes apparent from the above example, that a natural method for combining diverse and heterogeneous measurement information and linguistic characterizations is needed. We argue that *fuzzy aggregation* provides us with the appropriate natural functionality for this purpose. The term *natural* refers to the ability of the evaluator to express the evaluation findings in a manner that is coherent to their natural language. In other words, the fuzzy aggregation process translates the problem of combining numerical, ordinal or other measures into a collection of verbal characterizations for the system performance.

The proposed fuzzy aggregation method consists of four steps:

1. *Define weights in the metrics graph.* This process determines the importance of each node in the metrics graph with respect to the overall system performance. This decision relies heavily on the application domain as well as the requirements of each application. Hence, the determination of the weights may occur either a) semi-automatically, in case historical data on the importance of each node are available, possibly by an expert system, or b) directly by an expert user, the system designers in most cases.
2. *Define corresponding fuzzy scales for each metric.* The next step deals with the definition of fuzzy scales for the selected metrics. Fuzzy scales are defined by ordinal linguistic variables, such as *low*, *moderate*, *high* and membership functions that map numerical values to the above variables. Having the scales defined, one may already have scales for *natural* characterizations of performance, such as *high response time* or *moderate accuracy*, with respect to desired values.
3. *Convert actual measurements to fuzzy scales.* The conversion is a simple import of the selected measurements to the membership functions defined in the previous step.
4. *Apply a corresponding fuzzy aggregation operator at each view of the graph.* A wide variety of fuzzy aggregation operators exists (Grabisch et al. (1998)), which can be categorized in:
   - Conjunctive operators, that perform aggregation with the logical "and" connection.
   - Disjunctive operators, that perform aggregation with the logical "or" connection.
   - Compensative operators, which are comprised between minimum and maximum, such as mean or median operators.
   - Non-compensative operators, that do not belong to any of the above categories, such as symmetric sums.

Theoretical tools: Summary

In Table 3, we summarize the required methodological steps with respect to the theoretical tools, which take place at the evaluation process of a development methodology. In section 5.2, we present a real world case study on which the presented methodology is thoroughly applied.

Table 3. Summarization of methodological steps

1. Traverse metrics graph and select metrics
2. Provide domain specific metrics (optionally)
3. Determine metrics parameters
4. Specify measurement method and parameters
5. Execute experiments
6. Define weights in the graph
7. Define fuzzy scales and convert measurements accordingly
8. Select and apply aggregation operators on the collected measurements

4.3 Software Tools

In addition to the above presented theoretical tools, a complete off-the-shelf evaluation methodology must contain specifically implemented software tools that assist the evaluation procedure. In this section, we sketch the outline of such a software evaluation tool and the corresponding design guidelines that can be adopted by other similar tools.

The proposed evaluator tool is required to provide semi-automated assistance to the user throughout the following phases of evaluation:

- design
- run-time experimental procedure
- evaluation data summarization
- presentation of the results

A key requirement for this tool is that there must be minimum intervention to the code of an existing system and minimum prior knowledge of the system developer with respect to evaluation aspects. This requirement allows such software tools to apply to existing applications as an evaluation plug-in using dynamic code generation. Figure 5 depicts the six stages of assistance to the evaluation procedure.

1. *Selection of Metrics*: The first component of the proposed tool, is a Graphical User Interface that presents a metrics graph to the user and provides him/her with edit operations. According to the needs of the application, the user selects paths that lead to useful metric nodes and deletes paths and nodes that are unnecessary. As a result, the user determines a subset of the initial graph that is specific to the initiated evaluation procedure. Finally, the parameters of each selected metric as well as corresponding aggregation weights are defined.
2. *Dynamic Interface Generation*: Having the metrics subtree that has been produced in the previous stage, the tool dynamically generates a Java programming interface that corresponds to this evaluation procedure. Elements of agent behavior, communication and interaction are incorporated to implement essential abstract methods that are instantiated in the next stage.

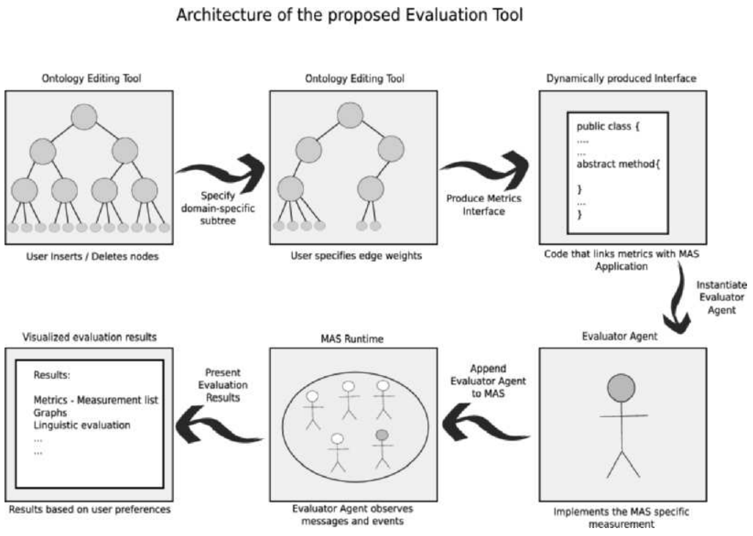

Architecture of the proposed Evaluation Tool
Ontology Editing Tool
User Inserts / Deletes nodes
Specify domain-specific subtree
Ontology Editing Tool
User specifies edge weights
Produce Metrics Interface
Dynamically produced Interface
public class {
....
...
abstract method{
}
...
}
Code that links metrics with MAS Application
Instantiate Evaluator Agent
Evaluator Agent
Implements the MAS specific measurement
Append Evaluator Agent to MAS
MAS Runtime
Evaluator Agent observes messages and events
Present Evaluation Results
Visualized evaluation results
Results:
Metrics - Measurement list
Graphs
Linguistic evaluation
...
...
Results based on user preferences

Fig. 5. Outline of the proposed Agent Evaluator Software Tool

3. *Implementation of Evaluator Agent*:Based on the produced programming interface, the user of the tool provide the necessary code for an Evaluator Agent that implements this interface, taking into account all the application-specific details.
4. *Attachment of Evaluator Agent to a MAS at run-time*: The Evaluator Agent is imported to the running system at run-time, executing observation operations on actions, events and messages that relate to the selected metrics.
5. *Data Collection*: The Evaluator Agent records all the observed actions into a log file.
6. *Presentation of Results*: After completion of the system execution or the ellapse of a predefined interval, the proposed tool presents the outcome of the evaluation process to the user, by processing the log files, deriving appropriate measurements from recorded events and messages and aggregates the results accordingly.

5 Agent Academy II: an Agent-Oriented Software Engineering tool for building DM-enriched MAS

5.1 Agent Academy II

Agent Academy is an open-source project, currently in the second version[1]. The design and implementation has changed radically since version 1.0. The initial goals of the application have not changed though, focusing mainly in the integration of agent developing and data mining technologies. We tried also to make this integration as broad as possible. In this perspective, we added features we find valuable to developers such as project management capabilities or java editing tools. Agent Academy enables users to reuse existing code by allowing a user to simply paste it into AA directories.

Agent Academy consists of three basic tools that form also an abstraction of creating a Multi Agent System. These tools are:

- Behavior Design Tool, where the user can create JADE behaviors, compile them and use a number of capabilities that this tool provides (for example users can create java source code that sends/receives ACL messages using a simple GUI etc) A central feature of the BDT is that it supplies to the users an intuitive tool for keeping the source code structured. The idea behind this feature is that users can create blocks of code of any size (i.e. it can span in an entire method). These blocks have a description, a body (the source code) and the method in which they belong. All these attributes can change dynamically allowing the users to add the same block of code in many methods or create a high-level description of the source file by summing up all the blocks descriptions. Implementing these blocks as static variables enables users to exchange blocks among multiple source files.
- Agent Design Tool, with which the developers can create software agents and add the designed behaviors as the agents execution tasks. We think agent functionality should reside in the agents behaviors, so ADT has not extended capabilities. ADT is equipped with a number of capabilities that we think are really important to agent developers. We are going to briefly mention them here and describe them in greater detail later. The first among them is the debugging tool (aka Graphical Monitor Agent Execution) that its use can save lot of debugging time. By means of this monitor, users can watch the execution of agent behaviors. This process is dynamic, meaning that behaviors that are added to the agent long after agent initialization can also be monitored. A tool that is also available from other Agent Academy modules is the Data Mining Module. Users can launch this tool, create a data mining model in the fly and add it to agent code just like any other JADE behavior.

[1] Available at http://sourceforge.net/projects/agentacademy

- Multi-Agent System Design Tool, is the tool that developers can use to create Multi-Agent Systems. The functionality of this tool is limited, constrained mainly in the easy, user-friendly definition of the participants in the MAS being designed.

Agent Academy users are encouraged to implement the MAS creation process following steps that correspond to the order the AA tools were presented, although other approaches can also be effective. Beyond this agent-oriented functionality, Agent Academy comes with a set of tools for a small/medium scale Project Management. In more detail, users are given the possibility to keep a small account of their projects components by using the Project Notepad that is matrix in which they can store the agents and behaviors created inside their project. Users can also use standard clean/build/run project capabilities from the Agent Academy main window.

5.2 A real world demonstrator

For validating the proposed methodology, we have selected Supply Chain Management (SCM) as a representative domain for testing agents that utilize DM techniques. We have implemented an SCM agent under the name Mertacor that has successfully participated in past Trading Agent SCM Competitions. Mertacor combines agent features with DM techniques. In the remainder of this section, we provide an overview of the SCM domain, the competition scenario and Mertacor's architecture. We conclude by applying the proposed evaluation methodology to different implementations of Mertacor.

Supply Chain Management

SCM tasks comprise the management of materials, information and finance in a network consisting of suppliers, manufacturers, distributors and customers. SCM strategies target at the efficient orchestration of the sequence of tasks, from raw materials to end-user service. Traditional SCM relied heavily on rigid and predefined contracts between participating parties. However, the need for dynamic configuration of the supply chain, as indicated nowaydays by global markets, became imperative. Modern SCM approaches focus on the integration, optimization and management of the entire process of material sourcing, production, inventory management and distribution to customers.

The core design primitives for coping with SCM are a) coordination of distributed, heterogeneous information,b) efficient negotiation between participating entities and c) functional resource allocation. MAS are an ideal modelling and implementation solution to this inherently distributed problem (Ferber(1999)) (Wu et al. (2000)). Moreover, DM techniques have been successfully applied for SCM purposes in the past. DM has efficiently addresses issues of customer and supplier profiling, inventory scheduling and market based analysis.

SCM Trading Agent Competition Game

The Trading Agent Competition (TAC) is an annual, international competition that consists of two games: a) TAC Classic and b) TAC SCM. In the latter, the game scenario, as described in (Collins and Janson (2004)), consists of groups of six competing agents, each of which represents a PC assembler with limited production capacity an competes with other agents in selling PC units to customers. The agents' task is to efficiently manage a part of the supply chain, namely to negotiate on supply contracts, bid for customer offers, manage daily assembly activities and ship completed offers to customers. Negotiations with manufacturers and customers are performed through a Request-For-Quote (RFQ) mechanism, which proceeds in three steps:

- Buyer issues RFQs to one or more sellers
- Sellers respond to RFQs with offers
- Buyers accept or reject offers. An accepted offer becomes an order.

In order to get paid, the agent must deliver on-time, otherwise it is charged with a penalty. At the end of the game, the agent with the greatest revenue is declared winner. For more information on the game, read the game specification provided by (Collins and Janson (2004)).

Mertacor Architecture

Mertacor, as introduced in (Kontogounis et al. (2006)), consists of four cooperating modules (Figure 6):

1. the *Inventory Module*(IM). Mertacor introduces an assemble-to-order (ATO) strategy, which is a combination of two popular inventory strategies, namely *make-to-order* and *make-to-stock*.
2. the *Procuring Module*(PM). This module predicts future demand and orders affordable components, balancing between cheap procurement and running needs in the assembly line.
3. the *Factory Module*(FM). This module constructs assembly schedules and provides the Bidding Module with information on the factory production capacity, based on simulation of customer demand for the next 15 game days.
4. the *Bidding Module*(BM). This module attempts to predict a winning bid for each order, by performing DM on logs of past games, and makes respective offers for incoming orders.

Mertacor's core integrates these modules into a transparently robust unit that handles negotiations with both customers and suppliers. This architecture provides flexibility and extensibility, permitting the application of Mertacor's strategy to other real-life SCM environments.

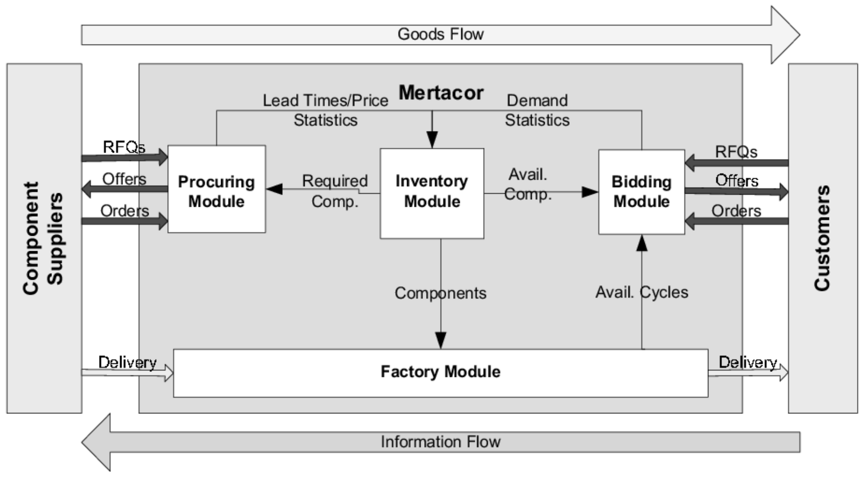


Fig. 6. Overview of Mertacor's architecture

Evaluating Mertacor's performance

In the remainder of this section, we apply the proposed evaluation methodology to various implementations of Mertacor. In our effort to assess the impact of DM in Mertacor's performance, we require that the experiments are planned is such way that deals with both DM algorithmic-specific efficacy and their impact on overall agent performance. We follow the methodological steps defined in Table 3.

Step 1: Traverse metrics graph and select metrics

Starting from the *System view*, we select the *Single Agent* node and its corresponding path. This choice is attributed to the nature of auctioning environments; we, being the developers of Mertacor, have complete control only on the agent's executing thread and observe the auctioning world only through Mertacor's perspective. We, therefore, need to focus on performance aspects that exclusively deal with this single agent.

At the *Linguistic Evaluation View*, we select the linguistic metrics of *Accuracy*, *Timeliness* and *Adaptability*. Indeed, from our experience in SCM auctions, these three characteristics are the most significant ones, since the outcome of each auction is heavily dependent on the deviation of the forecasted bid, the on-time delivery of the bid and the ability of the agent to adapt in dynamic environments, respectively.

At the *Generic Metrics View* we only select *Time*, as the standard metric for *Timeliness*. The rest of the metrics are domain specific and are, therefore, defined in the next methodological step.

Step 2: Provide domain specific metrics

Metrics for *Accuracy* should directly refer to DM related performance, since the outcome of the application of DM is directly related to the selected bid. For this purpose, we have selected the *Correlation Coefficient* (cc), the *Relative Absolute Error* (RAE) and the *Root Mean Square Error* (RMSE) metrics.

Finally, for *Adaptability*, we have selected *Competition Intensity* that defines the participation intensity in the auction environment. In highly competitive environments, our agent is required to exhibit a larger degree of adaptability.

An instance of the metrics graph for this evaluation effort is depicted in Figure 7.

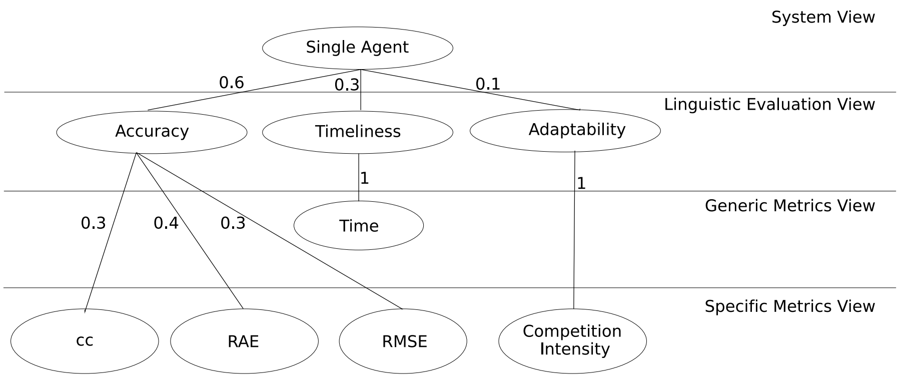


Fig. 7. Resulted metrics graph for Mertacor evaluation

Step 3: Determine metrics parameters

We now continue by defining the scale of each metric. For the three linguistic metrics, *Accuracy*, *Timeliness* and *Adaptability*, we define the corresponding fuzzy scales in *Step 7* of the methodology. For the generic and specific metrics, we provide the following scales:

1. *CC*: The correlation coefficient is the degree at which the forecasted bid and the resulted price are correlated. The *cc* lies in the [-1,1] interval.
2. *RAE*: The *Relative Absolute Error* is a percentage indicator for the deviation of the above mentioned variables.
3. *RMSE*: The *Root Mean Square Error* is another well-known DM metric for the above mentioned variables.
4. *Time*: In TAC SCM auctions, bids are normally submitted just before the end of each predefined auction interval. One could argue that, since this time constraint exists, all agents have a time barrier to bid and therefore

all bidding calculation procedures should be characterized either as successful or failed. In that context, timeliness is only a binary metric that provides no further performance indication. However, due to the modular architecture of Mertacor, the earliest possible decision on the bid, allows the agent to perform other game-related tasks in this interval. We therefore define *Time* as the time interval between the first call of the related bidding API function and the determination of the bidding value, in milliseconds.

5. *Competition Intensity*: We have selected two different competing environments that affect *Adaptability*: a) Finals, where the competition intensity is high, and b) Second finals, where the competition intensity is low.

Step 4: Specify measurement method and parameters

Estimation of the winning price of the bids can be modeled as a regression problem, where the desired output is the agent's bidding price for clients' RFQs and the inputs are the parameters related to the bid that are known to the agent. The initial set of attributes considered are the demand (Total PCs requested each day), the demand in the product's market range, the due date of the order, the reserve price of components, and the maximum and minimum prices of same type PCs sold in the last days (2 previous days for maximum 4 for minimum), as shown in Table 4.

Table 4. Set of SCM auction attributes for DM

Attribute description	Attribute name
Demand (Total PCs requested the day the RFQ was issued)	demandAll
Demand in the product's market range	demandRange
Duedate	dueDate
Reserve price	reservePrice
Maximum price of PCs of same type sold in the last 1 day	max1
Maximum price of PCs of same type sold in the last 2 days	max2
Minimum price of PCs of same type sold in the last 1 day	min1
Minimum price of PCs of same type sold in the last 2 days	min2
Minimum price of PCs of same type sold in the last 3 days	min3
Minimum price of PCs of same type sold in the last 4 days	min4
Winning price of the bid	price

Available data was split into three subsets, each one representing a different market range (LOW - MEDIUM - HIGH), both for the finals and second finals of the game, resulting to six different datasets (finalsLOWMEDIUMHIGH and secondFinalsLOWMEDIUMHIGH).

The instances within the initial datasets ranged from 45000 to 230000 instances. Analysis was performed on all datasets. Nevertheless, due to space limitation we shall discuss only one case (finalsLOW dataset), while analysis on the other cases was performed in an analogous manner. The initial dataset contained 156228 records of bids. In order to remove redundant information and enable quicker and more accurate training, a number of pre-processing filters were tested against the dataset. In particular, we applied the CfsSubsetEval1 (Hall (1998)) WrapperSubsetEval2 (Kohavi and John (1997)), and ReliefFAttributeEva3l (Sikonja and Kononenko (1997)) filters for attribute selection, using the GreedyStepwise and RandomSearch search methods. The trimmed dataset contained the following attributes as input: the demand, the reserve price for components, the maximum price of same type PCs for the two previous days and the minimum price for the previous day, while price was the output attribute. In order to reduce the number of instances for training, and since the class attribute (price) is numeric, the StratifiedRemoveFolds4 (Breiman and Stone (1984)) method was selected. Two datasets were finally produced, containing the one third (1/3) and one eighth (1/8) of the initial instances respectively.

Finally, for training purposes, four different classification (regression) and two meta-classification schemes were applied, in order to decide on the one that optimally meets the problem of predicting the winning bid of an order:

1. Linear Regression
2. Neural Networks
3. SMOreg (Support Vector Machines)
4. the M5' algorithm
5. Additive Regression
6. Bagging

Step 5: Execute experiments

In order to experiment on the data with a variety of training techniques and algorithms, the WEKA (Witten and Frank (2005)) was selected, providing with a wide range of filters for pre-processing, model evaluation, visualization and post-processing. The results of the experimental procedures are presented in Table 5 and Table 6, for low and high *Competition Intensity* values, respectively.

Step 6: Define weights in the graph

This step requires a subjective, expert-initiated attribution of weights to the corresponding edges of the metrics graph. Driven by our experience in the field, we assign a higher weight to *Accuracy* (0.6) and lesser weights to *Timeliness* (0.3) and *Adaptability* (0.1). The corresponding weights are illustrated in Figure 7.

Step 7: Define fuzzy scales and convert measurements accordingly

We provide the following fuzzy sets for the selected metrics:

Table 5. Results of experiments for low Competition Intensity

Algorithm	CC	RAE (%)	RMSE	Time	Data Subset
Linear Regression	0.93	28.99	90.17	108	LOW
Neural Networks	0.93	32.91	94.69	111	LOW
Support Vector Machines	0.93	26.47	89.08	157	LOW
M5'	0.95	22.77	61.09	140	LOW
Additive Regr.	1.00	3.21	22.12	192	LOW
Bagging	0.98	14.89	52.02	201	LOW
Linear Regression	0.94	26.50	112.71	129	MEDIUM
Neural Networks	0.95	24.85	105.69	133	MEDIUM
Support Vector Machines	0.93	28.66	109.61	182	MEDIUM
M5'	0.97	19.30	86.90	168	MEDIUM
Additive Regr.	1.00	3.01	23.53	230	MEDIUM
Bagging	0.98	13.26	65.52	237	MEDIUM
Linear Regression	0.94	26.78	105.51	140	HIGH
Neural Networks	0.95	27.83	105.21	144	HIGH
Support Vector Machines	0.94	25.82	103.32	204	HIGH
M5'	0.96	21.29	87.14	183	HIGH
Additive Regr	1.00	2.98	24.70	249	HIGH
Bagging	0.98	15.13	65.69	260	HIGH

Table 6. Results of experiments for high Competition Intensity

Algorithm	CC	RAE (%)	RMSE	Time	Data Subset
Linear Regression	0.98	14.34	63.40	110	LOW
Neural Networks	0.97	21.26	64.82	112	LOW
Support Vector Machines	0.96	17.48	72.84	155	LOW
M5'	0.98	13.49	56.79	145	LOW
Additive Regr.	0.97	19.24	67.51	189	LOW
Bagging	0.99	5.62	27.76	199	LOW
Linear Regression	0.97	16.95	74.33	133	MEDIUM
Neural Networks	0.97	20.21	75.20	132	MEDIUM
Support Vector Machines	0.97	17.54	73.81	193	MEDIUM
M5'	0.99	9.91	46.93	172	MEDIUM
Additive Regr.	0.96	25.98	92.30	227	MEDIUM
Bagging	1.00	4.84	31.38	233	MEDIUM
Linear Regression	0.97	16.55	68.14	142	HIGH
Neural Networks	0.98	18.94	71.91	144	HIGH
Support Vector Machines	0.97	16.27	72.31	208	HIGH
M5'	0.99	10.03	45.35	178	HIGH
Additive Regr.	0.95	28.26	94.68	242	HIGH
Bagging	0.99	5.70	34.90	263	HIGH

- Fuzzy variables *very low*,*low*,*medium*,*high* and *very high* for the *RAE* and *RMSE* metrics
- Fuzzy variables *low* and *high* for the *CC* and *Competition Intensity* metrics
- Fuzzy variables *low*, *medium* and *high* for the *Time* metric

The corresponding fuzzy membership functions for *CC*, *RAE*, *RMSE* and *Time* are depicted in Figure 8.

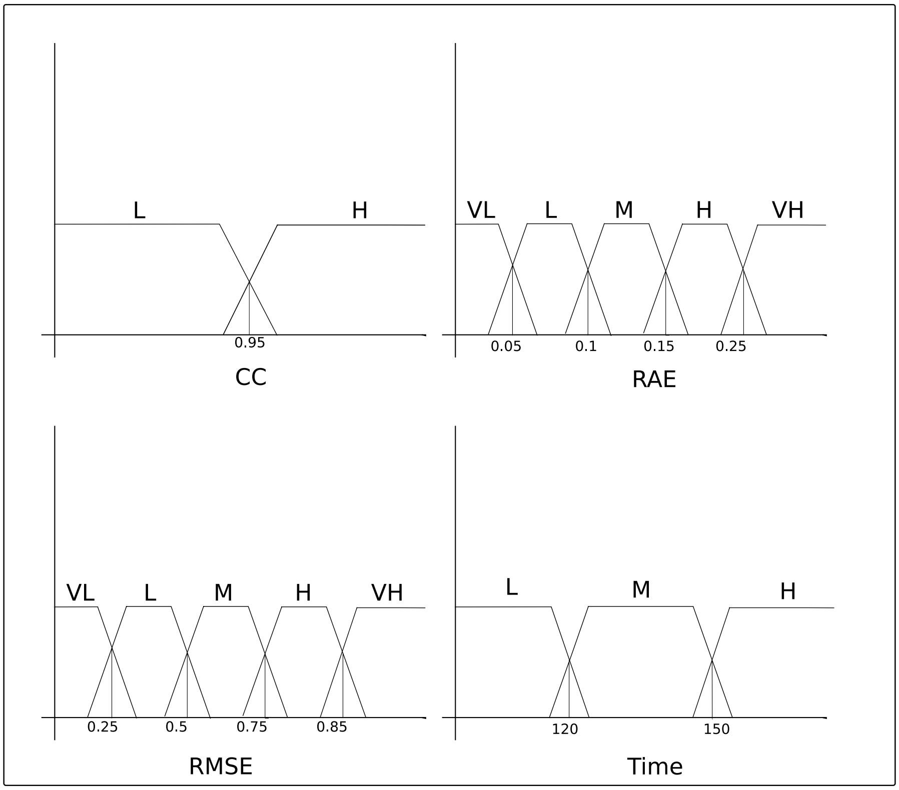


Fig. 8. Fuzzy membership functions for the selected metrics

Step 8: Select and apply aggregation operators on the collected measurements

The final step of the methodology consists of the application of the selected aggregation method. As described in (Grabisch et al. (1998)), the application of weighted operators result into a single characterization for every linguistic metric. After summarizing the results it can be seen that *Additive Regression* exhibit the best performance for all data subsets, as it balances between large accuracy and adequate time responses, for both high and low *Competition Intensity.*

6 Concluding remarks

As the number of application that integrate DM and AT increase, the need for assessing the overall system performance is imperative. In this work, we have presented a generalized methodology for evaluating agents and MAS that employ DM techniques for knowledge extraction and knowledge model generation. The proposed methodology comprises a set of concise steps that guide an evaluator through the evaluation process. A novel theoretical representation tool introduces a metrics graph and appropriate selection guidelines for measurement and aggregation methods. A real world DM-enriched agent in the field of Supply Chain Management has used to demonstrate the applicability of the proposed methodology. Future work in this direction include the specification of a unique metrics ontology for the proposed metrics representation graph and the expansion of the graph with a complete set of real world metrics, borrowed either from the software engineering discipline or existing, ad-hoc efforts in intelligent systems evaluation. Finally, the proposed methodology must be thoroughly tested in a number of diverse and representative case studies.

References

James Albus, Elena R. Messina, and John M. Evans. (2000), Performance metrics for intelliget systems (permis) white paper. In *Proc. of the First International Workshop on Performance Metrics for Intelligent Systems (PERMIS).*

Dennis Allard and Chris Fraley. (1997), Non parametric maximum likelihood estimation of features in saptial point process using voronoi tessellation. *Journal of the American Statistical Association.*

Brian W. Arthur. (1994), Inductive reasoning and bounded rationality. *American Economic Review*, 84:406–411.

Ioannis N. Athanasiadis and Pericles A. Mitkas. (2005),Social influence and water conservation: An agent-based approach. *Computing in Science and Engg.*, 7 (1):65–70, 2005. ISSN 1521-9615.

Friedman J.H. unsrt Olshen R.A. Breiman, L. and C.J. Stone. (1984), *Classification and Regression Trees.* Chapman and Hall, New York.

J. L. Burke, R.R. Murphy, D. R. Riddle, and T. Fincannon. (2002), Task performance metrics in human-robot interaction: Taking a systems approach. In *Proc. of the Fourth International Workshop on Performance Metrics for Intelligent Systems (PERMIS).*

Arunachalam R. Sadeh N. Ericsson J. Finne N. Collins, J. and S. Janson. (2004), The supply chain management game for the 2005 trading agent competition. Technical report, CMU.

M. de los Angeles Martin and Luis Olsina. (2003), Towards an ontology for software metrics and indicators as the foundation for a cataloging web system. In *LA-WEB '03: Proceedings of the First Conference on Latin American Web Congress*, page 103, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-2058-8.

Christos Dimou, Alexandros Batzios, Andreas L. Symeonidis, and Pericles A. Mitkas. (2006), A multi-agent simulation framework for spiders traversing the semantic web. In *Web Intelligence*, pages 736–739. IEEE Computer Society, 2006. ISBN 0-7695-2747-7.

Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. (1996), Knowledge discovery and data mining: Towards a unifying framework. In *KDD*, pages 82–88.

Norman E. Fenton. (1991), *Software Metrics: A Rigorous Approach.* Chapman & Hall, Ltd., London, UK, ISBN 0442313551.

Jacques Ferber. (1999), *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, ISBN 0201360489.

A. A. A. Fernandes. (2000), Combining inductive and deductive inference in knowledge management tasks. In *DEXA '00: Proceedings of the 11th International Workshop on Database and Expert Systems Applications*, page 1109, Washington, DC, USA, IEEE Computer Society. ISBN 0-7695-0680-1.

Ian T. Foster, Carl Kesselman, and Steven Tuecke. (2001), The anatomy of the grid - enabling scalable virtual organizations. *CoRR*, cs.AR/0103025.

Ian T. Foster, Nicholas R. Jennings, and Carl Kesselman. (2004), Brain meets brawn: Why grid and agents need each other. In *AAMAS*, pages 8–15.

Boris Galitsky and Rajesh Pampapathi (2003), Deductive and inductive reasoning for processing the claims of unsatisfied customers. In Paul W. H. Chung, Chris J. Hinde, and Moonis Ali, editors, *IEA/AIE*, volume 2718 of *Lecture Notes in Computer Science*, pages 21–30. Springer, ISBN 3-540-40455-4.

Nicholas Gibbins, Stephen Harris, and Nigel Shadbolt (2003), Agent-based semantic web services. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 710–717, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-680-3.

M.A. Goodrich, E.R. Boer, J. W. Crandall, R.W. Ricks, and M.L. Quigley (2002), Behavioral entropy in human-robot interaction. In *Proc. of the Fourth International Workshop on Performance Metrics for Intelligent Systems (PERMIS).*

Michel Grabisch, Sergei A. Orlovski, and Ronald R. Yager (1998), Fuzzy aggregation of numerical preferences. pages 31–68.

Amy R. Greenwald and Peter Stone. (2001), Autonomous bidding agents in the trading agent competition. *IEEE Internet Computing*, 5(2):.

M. A Hall. (1998), Correlation-based feature subset selection for machine learning. Technical report, Thesis submitted in partial fulfilment of the requirements of the degree of Doctor of Philosophy at the University of Waikato.

X. Hu and B. Zeigler. (2002), Measuring cooperative robotic systems using simulation-based virtual environment. In *Proc. of the Fourth International Workshop on Performance Metrics for Intelligent Systems (PERMIS).*

Markus C. Huebscher and Julie A. McCann. (2004), Evaluation issues in autonomic computing. In *Proceedings of Grid and Cooperative Computing Workshops (GCC)*, pages 597–608.

Bernard J. Jansen, Tracy Mullen, Amanda Spink, and Jan Pedersen (2006), Automated gathering of web information: An in-depth examination of agents interacting with search engines. *ACM Trans. Inter. Tech.*, 6(4):442–464, 2006. ISSN 1533-5399.

Nicholas R. Jennings. (2001), An agent-based approach for building complex software systems. *Commun. ACM*, 44(4):35–41, 2001. ISSN 0001-0782.

Nicholas R. Jennings. (1999) Agent-oriented software engineering. In Ibrahim F. Imam, Yves Kodratoff, Ayman El-Dessouki, and Moonis Ali, editors, *IEA/AIE*, volume 1611 of *Lecture Notes in Computer Science*, pages 4–10, ISBN 3-540-66076-3.

Bob Kero, Lucian Russell, Shalom Tsur, and Wei-Min Shen. (1995) An overview of database mining techniques. In *KDOOD/TDOOD*, pages 1–8.

Barbara Ann Kitchenham. Evaluating software engineering methods and tool, part 2: selecting an appropriate evaluation method technical criteria. *SIGSOFT Softw. Eng. Notes*, 21(2):11–15, 1996. ISSN 0163-5948.

Y. Kodratoff. (1988), *Introduction to machine learning.* Pitman Publishing.

R. Kohavi and G. John. (1997), Wrappers for feature subset selection. *Artificial Intelligence journal, special issue on relevance*, 97(1-2):273–324.

I. Kontogounis, Chatzidimitriou, A. K., Symeonidis, and P.A. Mitkas. (2006), A robust agent design for dynamic scm environments. In *LNAI, Vol. 3955*, pages 127–136. Springer-Verlag.

Klaus Krippendorff, (1986), *A Dictionary of Cybernetics.* The American Society of Cybernetics, Norfolk, VA, USA.

Ingo Muller, Ryszard Kowalczyk, and Peter Braun (2006), Towards agent-based coalition formation for service composition. In *IAT '06: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2006 Main Conference Proceedings) (IAT'06)*, pages 73–80, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2748-5.

A. Negri, A. Poggi, M. Tomaiuolo, and P. Turci (2006), Agents for e-business applications. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 907–914, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-303-4.

A. L. Nelson, E. Grant, and T. C. Henderson. (2002), Competitive relative performance evaluation of neural controllers for competitive game playing with teams of real mobile robots. In *Proc. of the Third International Workshop on Performance Metrics for Intelligent Systems (PERMIS).*

Hyacinth S. Nwana. (1996), Software agents: An overview. *Knowledge Engineering Review*, 11(3):1–40.

Anna Perini, Paolo Bresciani, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. (2001), Towards an agent oriented approach to software engineering. In Andrea Omicini and Mirko Viroli, editors, *WOA*, pages 74–79. Pitagora Editrice Bologna, 2001. ISBN 88-371-1272-6.

Martin K. Purvis, Stephen Cranefield, Roy Ward, Mariusz Nowostawski, Daniel Carter, and Geoff Bush. (2003), A multi-agent system for the integration of distributed environmental information. *Environmental Modelling and Software*, 18(6):565–572.

J. Scholtz, B. Antonishek, and J. Young. (2002) Evaluation of human-robot interaction in the nist reference search and rescue test arenas. In *Proc. of the Fourth International Workshop on Performance Metrics for Intelligent Systems (PERMIS).*

T. K. Shih. (2000) Evolution of mobile agents. In *Proc. of the First International Workshop on Performance Metrics for Intelligent Systems (PERMIS).*

M.R. Sikonja and I. Kononenko. (1997) An adaptation of relief for attribute estimation on regression. machine learning. In *Proceedings of 14th International Conference on Machine Learning D., Fished (ed.)*, Nashville, TN, USA.

Andreas L. Symeonidis and Pericles A. Mitkas. (2005), *Agent Intelligence Through Data Mining.* Springer Science and Business Media.

Tadeusz Szuba. (2002), Universal formal model of collective intelligence and its iq measure. In *CEEMAS '01: Revised Papers from the Second International Workshop of Central and Eastern Europe on Multi-Agent Systems*, pages 303–312, London, UK, 2002. Springer-Verlag. ISBN 3-540-43370-8.

Jia Tang and Minjie Zhang. (2006), An agent-based peer-to-peer grid computing architecture: convergence of grid and peer-to-peer computing. In *ACSW Frontiers '06: Proceedings of the 2006 Australasian workshops on Grid computing and e-research*, pages 33–39, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc. ISBN 1-920-68236-8.

Gerhard Weiss. (2001) Agent orientation in software engineering. *Knowl. Eng. Rev.*, 16(4):349–373, 2001. ISSN 0269-8889.

Michael P. Wellman and Peter R. Wurman. (1998), Market-aware agents for a multiagent world. *Robotics and Autonomous Systems*, 24(3-4):115–125.

Ian H. Witten and Eibe Frank. (2005), *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, San Francisco.

Michael Wooldridge. (1999), Intelligent agents. In G. Weiss, editor, *Multiagent Systems.* The MIT Press.

J. Wu, M. Ulieru, M. Cobzaru, and D. Norrie. (2000), Supply chain management systems: state of the art and vision. In *9th International Conference on Management of Innovation and Technology*, pages 759–764. IEEE Press, 2000. ISBN 3-540-43370-8.

Peter R. Wurman, Michael P. Wellman, and William E. Walsh. (1998), The michigan internet auctionbot: a configurable auction server for human and software agents. In *AGENTS '98: Proceedings of the second international conference on Autonomous agents*, pages 301–308, New York, NY, USA, 1998. ACM Press. ISBN 0-89791-983-1.

Lotfi A. Zadeh. (2002), In quest of performance metrics for intelligent systemsa challenge that cannot be met with existing methods. In *Proc. of the Third International Workshop on Performance Metrics for Intelligent Systems (PERMIS).*

N. Zimmerman, C. Schlenoff, S. Balakirsky, and R. Wray. (2002), Performance evaluation of tools and techniques for representing cost-based decision criteria for on-road autonomous navigation. In *Proc. of the Third International Workshop on Performance Metrics for Intelligent Systems (PERMIS).*

Approximate Frequent Itemset Mining In the Presence of Random Noise

Hong Cheng[1], Philip S. Yu[2] and Jiawei Han[1]

[1] University of Illinois at Urbana-Champaign {hcheng3,hanj}@cs.uiuc.edu
[2] IBM T. J. Watson Research Center psyu@us.ibm.com

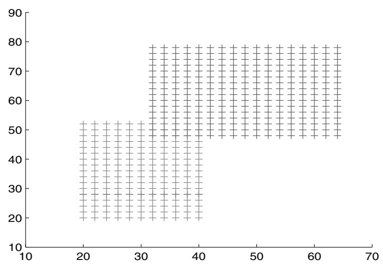
Summary. Frequent itemset mining has been a focused theme in data mining research and an important first step in the analysis of data arising in a broad range of applications. The traditional exact model for frequent itemset requires that every item occur in each supporting transaction. However, real application data is usually subject to random noise or measurement error, which poses new challenges for the efficient discovery of frequent itemset from the noisy data.

Mining approximate frequent itemset in the presence of noise involves two key issues: the definition of a noise-tolerant mining model and the design of an efficient mining algorithm. In this chapter, we will give an overview of the approximate itemset mining algorithms in the presence of random noise and examine several noise-tolerant mining approaches.

Key words: error-tolerant itemset, approximate frequent itemset, core pattern recovery

1 Introduction

Frequent itemset mining has been a focused theme in data mining research with a large number of scalable mining methods proposed (Agrawal *et al.*, 1993 , Agrawal and Srikant, 1994, Han *et al.*, 2000, Zaki *et al.*, 2000) and various extensions including closed itemsets, maximal itemsets and so on (Pei *et al.*, 2000, Zaki and Hsiao, 2002, Bayardo, 1998, Burdick *et al.*, 2001). Frequent patterns have found broad applications in areas like association rule mining (Agrawal *et al.*, 1993), indexing (Yan *et al.*, 2004), classification (Liu *et al.*, 1998, Li *et al.*, 2001, Cong *et al.*, 2005, Cheng *et al.*, 2007) and clustering (Wang *et al.*, 1999). In these applications, the ultimate goal is to discover interesting associations between objects and attribute subsets, rather than association among attributes alone. One important experimental application of frequent itemset mining is the exploration of gene expression data, where the


(a) Embedded True Patterns

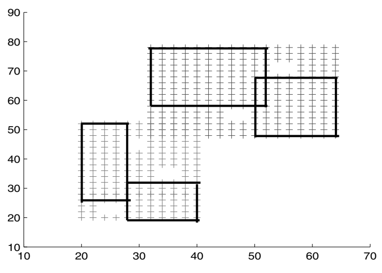

(b) Observed Distorted Patterns

Fig. 1. Patterns with and without Random Noise

joint discovery of both the set of conditions that significantly affect gene regulation and the set of co-regulated genes is of great interest. Another important application of frequent itemset mining is frequent pattern-based classification, where the associations between attributes and their relation to the class labels or functions are explored.

Despite the exciting progress in the field of frequent itemset mining and its extensions, an intrinsic problem with the exact frequent itemset mining is the rigid definition of support. An itemset x is supported by a transaction t, if each item of x exactly appears in t. An itemset x is frequent if the number of transactions supporting it is no less than a user-specified minimum support threshold (denoted as *min_sup*). However, in real applications, a database is typically subject to random noise or measurement error, which poses new challenges for the discovery of frequent itemsets. For example, in a customer transaction database, random noise could be caused by an out-of-stock item, promotions or some special event like the world cup, holidays, *etc.*. Measurement error could be caused by noise from experiments, uncertainty involved in discretizing continuous values, stochastic nature of the study field, *etc.*. In privacy-preserving data mining (Agrawal and Srikant, 2000, Verykios *et al.*, 2004), random noise is added to perturb the true values of the original database. Such random noise can distort the true underlying patterns. Theoretical analysis by (Liu *et al.*, 2006) shows that in the presence of even low levels of noise, large frequent itemsets are broken into fragments of logarithmic size, thus the itemsets cannot be recovered by the exact frequent itemset mining algorithms. Figure 1 shows two transaction databases. The x-axis represents items and the y-axis represents the transactions. Figure 1 (a) shows the embedded true patterns in the database without random noise while (b) shows the observed distorted patterns in the presence of random noise (several "holes" appear in the embedded patterns due to random noise). If the exact frequent itemset mining algorithms are applied to a database subject to random noise, as in Figure 1 (b), the original embedded true patterns will be fragmented into several smaller ones which are highlighted by the bounding boxes.

In this chapter, we will give an overview of approximate itemset mining algorithms in the presence of random noise and examine three different approaches: (1) a heuristic error-tolerant itemset (ETI) approach (Yang *et al.*, 2001), (2) an apriori-based approximate frequent itemset (AFI) approach (Liu *et al.*, 2006) and (3) a core pattern recovery (AC-Close) approach (Cheng *et al.*, 2006) to solve this problem.

2 Preliminary Concepts

Let a transaction database D take the form of an $n \times m$ binary matrix. Let $I = \{i_1, i_2, \ldots, i_m\}$ be the set of all items and $T = \{t_1, t_2, \ldots, t_n\}$ be the set of all transactions in D. A subset of I is called an *itemset*. Each row of D is a transaction $t \in T$ and each column is an item $i \in I$. A transaction t supports an itemset x, if for each item $i \in x$, the corresponding entry $D(t, i) = 1$. An itemset x is *frequent* if the fraction of transactions supporting it is no less than a user-specified threshold *min_sup*.

An intuitive approach for handling errors is to relax the requirement that a sub-matrix determined by the frequent itemset consists entirely of 1s, and allow it instead to contain a large fraction of 1s and a small fraction of 0s. Based on the same philosophy, different studies proposed different mining models and constraints on top of which, efficient mining algorithms and search/pruning strategies have been developed. Since each study has some variations of the mining model, the specific definitions and concepts are introduced with each approach in the following sections respectively.

Table 1 summarizes and compares the characteristics of the three different mining approaches in four aspects. Although the model definition and algorithmic properties will be provided in the following sections in detail, we give a brief explanation for Table 1. The first criterion *error control* describes how each noise-tolerant model defines the fraction of errors allowed. The ETI model controls the fraction of errors in each row or in the sub-matrix formed by the itemset; while the AFI model and the AC-Close model control the fraction of errors in both rows and columns. In addition, AC-Close enforces a core pattern constraint which will be explained later. For *mining methodology*, both ETI and AFI work in a bottom-up way *i.e.*, generating itemsets in a size-increasing order; while AC-Close works in a top-down fashion, *i.e.*, generating itemsets in a size-decreasing order. All three methods search for the itemsets in a breadth-first manner. As for the *result quality* criterion, it is assessed through simulation experiments: given a noise-free dataset D_t and known embedded patterns F_{true}, random noise is added to D_t to derive a noisy dataset D. Then noise-tolerant mining algorithms are applied on D to recover the true patterns. By knowing the true patterns, the quality of mining results could be evaluated. Two criteria are used for the evaluation purpose: *recall* (or *recoverability*) and *precision* (or *spuriousness*). According to the performance study in (Liu *et al.*, 2006, Cheng *et al.*, 2006), AFI outperforms ETI on both criteria of recall and precision. AC-Close achieves a similar or slightly lower recall than AFI while having a much higher precision than AFI. As for mining *efficiency* and *scalability*, AFI is shown to outperform ETI by an order of magnitude; while AC-Close is shown to outperform AFI by at least two orders of magnitude.

Table 1. Algorithm Characteristics Comparison

Criterion	ETI	AFI	AC-Close
Error Control	row or sub-matrix	row and column	row and column core pattern constraint
Mining Methodology	bottom-up breadth-first	bottom-up breadth-first	top-down breadth-first
Result Quality	low	moderate	high
Efficiency/Scalability	low	moderate	high

3 A Heuristic Mining Approach

(Yang *et al.*, 2001) was a pioneer study which generalizes the frequent itemsets allowing for the notion of errors in the itemset definition. It proposed an error-tolerant model (ETI) and developed a heuristic mining algorithm that identifies the error-tolerant frequent clusters of items in transactional databases. The proposed ETI algorithm was also applied to three application scenarios: (1) clustering high dimensional data; (2) query selectivity estimation; and (3) collaborative filtering.

3.1 An Error-Tolerant Itemset Model

In the binary matrix representation of the data, an error-tolerant frequent itemset is represented as a set of dimensions (called *defining dimensions* or DD in short) where 1 appears with high probability among a set of rows. Two constraints are enforced: *min_sup* and an error threshold ϵ. With different degrees of relaxation, two slightly different error-tolerant itemset models are proposed: strong ETI and weak ETI.

Definition 1. *Given min_sup* $= s$ *and error threshold* ϵ*, a strong ETI consists of a set of items, called defining dimensions* $DD \subseteq I$*, such that there exists a subset of transactions* $R \subseteq T$ *consisting of at least* $s|T|$ *transactions* *and, for each $t \in R$, the fraction of items in DD which are present in t is at least* $1 - \epsilon$.

Definition 2. *Given min_sup = s and error threshold ϵ, a weak ETI consists of a set of items, called defining dimensions $DD \subseteq I$, such that there exists a subset of transactions $R \subseteq T$, $|R| \geq s|T|$, and*

$$\frac{\sum_{t \in R} \sum_{d \in DD} D(t,d)}{|R| \cdot |DD|} \geq (1 - \epsilon) \qquad (1)$$

The strong ETI controls the fraction of errors in each supporting transaction to be no greater than ϵ; while the weak ETI only requires the sub-matrix formed by R and DD contains a fraction of errors to be no greater than ϵ. It is clear that anything that satisfies the strong definition also satisfies the weak definition, but not vice versa.

3.2 ETI Mining Algorithm

Based on the ETI models and their properties, (Yang *et al.*, 2001) first proposed an exhaustive mining algorithm to discover the strong or weak ETIs, as presented in Algorithm 1.

Algorithm 1: Exhaustive ETI Mining Algorithm

```
1: Find all dimensions $d_i$ where the global count of 1s is at least $s|T|(1 - \epsilon)$.
   Each of these dimensions forms a singleton weak ETI. Each of these
   singleton sets is called a "seed". Set $i = 1$.
2: For every seed that contains $i$ dimensions, grow it by adding a new
   dimension so that the new seed still forms a weak ETI. If one or more
   such dimensions can be found, keep all the new seeds.
3: Increment $i$ and repeat step 2 until no more growing is possible.
4: Among all seeds, pick those satisfying the strong ETI definition.
```

The algorithm mines the complete set of weak or strong ETIs. Thus the time complexity of this algorithm is exponential in the maximum number of ETI defining dimensions. To reduce time complexity, a polynomial time greedy algorithm was proposed that finds most of the ETIs by exploiting several heuristics. These heuristics are applied to step 2 in Algorithm 1 and summarized as follows.

Heuristics to Reduce Complexity

1. When looking for a dimension to grow a seed, consider only those dimensions that have been picked in step 1.

2. When testing whether a dimension can be added to a seed, the expanded seed should still define a weak ETI, and that the new dimension have at most a fraction of ϵ 0s within the weak ETI.
3. When two or more dimensions are candidates for seed growth, keep only one. Throw away the old seed once it has been grown to a new seed.

These three heuristics can greatly reduce the mining time complexity by growing greedily with the most "promising" dimensions. As a result, these heuristics also cause some ETIs to be missed. To alleviate this problem, an iterative scheme is used to heuristically search the missed ETIs.

4 An Apriori-based Approach

Recently, (Liu *et al.*, 2006) proposed a different noise-tolerant itemset model: approximate frequent itemset model. According to the model, an apriori-based mining algorithm AFI was proposed. The algorithm takes a candidate generate-and-test approach, using the apriori property to prune the search space and generating the approximate frequent itemsets in a level-wise breadth-first manner.

4.1 An Approximate Frequent Itemset Model

In the approximate frequent itemset model, besides *min_sup* constraint, two additional constraints are proposed: *row error threshold* and *column error threshold*, to control the fraction of noise in both dimensions of transactions and items, respectively. The row error threshold indicates that, for a given itemset, a supporting transaction should contain most of the items. Similarly, the column error threshold guarantees that an associated item has to appear in most of the supporting transactions in order to be included in an itemset.

The definition of approximate frequent itemset is given as follows.

Definition 3. *Let $min\_sup = s$, ϵ_r, $\epsilon_c \in [0,1]$ and the transaction database be D. An itemset $x \subseteq I$ is an approximate frequent itemset if there exists a set of transactions $T_a(x) \subseteq T$ with $|T_a(x)| \geq s|T|$, and the following two conditions hold:*

$$1.\ \forall i \in T_a(x),\ \frac{1}{|x|}\sum_{j\in x} D(i,j) \geq (1-\epsilon_r)$$

$$2.\ \forall j \in x,\ \frac{1}{|T_a(x)|}\sum_{i\in T_a(x)} D(i,j) \geq (1-\epsilon_c)$$

ϵ_r and ϵ_c are referred to as row error threshold and column error threshold, respectively. For the approximate itemset x, its supporting transaction set is denoted as $T_a(x)$ where each $t \in T_a(x)$ approximately supports x. The support is denoted as $sup_a(x) = \frac{|T_a(x)|}{|T|}$ and the absolute count is denoted as $sup\_cnt_a(x) = |T_a(x)|$.

4.2 AFI Mining Algorithm

Mining approximate frequent itemsets poses a number of algorithmic challenges beyond those faced when mining exact itemsets. The foremost difficulty is that noise-tolerant itemset mining cannot employ the anti-monotone property that has led to the success of frequent itemset mining. An important contribution by (Liu *et al.*, 2006) is the noise-tolerant Apriori property derived, which effectively enforces the anti-monotone property in the approximate itemset mining and therefore limits the search space. Another challenge is that the approximate frequent itemset criterion allows the number of errors to increase with the size of the itemset. It is therefore critical to take account of the additional errors in an itemset as its dimensionality increases while collecting the supporting transactions. These two issues and proposed solutions by (Liu *et al.*, 2006) are presented as follows respectively.

Noise-Tolerant Support Pruning

The anti-monotone property of exact frequent itemsets is the key to minimizing exponential searches in frequent itemset mining. In particular, the anti-monotone property ensures that a $(k+1)$ exact itemset can not be frequent if any one of its k sub-itmesets is not frequent. However, this property is no longer true for the approximate frequent itemsets.

(Liu *et al.*, 2006) derived a noise-tolerant support to serve as the Apriori pruning threshold, which prunes the search space effectively.

Definition 4. *Given* ϵ_r *and* ϵ_c *and* min_sup*, the noise-tolerant pruning support for a length-*k *itemset is*

$$min\_sup^k = max\{0, min\_sup \cdot \big(1 - \frac{k\epsilon_c}{\lfloor k\epsilon_r \rfloor + 1}\big)\} \qquad (2)$$

The noise-tolerant support threshold is used as the basis of a pruning strategy for the approximate itemset mining. It removes the supersets of a size-k approximate itemset x from further consideration when the frequency of x is less than min_sup^k.

0/1 Extensions

Starting with single items, the AFI algorithm generates size-$(k+1)$ itemsets from size-k itemsets in a level-wise way. The number of 0s allowed in an itemset grows with the length of the itemset in a discrete manner. If $\lfloor (k+1)\epsilon_r \rfloor > \lfloor k\epsilon_r \rfloor$, then transactions supporting the $(k+1)$-itemset are permitted one more zero than transactions supporting the k-itemset. If $\lfloor (k+1)\epsilon_r \rfloor = \lfloor k\epsilon_r \rfloor$, no additional zeros are allowed at level $(k+1)$ compared with level k. In the first case, if an additional zero is allowed at level $(k+1)$, any transaction supporting a k-itemset should also support its $(k+1)$ superset. On the other hand, for the second case when no additional zeros are allowed at level $(k+1)$, a transaction that does not support k-itemset will not have enough 1s to support its $(k+1)$ superset. These two properties are formally addressed by (Liu *et al.*, 2006) as follows.

Lemma 1. *(1-Extension) If* $\lfloor (k+1)\epsilon_r \rfloor = \lfloor k\epsilon_r \rfloor$*, then any transaction that does not support a* k*-itemset will not support its* $(k+1)$ *superset.*

In the case of 1-extension, the transaction set of a $(k+1)$-itemset is the *intersection* of the transaction sets of its length k subsets.

Lemma 2. *(0-Extension) If* $\lfloor (k+1)\epsilon_r \rfloor = \lfloor k\epsilon_r \rfloor + 1$*, then any transaction that supports a* k*-itemset also supports its* $(k+1)$ *superset.*

In the case of 0-extension, the transaction set of a $(k+1)$-itemset is the *union* of the transaction sets of its length k subsets.

0-extension and 1-extension suggest two basic steps to be taken for efficient computation and maintenance of the supporting transactions. They allow the algorithm to obtain the supporting transactions of an itemset from its subsets while avoiding the repeated scans of the original database.

AFI Mining Algorithm

The AFI algorithm uses the 0-extension and 1-extension techniques together with the noise-tolerant support-based pruning strategy to perform the mining of approximate frequent itemsets. The algorithm is presented as follows.

The level-wise mining process generates a superset of the approximate frequent itemsets AFI_p. Line 14 in Algorithm 2 further checks the generated approximate frequent itemsets w.r.t. *min_sup* and ϵ_c and filters those itemsets which fail to satisfy *min_sup* or ϵ_c. This post-processing can be done separately from the level-wise generation since it will neither benefit nor prohibit the traversing of the search space.

5 A Core Pattern Recovery Approach

5.1 Motivation

The approximate frequent itemset model proposed by (Liu *et al.*, 2006) can effectively control the percentage of noise in both dimensions of transactions and items, and the proposed AFI algorithm can successfully discover some true patterns which are distorted by the random noise. However, a large number of "uninteresting" candidates are explored during the mining process. Such candidates are uninteresting in the sense that they are spurious: they correspond to no true frequent patterns in the unobserved true transaction database. They are generated as a result of indiscriminative combination of items and

Algorithm 2: AFI Mining Algorithm

```
Input: $D$, $\epsilon_r$, $\epsilon_c$, $min\_sup$
Output: The set of approximate frequent itemsets

1: for $i = 1$ to $m$ do
2:     $T(i)$=genSupport($D$,$i$);
3: $k = 1$;
4: $L_1 = \bigcup_{i>0}^{m}\{i\}$;
5: repeat
6:     $k := k + 1$;
7:     $L_k$ :=GenCandidateItemset($L_{k-1}$, $min\_sup^{k-1}$);
8:     if ($\lfloor(k+1)\epsilon_r\rfloor = \lfloor k\epsilon_r\rfloor$)
9:       $T(L_k)$ :=1-extension($I$, $L_{k-1}$);
10:    else
11:      $T(L_k)$ :=0-extension($I$, $L_{k-1}$);
12:    $AFI_p := AFI_p \cup L_k$;
13: until $L_k = \phi$
14: $AFI$ :=filter($AFI_p$, $min\_sup$, $\epsilon_c$);
15: return $AFI$;
```

relaxed mining methods. Although some of the candidates are filtered by the error thresholds, others which pass the error check are output together with the recovered true patterns. Let's first examine Example 1.

Example 1. Table 2 shows a transaction database D. Let $min\_sup = 3$, $\epsilon_r = 1/3$ and $\epsilon_c = 1/2$.

Table 2. A Sample Purchase Database D

TID	Burger	Coke	Diet Coke
0	1	1	0
1	1	1	0
2	1	0	1
3	1	0	1

According to the approximate frequent itemset model in (Liu *et al.*, 2006), {*burger, coke, diet coke*} is discovered as an approximate frequent itemset with support 4. However, the exact support of {*burger, coke, diet coke*} in D is 0. This pattern would be deemed uninteresting since *coke* and *diet coke* are seldom purchased together in reality. Those 0s in D are not caused by random noise, but reflect the real data distribution. However, with the user-specified parameters ϵ_r and ϵ_c, it is recovered as an approximate frequent itemset by the AFI algorithm.

If the set of true frequent itemsets is treated as the ground truth, such uninteresting patterns which are excluded from the true set, are deemed as *false positive.* (Liu *et al.*, 2006) discovers a complete set of approximate frequent itemsets satisfying the error thresholds. A large number of false-positive patterns can be generated due to the exponential combinations, which make the problem computationally intractable. Even worse, if such false-positive candidates pass the error threshold check and are output as the final result, it is hard for an end user to distinguish the recovered true patterns from these false positives.

To tackle such a problem, (Cheng *et al.*, 2006) proposed a core pattern model and recovered the approximate frequent itemsets from core patterns. Intuitively, an itemset x is a core pattern if its exact support in the noisy database D is no less than αs, where $\alpha \in [0, 1]$ is a *core pattern factor*, and s is the *min_sup* threshold. A probability model was derived to estimate the probability of a true frequent pattern (in the noise-free database) remaining as a core pattern in D in the presence of noise. With some realistic parameter settings, it was shown that such true patterns remain as core patterns in the noisy database with high probability.

With the core pattern constraint, the set of core patterns are used as initial seeds for approximate itemset generation. To further reduce the output size, the concept of *approximate closed itemset* was proposed. An efficient algorithm AC-Close was developed to mine the approximate closed itemsets. A *top-down* mining strategy is exploited where the large-size approximate itemsets are discovered before the small-size ones, which makes full use of the pruning power of *min_sup* and closeness and thus, narrows down the search space dramatically.

5.2 A Core Pattern Recovery Model

First, the core pattern is defined as follows.

Definition 5. *An itemset x is a core pattern in the noisy database D if the exact support of x, $sup_e(x) \geq \alpha s$, where $\alpha \in [0, 1]$ is the core pattern factor and s is the* min_sup *threshold.*

Accordingly, the core pattern recovery model is proposed for mining the approximate frequent itemsets.

Definition 6. *Let $min\_sup = s$, ϵ_r, $\epsilon_c \in [0, 1]$ and the transaction database be D. An approximate frequent itemset x is (1) a core pattern with $sup_e(x) \geq \alpha s$, and (2) if there exists a set of transactions $T_a(x) \subseteq T$ with $|T_a(x)| \geq s|T|$, and the following two conditions hold:*

1. $\forall i \in T_a(x),\ \frac{1}{|x|}\sum_{j \in x} D(i,j) \geq (1-\epsilon_r)$

2. $\forall j \in x,\ \frac{1}{|T_a(x)|}\sum_{i \in T_a(x)} D(i,j) \geq (1-\epsilon_c)$

ϵ_r and ϵ_c are referred to as row error threshold and column error threshold, respectively.

To further reduce the number of approximate frequent itemsets discovered, a new concept of *approximate closed itemsets* was proposed.

Definition 7. *An approximate frequent itemset x is closed if there exists no itemset y such that (1) y is a proper superset of x, and (2) $sup_a(y) \geq sup_a(x)$.*

An important difference between the closeness definition for approximate itemset and that for exact itemset (Pei *et al.*, 2000, Zaki and Hsiao, 2002) is the second condition in Definition 7. From the AFI algorithm, one should note that, given two approximate frequent itemsets x and y where $x \subseteq y$, either of the following two conditions holds: (1) $sup_a(x) \geq sup_a(y)$, and (2) $sup_a(x) < sup_a(y)$. The second condition holds because of the "0-extension" effect, *i.e.*, compute a size-k itemset's supporting transactions by taking the *union* of all supporting transactions of its size-$(k-1)$ sub-itemsets. Considering such a factor, an approximate frequent itemset x is non-closed if there is a superset y such that $sup_a(y) \geq sup_a(x)$. The itemset x, to some extent, is deemed less interesting, since there exists a super-pattern y which subsumes x in both directions of items and transactions.

The problem of *approximate closed itemset* mining from core patterns is the mining of all itemsets which are (1) core patterns w.r.t. α; (2) approximate frequent itemsets w.r.t. ϵ_r, ϵ_c and *min_sup*; and (3) closed.

5.3 Approximate Closed Itemsets Mining

First of all, some theoretical analysis of the core pattern recovery model is provided, by modelling the probability of a true frequent pattern being recovered as a core pattern in the database with random noise. This model shows that, the core pattern recovery mechanism can effectively recover the true patterns with high probability, at some realistic noise level. We will discuss how to generate the candidate approximate itemsets from the core patterns with ϵ_r, ϵ_c and *min_sup*, and develop several pruning strategies as well.

A Probabilistic Model

With the core pattern constraint, a set of core patterns (denoted as $\mathcal{C}$) can be discovered as the initial seeds for approximate itemset generation. Each core pattern $x \in \mathcal{C}$ is extended into an approximate one by allowing some noise. Before developing the techniques for generating approximate itemsets, we address the question of whether it is reasonable to assume the core pattern constraint. Since it is possible that a true frequent pattern x fails the core pattern requirement ($sup_e(x) \geq \alpha s$), and thus is excluded from the final result set, this approach could miss certain true frequent patterns. One may ask, how likely is it to miss a true frequent pattern by focusing on the core pattern only? The following lemma provides an answer to this question.

Lemma 3. *Assume an unobserved true database D_t has N transactions and $min\_sup = s$. A true frequent pattern x of size l, has exact support count $sup\_cnt_e(x, D_t) = n \geq sN$. Assume the core pattern factor is α. If, in the presence of random noise, each entry in D_t is flipped independently from 1 to 0 with a probability p, resulting in the noisy database D. Then the probability of x remaining in D with $sup_e(x) \geq \alpha s$ is*

$$P(sup_e(x) \geq \alpha s) = I_{(1-p)^l}(\alpha sN, n - \alpha sN + 1)$$

where $I_{(1-p)^l}(\alpha sN, n - \alpha sN + 1)$ is the regularized incomplete beta function (Abramowitz and Stegun, 1964, Press et al., 1992).

Proof. For each transaction t out of the n which supports x, the probability of t still supporting x, in the presence of noise, is $(1-p)^l$ and the probability of not supporting x is $1-(1-p)^l$. Each such transaction flipping is a Bernoulli trial. Obtaining k supporting transactions out of n where $\alpha sN \leq k \leq n$ corresponds to $sup\_cnt_e(x, D) = k$, $k \in [\alpha sN, n]$. Then the probability of $sup\_cnt_e(x, D) = k$ is

$$P(sup\_cnt_e(x, D) = k) = \binom{n}{k}(1-p)^{kl}(1-(1-p)^l)^{(n-k)} \tag{3}$$

Eq. (3) follows a binomial distribution. Summing over all $k \in [\alpha sN, n]$ on Eq. (3) derives the probability of $sup_e(x) \geq \alpha s$ in D, as follows.

$$\begin{aligned} P(sup_e(x) \geq \alpha s) &= \sum_{k=\alpha sN}^{n} P(sup\_cnt_e(x, D) = k) \\ &= \sum_{k=\alpha sN}^{n} \binom{n}{k}(1-p)^{kl}(1-(1-p)^l)^{(n-k)} \\ &= I_{(1-p)^l}(\alpha sN, n - \alpha sN + 1) \\ &= \frac{B((1-p)^l; \alpha sN, n - \alpha sN + 1)}{B(\alpha sN, n - \alpha sN + 1)} \end{aligned}$$

where $B(\alpha sN, n-\alpha sN+1)$ is the beta function and $B((1-p)^l; \alpha sN, n-\alpha sN+1)$ is the incomplete beta function (Abramowitz and Stegun, 1964,Press *et al.*, 1992).

An application of Lemma 3 to some realistic assumption over some transaction database clearly shows, it is with high probability that a true frequent pattern x still appears in D with $sup_e(x) \geq \alpha s$ in the presence of random noise. For example, for $N = 1,000,000$, $s = 0.01$, $n = 12,000$, $l = 5$, and $p = 0.05$. For $\alpha \leq 0.9$, $P(sup_e(x) \geq \alpha s) = 99.99\%$; for $\alpha = 0.92$, $P(sup_e(x) \geq \alpha s) = 96.92\%$. Therefore, with the random noise level of $p = 0.05$, an itemset x has exact count no less than $9,000$ in D with probability 99.99%, and has exact count no less than 9,200 with probability 96.92%.

Candidate Approximate Itemset Generation

Based on Lemma 3, an efficient algorithm can be designed to discover the approximate frequent itemsets from core patterns. It first mines the set of core patterns from D with $min\_sup = \alpha s$. These patterns are treated as the initial seeds for possible further extension to approximate frequent itemsets. In this section, we discuss how to generate the candidate approximate frequent itemsets from the set of core patterns.

Assume we have discovered a set of core patterns with $min\_sup = \alpha s$. Let $\mathcal{C}$ be the set of core patterns. A lattice $\mathcal{L}$ is built over $\mathcal{C}$. Example 2 is used to illustrate the process.

Example 2. Table 3 shows a sample transaction database D with 7 transactions and 4 items $\{a, b, c, d\}$. Let $\epsilon_r = 0.5$, $\epsilon_c = 0.5$, the absolute $min\_sup = 3$, and $\alpha = 1/3$.

Table 3. A Transaction Database D

TID	a	b	c	d
0	1	1	1	0
1	1	0	0	0
2	1	1	1	1
3	0	0	1	1
4	1	1	0	0
5	1	0	1	1
6	0	1	1	1

Mining the core patterns from D with the absolute support $\alpha * min\_sup = 1$ generates 15 core patterns. Figure 2 shows the lattice of core patterns, where the closed core patterns are in bold.

First, for the size-4 core pattern $\{a, b, c, d\}$, the number of 0s allowed in a supporting transaction is $\lfloor 4 * 0.5 \rfloor = 2$ given $\epsilon_r = 0.5$. Traverse upward in the lattice for 2 levels (*i.e.*, levels 2 and 3), which constitute the *extension space* for $\{a, b, c, d\}$. The extension space for a core pattern is defined as follows.

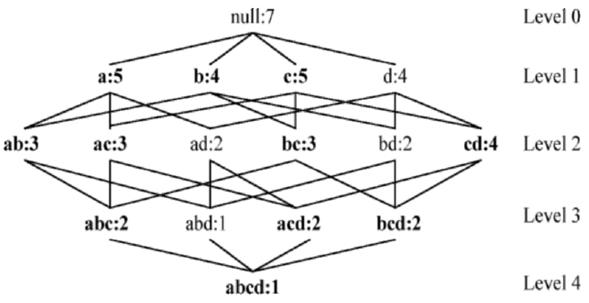


Fig. 2. The Lattice $\mathcal{L}$ of Core Patterns

Definition 8. *For a core pattern y with size l, the extension space, denoted as* $ES(y)$*, is a set of sub-patterns of y from level* $(\lceil l * (1 - \epsilon_r) \rceil)$ *to level* $(l - 1)$ *in the lattice, given* $\epsilon_r \in [0, 1]$.

Because of the ϵ_r fraction of errors allowed in a transaction, for a core itemset y and each sub-pattern $x \in ES(y)$, any transaction supporting x also approximately supports y. According to this property, the transaction set $T_a(y)$ of y is the *union* of the exact transaction set $T_e(x)$ of each sub-pattern $x \in ES(y)$. Thus, for $\{a, b, c, d\}$, its transaction set $T_a(\{a, b, c, d\})$ is the union of the exact transaction sets of all 10 itemsets at levels 2 and 3 of Figure 2.

To optimize the union operation, only the closed core patterns in the extension space are included, while those non-closed ones are excluded for union. This is because, the exact transaction sets of non-closed sub-patterns will not contribute to the approximate transaction set of the core pattern. For the non-closed itemset $\{a, b, d\}$ at level 3, $T_e(\{a, b, d\}) = T_e(\{a, b, c, d\}) = \{2\}$. So it will not contribute any additional transactions to $T_a(\{a, b, c, d\})$ through the union operation. It is the same for $\{a, d\}$ and $\{b, d\}$ at level 2, since they are subsumed by $\{a, c, d\}$ and $\{b, c, d\}$, respectively. This property is stated formally in Lemma 4.

Lemma 4. *For a candidate approximate itemset y and each closed core pattern* $x \in ES(y)$*, the approximate transaction set* $T_a(y)$ *is computed by taking the union of the exact transaction set* $T_e(x)$ *of x.*

By taking the union of the exact transaction sets of the 7 closed patterns at levels 2 and 3, the approximate transaction set $T_a(\{a, b, c, d\}) = \{0, 2, 3, 4, 5, 6\}$. Thus, $\{a, b, c, d\}$ is a candidate approximate itemset since it satisfies *min_sup* and ϵ_r. It needs to be checked w.r.t. ϵ_c and closeness, which will be discussed separately.

To summarize, the steps to identify candidate approximate itemsets include:

1. For each core pattern y in the lattice, identify its extension space $ES(y)$ according to ϵ_r;

2. Pick the closed sub-patterns of y from $ES(y)$, take the union of the exact transaction sets of them;
3. Check against *min_sup*. Keep those which satisfy *min_sup* as candidate approximate frequent itemsets.

Pruning by ϵ_c

After the candidate generation, all the candidate approximate itemsets are identified which satisfy *min_sup* and ϵ_r. The next step is to check a candidate approximate itemset x w.r.t. the column error threshold ϵ_c. According to the second condition in Definition 3, to check the ϵ_c constraint, it needs to scan the approximate transaction set $T_a(x)$ and accumulate the count of each item in x. However, a pruning strategy, referred to as ϵ_c *early pruning*, allows us to identify some candidates which violate the ϵ_c constraint without scanning $T_a(x)$. This pruning strategy is stated as follows.

Lemma 5. *Let $x = i_1 ... i_n$ be an itemset and the exact support of a single item $i_k \in x$ in D be $sup_e(i_k)$, $k \in [1, n]$. Let the support of the approximate pattern x be $sup_a(x)$. If*

$$\exists i_k \in x, \frac{sup_e(i_k)}{sup_a(x)} < (1 - \epsilon_c)$$

satisfies, then x cannot pass the ϵ_c check.

Proof. Let the exact support count of item i_k in the transaction set T_a be $sup\_cnt_e(i_k, T_a)$, then

$$\frac{sup\_cnt_e(i_k, T_a)}{|T_a|} \le \frac{sup\_cnt_e(i_k, D)}{|T_a|} = \frac{sup_e(i_k)}{sup_a(x)}$$

If $\frac{sup_e(i_k)}{sup_a(x)} < (1 - \epsilon_c)$, then $\frac{sup\_cnt_e(i_k, T_a)}{|T_a|} < (1 - \epsilon_c)$, which violates Definition 2. As a result, item i_k cannot pass the ϵ_c check. The ϵ_c early pruning is effective especially when ϵ_c is small, or there exists an item in x with very low exact support in D. If the pruning condition is not satisfied, a scan on $T_a(x)$ is performed for the ϵ_c check.

For a candidate pattern x which violates the ϵ_c constraint, we can either prune it or find a maximal subset of $T_a(x)$ which satisfies ϵ_c. A heuristic algorithm for finding a maximal AFI is introduced in (Liu *et al.*, 2005).

Top-down Mining and Pruning by Closeness

The previous steps focus on how to generate the individual candidate approximate itemset. In this part, an efficient *top-down* search strategy is designed as the mining framework, which enables effective pruning by the closeness definition and the *min_sup* threshold.

A top-down mining strategy is taken over the lattice $\mathcal{L}$, such that the mining starts with the largest pattern in $\mathcal{L}$ and proceeds level by level, in the size decreasing order of core patterns. Let's look at an example.

Example 3. Mining on the lattice $\mathcal{L}$ in Figure 2 starts with the size-4 pattern $\{a, b, c, d\}$. Since the number of 0s allowed in a transaction is $\lfloor 4 * 0.5 \rfloor = 2$, its extension space includes closed patterns at levels 2 and 3, *i.e.*, $ES(\{a, b, c, d\}) = \{abc : 2, acd : 2, bcd : 2, ab : 3, ac : 3, bc : 3, cd : 4\}$. The transaction set $T_a(\{a, b, c, d\})$ is computed by taking the union of the transaction sets of the above 7 patterns. Further checking shows that $\{a, b, c, d\}$ satisfies *min_sup* and ϵ_c as well. Since $\{a, b, c, d\}$ is the approximate itemset of the largest size, it is an approximate closed itemset.

When the mining proceeds to level 3, for example, the size-3 pattern $\{a, b, c\}$. The number of 0s allowed in a transaction is $\lfloor 3 * 0.5 \rfloor = 1$, so its extension space includes its closed sub-patterns at level 2, *i.e.*, $ES(\{a, b, c\}) = \{ab : 3, ac : 3, bc : 3\}$. The transaction set $T_a(\{a, b, c\})$ is computed by taking the union of the transaction sets of the above 3 patterns.

Since $ES(\{a, b, c\}) \subseteq ES(\{a, b, c, d\})$ holds, $T_a(\{a, b, c\}) \subseteq T_a(\{a, b, c, d\})$ holds too. In this case, after the computation on $\{a, b, c, d\}$, we can prune $\{a, b, c\}$ without actual computation with either of the following two conclusions: (1) if $\{a, b, c, d\}$ satisfies the *min_sup* threshold and the ϵ_c constraint, then no matter whether it is closed or non-closed, $\{a, b, c\}$ can be pruned because it will be a non-closed approximate itemset; or (2) if $\{a, b, c, d\}$ does not satisfy the *min_sup* threshold, then $\{a, b, c\}$ can be pruned because it will not satisfy the *min_sup* threshold either. In the first condition, we say *no matter whether* $\{a, b, c, d\}$ *is closed or non-closed,* $\{a, b, c\}$ *is non-closed.* This is because, if $\{a, b, c, d\}$ is closed, then $\{a, b, c\}$ is subsumed by $\{a, b, c, d\}$ and thus is non-closed; if $\{a, b, c, d\}$ is non-closed, then there must exist a closed approximate super-pattern which subsumes $\{a, b, c, d\}$, and then subsumes $\{a, b, c\}$ as well. Thus $\{a, b, c\}$ is non-closed.

Similarly, the other three core patterns at level 3, $\{abd : 1, acd : 2, bcd : 2\}$ can be pruned after the computation on $\{a, b, c, d\}$.

We refer to the pruning technique in Example 3 as *forward pruning*, which is formally stated in Lemma 6.

Lemma 6. *If* $\lfloor (k+1) \cdot \epsilon_r \rfloor = \lfloor k \cdot \epsilon_r \rfloor + 1$*, after the computation on a size-*$(k+1)$ *pattern is done, all its size-k sub-patterns in the lattice* $\mathcal{L}$ *can be pruned with either of the following two conclusions: (1) if the size-*$(k+1)$ *pattern satisfies min_sup and* ϵ_c*, then the size-k patterns can be pruned because they are non-closed; or (2) if the size-*$(k+1)$ *pattern does not satisfy min_sup, then the size-k patterns can be pruned because they do not satisfy min_sup either.*

Forward pruning, naturally integrated with the top-down mining, can reduce the search space dramatically due to the *min_sup* threshold and the closeness constraint.

Another pruning strategy, called *backward pruning*, is proposed as well to ensure the closeness constraint. Let's look at an example before stating it formally.

Example 4. When the mining proceeds to level 2, for example, the core pattern $\{a, d\}$ is extended to a candidate approximate pattern, with the transaction set $T_a(\{a, d\}) = \{0, 1, 2, 3, 4, 5, 6\}$. Since it satisfies *min_sup* and ϵ_c, it has to be checked against all its approximate closed super-patterns for its closeness. In this case, the approximate closed super-pattern $\{a, b, c, d\}$ is checked with $T_a(\{a, b, c, d\}) = \{0, 2, 3, 4, 5, 6\}$. Since $|T_a(\{a, d\})| > |T_a(\{a, b, c, d\})|$, $\{a, d\}$ is an approximate closed itemset.

Lemma 7 formally states the backward pruning technique.

Lemma 7. *If a candidate approximate pattern x satisfies min_sup, ϵ_r and ϵ_c, it has to be checked against each approximate closed itemset y where $x \subseteq y$. If there exists no approximate closed itemset y such that $x \subseteq y$ and $sup_a(x) \leq sup_a(y)$, then x is an approximate closed itemset.*

5.4 AC-Close Algorithm

Integrating the top-down mining and the various pruning strategies, an efficient algorithm AC-Close was developed in (Cheng *et al.*, 2006) to mine the approximate closed itemsets from the core patterns, presented in Algorithm 3.

In Algorithm 3, ACI represents the final set of approximate closed itemsets, $T_a(x)$ is the approximate transaction set of an itemset x, C_k is the set of size-k candidate approximate itemsets, and L_k is the set of size-k approximate closed itemsets. $forwardPrune(C_{k-1}, x)$ prunes the sub-patterns of x from the size-$(k-1)$ candidate set before computation proceeds to them, according to Lemma 6. $backwardPrune(C_k, ACI)$ prunes the non-closed approximate itemsets from C_k, according to Lemma 7. "$*$" at line 9 means the ϵ_c early pruning is applied for checking the ϵ_c constraint. If it applies, there is no need to scan the transaction set $T_a(x)$ for the ϵ_c checking.

6 Experimental Study

In this section, experimental results reported by (Cheng *et al.*, 2006) are presented to compare different mining algorithms. Since it is shown by (Liu *et al.*, 2006) that AFI systematically outperforms ETI in terms of both efficiency and result quality, the comparison between AFI and ETI is omitted. The results reported in this section focus on the comparison between AFI and AC-Close. For a detailed experimental results between AFI and ETI please refer to (Liu *et al.*, 2006).

Algorithm 3: The AC-Close Algorithm

```
Input: $D$, $min\_sup = s$, $\epsilon_r$, $\epsilon_c$, $\alpha$
Output: $ACI$: approximate closed itemsets

1: $\mathcal{C}$ = genFreqItemset($D$, $\alpha s$);
2: $\mathcal{L}$ = buildLattice($\mathcal{C}$);
3: $k$ = max level of $\mathcal{L}$;
4: $C_k = \{x|x \in \mathcal{L}\ and\ size(x) = k\}$;
5: repeat
6:     for $x \in C_k$
7:         $ES$ = genExtensionSpace($x$, $\mathcal{L}$);
8:         $T_a(x)$ = unionTransaction($ES$);
9:         if ($x$ not satisfies $s$ or $\epsilon_c$)$^*$
10:            $C_k = C_k - \{x\}$;
11:        if($\lfloor k \cdot \epsilon_r \rfloor = \lfloor (k-1) \cdot \epsilon_r \rfloor + 1$)
12:            $C_{k-1}$ = forwardPrune($C_{k-1}, x$);
13:    end for
14:    $L_k$ = backwardPrune($C_k$, $ACI$);
15:    $ACI = ACI \cup L_k$;
16:    $k = k - 1$;
17: until $k = 0$
18: return $ACI$
```

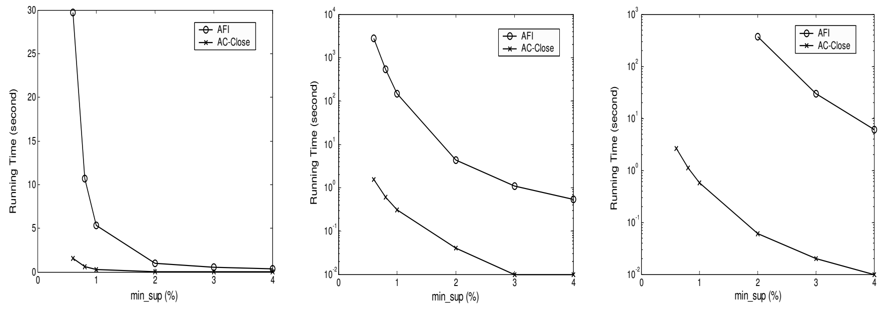
Experiment is carried out both on synthetic datasets and the UCI datasets. Efficiency and result quality of both algorithms are evaluated and reported. Both algorithms are coded in Microsoft Visual C++ and experiments were run on a 3GHz PC with 2GB memory.

Three groups of experiments were conducted for the performance comparisons. The first tested the efficiency and scalability of AFI and AC-Close w.r.t. various parameters. The second group tested the result quality of AFI and AC-Close on synthetic datasets with a controlled fraction of random noise embedded and with known underlying patterns. Finally, both algorithms were applied to a UCI dataset with known underlying patterns.

6.1 Scalability

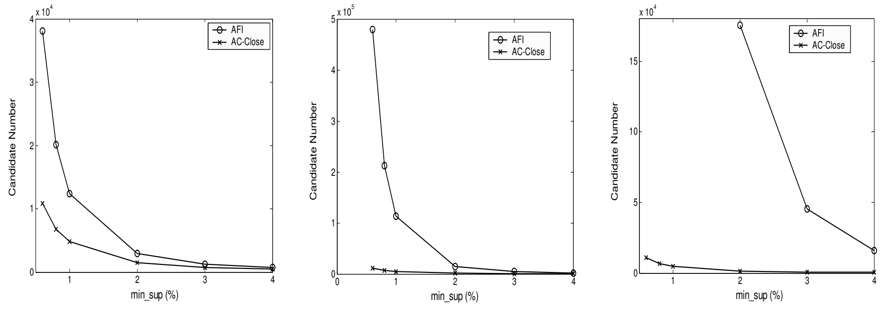
The IBM synthetic data generator is used to generate synthetic datasets for the scalability test. A dataset T10.I100.D20K (Agrawal and Srikant, 1994) is generated with 20K transactions, 100 distinct items and an average of 10 items per transaction.

Figure 3 shows the running time of both algorithms by varying *min_sup*. The three figures show the performance with $\epsilon_r = \epsilon_c = 0.15,\ 0.20,\ 0.25$ respectively. $\alpha = 0.8$ is used in AC-Close. In all three cases, AC-Close runs



(a) $\epsilon_r = \epsilon_c = 0.15$ (b) $\epsilon_r = \epsilon_c = 0.20$ (c) $\epsilon_r = \epsilon_c = 0.25$

Fig. 3. Running Time of AFI and AC-Close, varying *min_sup*, on T10.I100.D20K



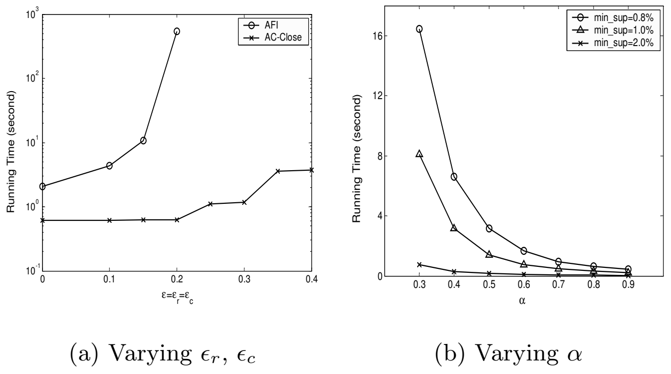
(a) $\epsilon_r = \epsilon_c = 0.15$ (b) $\epsilon_r = \epsilon_c = 0.20$ (c) $\epsilon_r = \epsilon_c = 0.25$

Fig. 4. Number of Candidates Generated by AFI and AC-Close, varying *min_sup*, on T10.I100.D20K

much faster than AFI. In addition, the time difference increases as ϵ_r and ϵ_c increase.

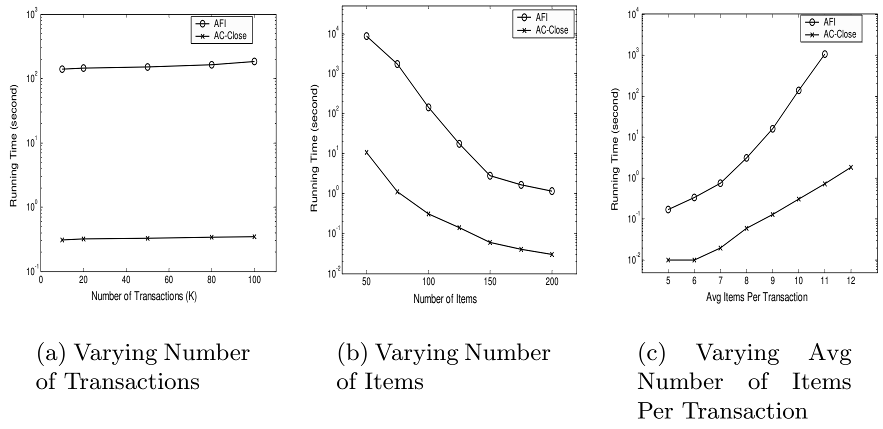
Figure 4 shows the number of candidate approximate frequent itemsets generated by both algorithms during the same experiment in Figure 3. In AFI, the candidates are the patterns generated during the mining process where the size-k candidates satisfy the noise-tolerant pruning support min_sup^k. In AC-Close, the candidates are the core patterns. In all three cases, AFI consistently generates far more candidates than AC-Close. This result shows that AFI has a much larger search space than AC-Close.

Figure 5 shows the running time of both algorithms by varying ϵ_r, ϵ_c and α respectively. To reduce the parameter space, in Figure 5 (a), we set $\epsilon = \epsilon_r = \epsilon_c$. $min\_sup = 0.8\%$ and $\alpha = 0.8$ are used in this experiment. The



(a) Varying ϵ_r, ϵ_c (b) Varying α

Fig. 5. Running Time of AFI and AC-Close, varying ϵ_r, ϵ_c and α, on T10.I100.D20K



(a) Varying Number of Transactions (b) Varying Number of Items (c) Varying Avg Number of Items Per Transaction

Fig. 6. Other Scalability Tests

running time of AFI increases very quickly as ϵ increases while the efficiency of AC-Close is not affected too much by ϵ. The size of approximate itemsets increases by allowing more noise in an itemset. So more candidate itemsets are generated by AFI and the computation time increases exponentially w.r.t. the itemset size. Since AC-Close still focuses on mining the core pattern set as ϵ increases, the candidate set remains the same.

Figure 5 (b) shows the running time of AC-Close by varying the core pattern factor α. $\epsilon_r = \epsilon_c = 0.20$ is used. As α decreases, the core pattern set becomes larger. Therefore, more candidates are generated for approximate itemset mining and the computation time increases. Nevertheless, AC-Close is shown to be very efficient even when α is set to as low as 0.3.

Figure 6 shows the scalability tests by varying different statistics of the transaction database. In all three experiments, $min\_sup = 1\%$, $\epsilon_r = \epsilon_c = 0.20$ and $\alpha = 0.8$ are used.

Figure 6 (a) shows the running time by varying the number of transactions. The number of distinct items is 100 and the average items per transaction is 10. Since both algorithms only perform the union or intersection operations on the transaction id lists without scanning the database repeatedly, increasing the transaction number does not affect the performance much.

Figure 6 (b) shows the running time by varying the number of distinct items in the database. The number of transactions is 10K and the average transaction length is 10. When the number of items is large, the database becomes sparse. So both algorithms run more efficiently. As the number of items decreases, the database becomes dense and the number of qualified approximate frequent itemsets increases. Figure 6 (b) shows that the running time of AFI increases more rapidly than that of AC-Close.

Figure 6 (c) shows the running time by varying the average number of items per transaction. The number of transactions is 10K and the number of distinct items is 100. As the average transaction length increases, the combination between items increases exponentially. The figure shows that the running time of AFI increases much faster.

6.2 Quality

Besides the scalability tests, the quality of mining results are also compared between AFI and AC-Close. A synthetic dataset D_t, T10.I100.D20K, is used as the noise-free dataset. Noise is introduced to flip each entry of D_t with a probability p. The noisy dataset is denoted as D. Exact frequent pattern mining is applied to D_t and the exact frequent itemsets are treated as the ground truth, denoted as F_{true}. In addition, AFI and AC-Close are applied to the noisy dataset D and the approximate itemsets are treated as the recovered patterns, denoted as F_{apr}. For comparison purpose, AC-Close outputs the approximate frequent itemsets instead of the closed ones. Two evaluation metrics *precision* and *recall* are used as the measure, as defined below.

$$precision = \frac{|F_{true} \cap F_{apr}|}{|F_{apr}|}, \quad recall = \frac{|F_{true} \cap F_{apr}|}{|F_{true}|}$$

Tables 4 and 5 show the quality comparison between AFI and AC-Close, with $p = 0.05$ and $p = 0.20$ respectively. $\epsilon_r = \epsilon_c = 0.20$ and $\epsilon_r = \epsilon_c = 0.25$ are set in the two cases respectively. In both cases, $\alpha = 0.8$ is used.

In Table 4, AFI achieves the same recall values as AC-Close but AFI also generates some false positive patterns which do not appear in the true pattern set. In contrast, the precision of AC-Close is 100%. This shows that the core pattern approach can effectively recover the true patterns from the noisy dataset.

Table 4. Precision and Recall of Mining Result by AFI and AC-Close, Noise Level $p = 0.05$

***min_sup*(%)**	**Precision**		**Recall**	
	AFI	AC-Close	AFI	AC-Close
0.6	90.04	100	76.70	76.70
0.8	90.46	100	78.61	78.61
1.0	91.69	100	78.30	78.30
2.0	97.52	100	81.71	81.71
3.0	99.40	100	81.61	81.61
4.0	100	100	81.27	81.27

Table 5. Precision and Recall of Mining Result by AFI and AC-Close, Noise Level $p = 0.20$

***min_sup*(%)**	**Precision**		**Recall**	
	AFI	AC-Close	AFI	AC-Close
0.6	60.51	100	33.64	33.13
0.8	63.22	100	34.68	34.35
1.0	62.84	100	35.56	35.52
2.0	80.87	100	40.18	40.18
3.0	87.37	100	40.89	40.89
4.0	89.66	100	44.96	44.96

In Table 5, the noise level $p = 0.20$ is higher. To recover the true patterns, the error threshold $\epsilon_r = \epsilon_c$ is set to 0.25. In all cases, the recall of AC-Close is either the same or very close to that of AFI. However, with the error threshold setting, AFI generates many more false positives, as indicated by the precision measure.

Table 6. Precision and Recall of the Mining Result by AC-Close, varying α, $p = 0.05$

α	*min_sup*=0.8%		*min_sup*=1.0%	
	Precision	Recall	Precision	Recall
0.9	100	78.61	100	78.30
0.8	100	78.61	100	78.30
0.7	100	78.61	100	78.30
0.6	100	78.61	100	78.30
0.5	100	78.61	100	78.30
0.4	99.96	78.61	100	78.30
0.3	99.81	78.61	99.94	78.30
AFI	90.46	78.61	91.69	78.30

Table 7. Precision and Recall of the Mining Result by AC-Close, varying α, $p = 0.20$

α	*min_sup*=0.8%		*min_sup*=1.0%	
	Precision	Recall	Precision	Recall
0.9	100	34.35	100	35.52
0.8	100	34.35	100	35.52
0.7	100	34.35	100	35.52
0.6	100	34.35	100	35.52
0.5	100	34.38	100	35.52
0.4	99.95	34.61	99.93	35.54
0.3	99.11	34.68	98.67	35.56
AFI	63.22	34.68	62.84	35.56

Tables 6 and 7 show the result quality of AC-Close at different α values, on T10.I100.D20K, at a noise level $p = 0.05$ and $p = 0.20$ respectively. $\epsilon_r = \epsilon_c = 0.20$ and $\epsilon_r = \epsilon_c = 0.25$ are used in the two cases accordingly. At the bottom row of each table, the precision and recall of AFI is shown as the comparison baseline. In Table 6 when the noise level is low, a low α value does not increase the recall (*i.e.*, discover more true patterns), but generates some false positive. This is because, the true patterns and their support are not affected much by the noise. Therefore, with a low level of random noise, it is not necessary to try a low α value. In Table 7 when the noise level is higher, a low α slightly increases the recall, but also generates some false positives. In both cases, the recall is either the same or very close to that of AFI but the precision is much higher than that of AFI.

6.3 Zoo Dataset

In this experiment, AFI and AC-Close are applied to the Zoo dataset from the UCI Machine Learning Repository (UCI, 2007). The Zoo dataset contains 101 instances and each instance has 15 boolean attributes and a class label (*mammal*, *bird*, *etc.*). The 15 boolean attributes including *hair*, *feathers* and *eggs* are used as items.

One mining task is to discover the common features of a set of animals in the same class. For example, mammals produce milk, are covered in hair, are toothed and grow tails. However, not every mammal exhibits these common features: platypuses lack teeth and dolphins are hairless. If such exceptions are not tolerated, it is hard to find the complete set of features that characterize a class.

For testing purposes, we adopted the 7 classes into which the instances were already categorized as the true underlying pattern. Then we examined how well the competing mining methods recovered these classes. We focused on the 4 classes with at least 5 instances.

Both AFI and AC-Close are tested in terms of the running time and result quality. $\epsilon_r = 0.25$, $\epsilon_c = 0.30$ and the absolute $min\_sup = 12$ are used. The core pattern factor is set to $\alpha = 0.5$. The running time of AFI is $1,471.973$ seconds and that of AC-Close is 1.594 seconds, almost 1000 times faster than AFI.

To measure the result quality, for each approximate pattern x discovered, we compare its supporting transaction set $T_a(x)$ against the transactions of the true class. The approximate pattern with the highest similarity to the true class is presented. Table 8 shows the transaction size (denoted as $|T_c|$) of each class and the transaction size (denoted as $|T_a|$) of the approximate patterns. For example, in Table 8, the true class of *mammal* has 41 instances, which correspond to the 41 transactions in the Zoo data belonging to the mammal class. Both AFI and AC-Close can discover an approximate frequent itemset which is approximately supported by the 41 transactions. As shown in Table 8, both algorithms can discover three classes with 100% match. For the fourth class, the approximate pattern of AFI has 12 transactions while that of AC-Close has 13 transactions.

Table 8. True Pattern Recovery in Zoo Dataset

Class	True Pat	AFI	AC-Close
mammal	41	41	41
bird	20	20	20
fish	13	13	13
sea creature	10	12	13

7 Related Work on Approximate Frequent Itemset

Other related studies on approximate frequent patterns include (Mannila and Toivonen, 1996, Boulicaut *et al.*, 2000, Pei *et al.*, 2003, Seppänen and Mannila, 2004, Steinbach *et al.*, 2004, Zhu *et al.*, 2007, Yan *et al.*, 2007). (Mannila and Toivonen, 1996) showed that approximate association rules are interesting and useful. (Boulicaut *et al.*, 2000) proposed the concept of free-sets and led to an error-bound approximation of frequencies.

The goal of (Pei *et al.*, 2003) is to derive a *condensed frequent pattern base*, a compact representation from which the support of all other frequent patterns can be approximated within some fixed error bound. This work emphasizes more on the compression issue of frequent patterns.

Seppänen and Mannila (Seppänen and Mannila, 2004) proposed to mine the dense itemsts in the presence of noise. A dense itemset is an itemset with a sufficiently large sub-matrix that exceeds a given density threshold of attributes present.

The support envelope technique proposed by Steinbach et al. (Steinbach *et al.*, 2004) identifies regions of the data matrix where each transaction contains at least a given number of items and each item appears in at least a given number of transactions. The support envelope is a tool for exploration and visualization of the high-level structures of association patterns in a transaction database. A symmetric ETI model is proposed such that the same fraction of errors are allowed in both rows and columns. However, no additional properties or algorithms are proposed to mine the symmetric ETIs.

A recent study by Zhu et al. (Zhu *et al.*, 2007) aimed at mining the *colossal* frequent patterns which are frequent itemsets of rather large size. The problem formulation in this study is not itemset recovery from noise, but efficient discovery of colossal patterns. A novel mining approach called *Pattern-Fusion* was proposed to efficiently find a good approximation to the colossal patterns. With this approach, a colossal pattern is discovered by fusing its small core patterns in one step, whereas the incremental pattern-growth mining strategies, such as those adopted in Apriori or FP-growth, have to examine a large number of mid-sized ones.

Yan et al. (Yan *et al.*, 2007) proposed a noise-tolerant model for mining frequent dense subgraphs across multiple graphs. This approach is applied for efficiently and systematically identifying frequent coexpression clusters. Given m microarray datasets, they model each microarray dataset as a coexpression graph, and search for densely connected subgraphs. Due to the noise and outliers in data and the unavoidable cutoff selection for edge construction, the exact match criterion is relaxed, otherwise pursuing exact match would overlook some coexpressed clusters. Therefore, instead of requiring the recurrence of the exact dense subgraph, it only requires the connectivity among the gene set to be higher than a threshold.

8 Conclusions

Frequent itemset mining is one of the fundamental tasks in data mining. In real applications where the data is typically subject to random noise or measurement error, exact frequent itemset mining no longer meets the needs of discovering the true patterns from the noisy dataset. A noise-tolerant mining model is the key solution in this application scenario.

In this chapter, we overview several different noise-tolerant mining models and the methodologies for efficient mining of approximate frequent itemsets. Our experimental study compares the mining efficiency and result quality of different mining approaches.

There are still many interesting issues to be further studied in the noise-tolerant mining models and their applications, including depth-first mining methods for approximate frequent itemsets; design of noise-tolerant models for more complicated patterns such as sequences or structures; subspace clustering in high-dimensional data and gene expression data as well as approximate itemset-based classification models in the presence of noise.

References

M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* Dover, 1964.

R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. SIGMOD'93*, pages 207–216, May 1993.

R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. VLDB'94*, pages 487–499, Sept. 1994.

R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proc. of SIGMOD*, pages 439–450, 2000.

R. J. Bayardo. Efficiently mining long patterns from databases. In *Proc. SIGMOD'98*, pages 85–93, June 1998.

J.F. Boulicaut, A. Bykowski, and C. Rigotti. Approximation of frequency queries by means of free-sets. In *Principles of Data Mining and Knowledge Discovery*, pages 75–85, 2000.

D. Burdick, M. Calimlim, and J. Gehrke. MAFIA: A maximal frequent itemset algorithm for transactional databases. In *Proc. ICDE'01*, pages 443–452, April 2001.

H. Cheng, X. Yan, J. Han, and C. Hsu. Discriminative frequent pattern analysis for effective classification. In *Proc. 2007 Int. Conf. Data Engineering (ICDE'07)*, Istanbul, Turkey, April 2007.

H. Cheng, P. S. Yu, and J. Han AC-Close: Efficiently Mining Approximate Closed Itemsets by Core Pattern Recovery. In *Proc. of ICDM*, pages 839–844, 2006.

G. Cong, K. Tan, A. Tung, and X. Xu. Mining top-k covering rule groups for gene expression data. In *Proc. of SIGMOD*, pages 670–681, 2005.

FIMI: Frequent itemset mining implementations repository. http://fimi.cs.helsinki.fi, 2003.

J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. SIGMOD'00*, pages 1–12, May 2000.

W. Li, J. Han, and J. Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. In *Proc. of ICDM*, pages 369–376, 2001.

B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proc. of KDD*, pages 80–86, 1998.

J. Liu, S. Paulsen, W. Wang, A. Nobel, and J. Prins. Mining approximate frequent itemset from noisy data. In *Technical report, Department of Computer Science, TR05-015*, 2005.

J. Liu, S. Paulsen, X. Sun, W. Wang, A. Nobel, and J. Prins. Mining approximate frequent itemsets in the presence of noise: Algorithm and analysis. In *Proc. SDM'06*, pages 405–416, April 2006.

H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In *Knowledge Discovery and Data Mining*, pages 189–194, 1996.

J. Pei, G. Dong, W. Zou, and J. Han. Mining condensed frequent pattern bases. In *Knowledge and Information Systems*, volume 6 of *5*, pages 570–594, 2004.

J. Pei, J. Han, and R. Mao. CLOSET: An efficient algorithm for mining frequent closed itemsets. In *Proc. DMKD'00*, pages 11–20, May 2000.

W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C.* Cambridge, 2nd edition, 1992.

J. Seppänen and H. Mannila. Dense itemsets. In *Proc. of KDD*, pages 683–688, 2004.

M. Steinbach, P. Tan, and V. Kumar. Support envelopes: A technique for exploring the structure of association patterns. In *Proc. KDD'04*, pages 296–305, Aug. 2004.

UCI: machine learning repository. http://www.ics.uci.edu/~ mlearn/MLSummary.html, 2007.

V. Verykios, E. Bertino, I. Fovino, L. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *SIGMOD Record*, 3:50–57, 2004.

K. Wang, C. Xu, and B. Liu. Clustering transactions using large items. In *Proc. of CIKM*, pages 483–490, 1999.

X. Yan, M. R. Mehan, Y. Huang, M. S. Waterman, P. S. Yu, and X. J. Zhou. A graph-based approach to systematically reconstruct human transcriptional regulatory modules. In *Proc. of ISMB*, 2007.

X. Yan, P. S. Yu, and J. Han. Graph Indexing: A frequent structure-based approach. In *Proc. of SIGMOD*, pages 335–346, 2004.

C. Yang, U. Fayyad, and P. S. Bradley. Efficient discovery of error-tolerant frequent itemsets in high dimensions. In *Proc. KDD'01*, pages 194–203, Aug. 2001.

M. J. Zaki. Scalable algorithms for association mining. *IEEE Trans. Knowledge and Data Engineering*, 12:372–390, 2000.

M. J. Zaki and C. J. Hsiao. CHARM: An efficient algorithm for closed itemset mining. In *Proc. SDM'02*, pages 457–473, April 2002.

F. Zhu, X. Yan, J. Han, P. S. Yu, and H. Cheng. Mining colossal frequent patterns by core pattern fusion. In *Proc. 2007 Int. Conf. Data Engineering (ICDE'07)*, Istanbul, Turkey, April 2007.

The Impact of Overfitting and Overgeneralization on the Classification Accuracy in Data Mining

Huy Nguyen Anh Pham[1] and Evangelos Triantaphyllou[1]

Department of Computer Science, 298 Coates Hall, Louisiana State University, Baton Rouge, LA 70803 hpham15,trianta@lsu.edu

Summary. Many classification studies often times conclude with a summary table which presents performance results of applying various data mining approaches on different datasets. No single method outperforms all methods all the time. Furthermore, the performance of a classification method in terms of its false-positive and false-negative rates may be totally unpredictable. Attempts to minimize any of the previous two rates, may lead to an increase on the other rate. If the model allows for new data to be deemed as unclassifiable when there is not adequate information to classify them, then it is possible for the previous two error rates to be very low but, at the same time, the rate of having unclassifiable new examples to be very high. The root to the above critical problem is the overfitting and overgeneralization behaviors of a given classification approach when it is processing a particular dataset. Although the above situation is of fundamental importance to data mining, it has not been studied from a comprehensive point of view. Thus, this chapter analyzes the above issues in depth. It also proposes a new approach called the Homogeneity-Based Algorithm (or HBA) for optimally controlling the previous three error rates. This is done by first formulating an optimization problem. The key development in this chapter is based on a special way for analyzing the space of the training data and then partitioning it according to the data density of different regions of this space. Next, the classification task is pursued based on the previous partitioning of the training space. In this way, the previous three error rates can be controlled in a comprehensive manner. Some preliminary computational results seem to indicate that the proposed approach has a significant potential to fill in a critical gap in current data mining methodologies.

Key words: classification, prediction, overfitting, overgeneralization, false-positive, false-negative, homogenous set, homogeneity degree, optimization

1 Introduction

The importance of collecting enormous amounts of data related to science, engineering, business, governance, and almost any endeavor of human activity or the natural world is well recognized today. Powerful mechanisms for collecting and storing data and managing them in large datasets are in place in many large and mid-range companies, not to mention research labs and various agencies. There is, however, a serious challenge in making good use of such massive datasets and trying to learn new knowledge of the system or phenomenon that created these data. Human analysts cannot process and comprehend such datasets unless they have special computational tools at their disposal.

The emerging field of data mining and knowledge discovery seeks to develop reliable and effective computational tools for analyzing large datasets for the purpose of extracting new knowledge from the data. Such new knowledge can be derived in the form of patterns that are embedded in the data.

Many applications of data mining involve the analysis of data that describe the state of nature of a hidden system of interest to the analyst. Such a system could be a natural or artificial phenomenon (such as the state of the weather or the result of a scientific experiment), a mechanical system (such as the engine of a car), an electronic system (such as an electronic device), and so on. Each data point describes the state of the phenomenon or system in terms of *a number of attributes* and *their values* for a given realization of the phenomenon or system. Furthermore, each data point is associated with *a class value* which describes a particular state of nature of this phenomenon or system.

For instance, a bank administrator could be interested in knowing whether a loan application should be approved or not based on some characteristics of applicants for credit. Here the two classes are: “approve” or “do not approve”. Attributes in this hypothetical scenario could be the age of the applicant, the income level of the applicant, the education level, whether he/she has a permanent job, etc. Then, the goal of the data mining process might be to extract any patterns that might be present in the data of successful credit applicants and also patterns that might be present in the data of non-successful applicants. By “successful applicants” we mean here those who can repay their loans without any negative complications, while with “non-successful applicants” we mean those who default their loans.

There could be many questions to be asked, but only a few of them would be important for the decision. With the abundance of the data available in this area, a careful analysis could provide a *pattern* that exposes the main characteristics of reliable loan applicants. Then, the data mining analyst would like to identify such patterns from past data for which we know the final outcome and use those patterns to decide whether a new application for credit should be approved or not.

In other words, many applications of data mining involve the analysis of data for which we know the class value of each data point. We wish to infer some patterns from these data which in turn could be used to infer the class value of new points for which we do not know their class value. These patterns may be defined on the attributes used to describe the available data (also known as the *training data*). For instance, for the previous bank example the patterns may be defined on the level of education, years on the same job, level of income, of the applicants.

This kind of data mining analysis is called *classification* or *class prediction* of new data points because it uses patterns inferred from training data to aid in the correct classification/class prediction of new data points for which we do not know their class value. We only know the values of the attributes (perhaps not all of them) of the new data points. This description implies that this type of data mining analysis, besides the typical data definition, data collection, and data cleaning steps, involves the inference of a model of the phenomenon or system of interest to the analyst. This model is the patterns mentioned above. The data involved in deriving this model are the training data. Next, this model is used to infer the class value of new points.

There have been many theoretical and practical developments in the last two decades in this field. Most recent methods include the Statistical Learning Theory (Vapnik, 1998), Artificial Neural Networks (ANNs) (Hecht-Nielsen, 1989) and (Abdi, 2003), Decision Trees (DTs) (Quinlan, 1993), logic-based methods (Hammer and Boros, 1994), (Triantaphyllou, 1994; and 2007), and Support Vector Machines (SVMs) (Vapnik, 1979; and 1998) and (Cristianini and John, 2003).

In many real-life or experimental studies of data mining, some classification approaches work better with some datasets, while they work poorly with other datasets for no apparent reason. For instance, DTs had some success in the medical domain (Zavrsnik *et. al.*, 1995). However, they also have had some certain limitations when they were used in this domain, as described for instance in (Kokol *et. al.*, 1998) and (Podgorelec, 2002). Furthermore, the success of SVMs has been shown in bioinformatics, such as in (Byvatov, 2003) and (Huzefa *et. al.*, 2005). At the same time, SVMs also did poorly in this field (Spizer *et. al.*, 2006). If the data mining approach is accurate, then the people praise the mathematical model and claim that it is a good model. However, there is no good understanding of why such models are accurate or not. Their performance is often times coincidental.

A growing belief is that overfitting and overgeneralization problems may cause poor performance of the classification/class prediction model. Overfitting means that the extracted model describes the behavior of known data very well but does poorly on new data points. Overgeneralization occurs when the system uses the available data and then attempts to analyze vast amounts of data that has not seen yet.

Assume that there are two classes which, arbitrarily, we will call the positive and the negative class. Then, one may infer the following two classification models. The first model (which we will call the "positive" model) describes patterns embedded in the positive examples and which do not exist in the negative examples. In a similar manner, we define the "negative" model. For instance, when developing a diagnosis system for some types of cancer one may want to derive two models that can classify a new patient to either, positive (which means has cancer) or negative (which means does not have cancer) cases.

The analyst may want one of the previous two models to be more "conservative" while the other model to be more "liberal". If both models are ultra "conservative" then the implication is that they would only classify new cases that are very closely related to cases they already have seen in the training data. In this situation, the net effect would be many cases to be left as *unclassifiable* by both systems. Similarly, if both systems are classifying new data in a "liberal" manner, then they may contradict each other too often when they are presented with new cases. Again, this situation might be undesirable. Thus, a "liberal" behavior by a classification model means that the model has a tendency for overgeneralization. A similar relationship exists between the concept of "conservative" and overfitting

This chapter aims at finding a way to balance both fitting and generalization in order to minimize the total misclassification cost of the final system. By doing so, it is hoped that the classification/prediction accuracy of the inferred models will be very high or at least as high as it can be achieved with the available training data. We plan to achieve this by balancing the previous two conflicting behaviors of the extracted systems.

The next section provides a preliminary description of the main research problem. The third section gives a summary of the main developments in the related literature. The proposed methodology is highlighted in the fourth section. That section shows how a balance between fitting and generalization has the potential to improve many existing classification algorithms. The fifth section discusses some promising preliminary results. These pilot results give an early indication of how this methodology may improve the performance of existing classification algorithms. Finally, this chapter ends with some conclusions.

2 Formal Problem Description

2.1 Some Basic Definitions

In order to help fix ideas, we first consider the hypothetical sample data depicted in Figure 1. Let us assume that the "circles" and "squares" in this figure correspond to sampled observations from two classes defined in 2-D.

In general, a *data point* is a vector defined on n variables along with their values. In the above figure, n is equal to 2 and the two variables are indicated by the X and Y axis. Not all values may be known for a given data point.

Data points describe the behavior of the system of interest to the analyst. For instance, in the earlier bank application a given data point may describe the level of education, years on the same job, level of income of a particular applicant, etc. The variables may be continuous, binary, or categorical, etc. All data are assumed to be deterministic and numeric at this point. The *state space* is the universe of all possible data points. In terms of Figure 1, the state space is any point in the X-Y plane.

We assume that there are only two classes. Arbitrarily, we will call one of them the *positive* class while the other the *negative* class. Thus, a *positive data point*, also known as a *positive example*, is a data point that has been evaluated to belong to the positive class. A similar definition exists for *negative data points* or *negative examples.*

Given a set of positive and negative examples, such as the ones depicted in Figure 1, this set is called the *training data* (examples) or the *classified examples.* The remaining of the data from the state space is called the *unclassified data* (examples).

2.2 Problem Description

We start the problem description with a simple analysis on the sample data depicted in Figure 1. Suppose that a data mining approach (such as a DT, ANN, or SVM) has been applied on these data. Next we assume that two classification systems have been inferred from these data. Usually, such classification systems arrange the training data into groups described by the parts of a decision tree or classification rules. In a way, these groups of training data define the patterns inferred from the data after the application of a data mining algorithm. For this hypothetical scenario, we assume that the data mining algorithm has inferred the system patterns depicted in Figure 2.(a).

In general, one classification system describes the positive data (and thus we will call it the *positive system*) while the other system describes the negative data (and thus we will call it the *negative system*). In Figure 2.(a) the positive system corresponds to sets A and B (which define the positive pattern) while sets C and D correspond to the negative system (and thus they define the negative pattern).

In many real-life applications, there are two different penalty costs if one erroneously classifies a true positive point as negative or if one classifies a true negative point as positive. The first case is known as false-positive, while the second case is known as false-negative. Furthermore, a closer examination of Figure 2.(a) indicates that there are some unclassifiable points which either are not covered by any of the patterns or are covered by patterns that belong to both classes. For instance, point N (indicated as a triangle) is not covered by any of the patterns, while point M (also a triangle) is covered by sets A and C which belong to the positive and the negative patterns, respectively.

For the first case, as point N is not covered by any of the patterns, the inferred system may declare it as an *unclassifiable* point. In the second case,

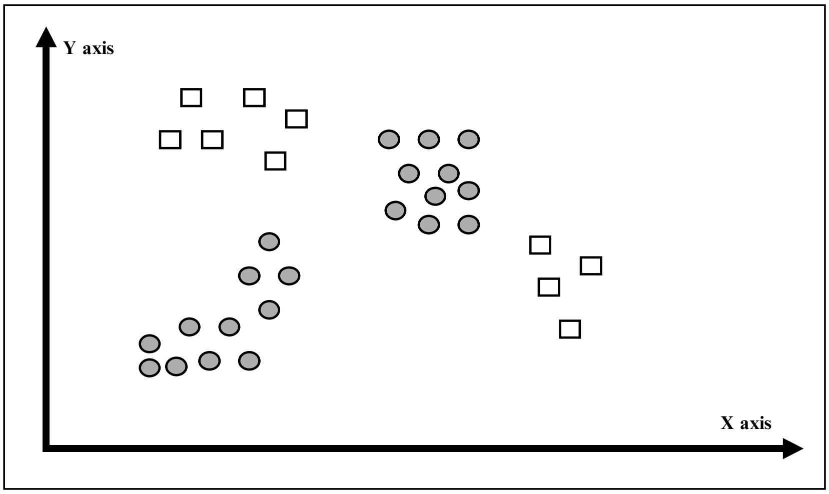


Fig. 1. Sample data from two classes in 2-D.

there is a direct disagreement by the inferred system as the new point (i.e., point M) is covered simultaneously by patterns of both classes. Again, such a point may be also declared as *unclassifiable*. Thus, in many real-life applications of data mining one may have to consider three different penalty costs as follows: one cost for the false-positive case, one cost for the false-negative case, and one cost for the unclassifiable case.

Next consider Figure 2.(b). Suppose that all patterns A, B, C and D have been reduced significantly but still cover the original training data. A closer examination of this figure indicates that now both points M and N are not covered by any of the inferred patterns. In other words, these points and several additional points which were classified before by the inferred systems now become unclassifiable.

Furthermore, the data points which before were simultaneously covered by patterns from both classes, and thus were unclassifiable, are now covered by only one type of pattern or none at all. Thus, it is very likely that the situation depicted in Figure 2.(b) may have a higher total penalty cost than the original situation depicted in Figure 2.(a). If one takes this idea of reducing the covering sets as much as possible to the extreme, then there would be one pattern (i.e., just a small circle) around each individual training data point. In this extreme case, the total penalty cost due to unclassifiable points would be maximum as the system would be able to classify the training data only and nothing else. The previous scenarios are known as *overfitting* of the training data.

On the other hand, suppose that the original patterns depicted as sets A, B, C and D (as shown in Figure 2.(a)) are now expanded significantly as in Figure 2.(c). A closer examination of this figure demonstrates that points M and N are now covered simultaneously by patterns of both classes. Also, more points are now covered simultaneously by patterns of both classes. Thus, under this scenario we also have lots of unclassifiable points because this scenario creates lots of cases of disagreement between the two classification systems (i.e., the positive and the negative systems). This realization means that the total penalty cost due to unclassifiable points will also be significantly higher than under the scenario depicted in Figure 2.(a). This scenario is known as *overgeneralization* of the training data.

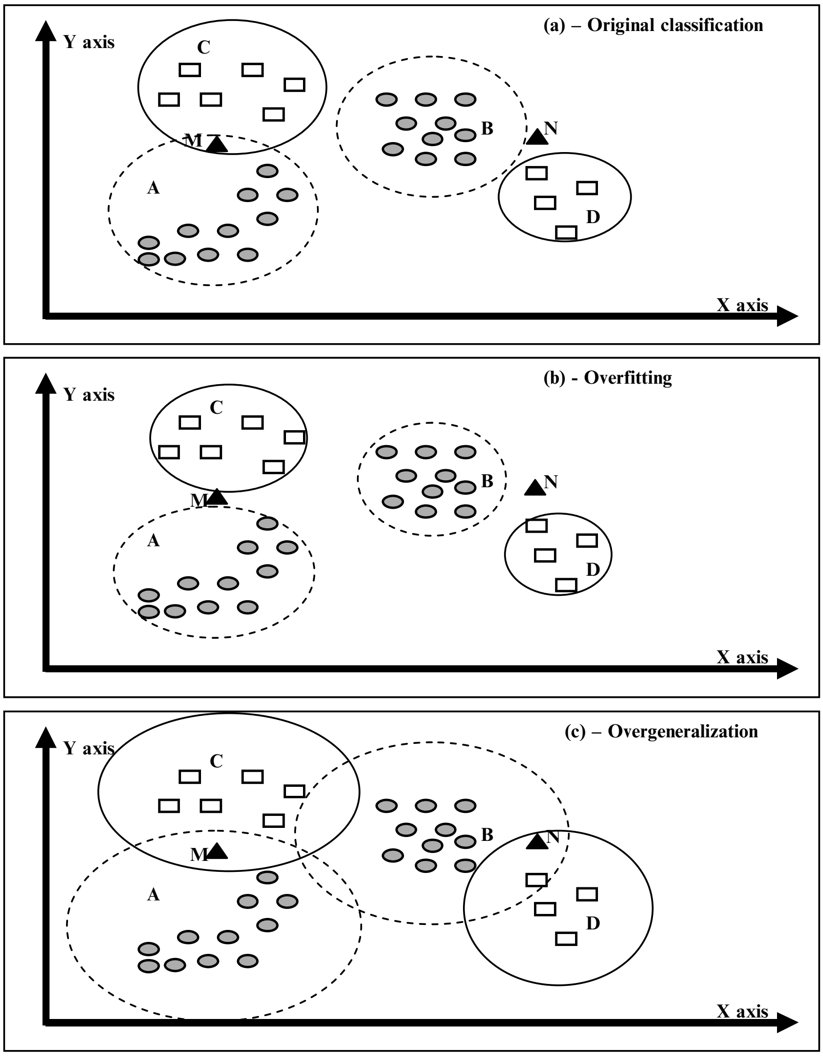


Fig. 2. An illustrative example of overfitting and overgeneralization.

Thus, we cannot separate the control of fitting and generalization into two independent studies. That is, we need to find a way to simultaneously balance fitting and generalization by adjusting the inferred systems (i.e., the positive and the negative systems) obtained from a classification algorithm. The balance of the two systems will target at minimizing the total misclassification costs of the final system.

In particular, let us denote C_{FP}, C_{FN}, and C_{UC} as the penalty costs for the false-positive, the false-negative, and the unclassifiable cases, respectively. Let *RATE_FP*, *RATE_FN*, and *RATE_UC* be the false-positive, the false-negative, and the unclassifiable rates, respectively. Then, the problem is to achieve a balance between fitting and generalization that would minimize, or at least significantly reduce, the total misclassification cost denoted as *TC*. The problem is defined in the following expression:

$$TC = \min \left(C_{FP} \times RATE\_FP + C_{FN} \times RATE\_FN + C_{UC} \times RATE\_UC \right) \quad (1)$$

This methodology may assist the data mining analyst to create classification systems that would be optimal in the sense that their total misclassification cost would be minimized.

In terms of Figures 2.(a), (b) and (c), let us now consider the situation depicted in Figure 3. At this point assume that in reality point M is negative while point N is positive. Figure 3 shows different levels of fitting and generalization for the two classification systems. For the sake of illustration, sets C and D are kept the same as in the original situation (i.e., as depicted in Figure 2.(a)) while set A has been reduced (i.e., it fits the training data more closely) and now it does not cover point M. On the other hand, set B is expanded (i.e., it generalizes the training data more) to cover point N. This new situation may correspond to a total misclassification cost that is smaller than those by any of the previous three scenarios. The following section will give a summary of the main developments in the related literature.

3 Literature Review

Most of the classification algorithms have focused on the minimization of the classification error of the training points. In this way, it is expected that new points will be classified with higher prediction accuracy. This section is a summary of the literature about ways that classification algorithms deal with the overfitting and the overgeneralization problems.

3.1 Decision Trees (DTs)

There are two methods for controlling the overfitting problem in DTs: pre-pruning methods in which the growing tree approach is halted by some early stopping rules before generating a fully grown tree, and post-pruning in which the DT is first grown to its maximum size and then we trim some partitions of the tree.

There was recently a lot of effort which has focused on improving the pre-pruning methods. (Kohavi, 1996) proposed the NBTree (a hybrid of decision-tree and naive- classifiers). The NBTree provides some early stopping rules by comparing two alternatives: partitioning the instance-space further on (i.e., continue splitting the tree based on some gain ratio stopping criteria) versus stopping the partition and producing a single Naïve Bayes classifier. (Zhou and Chen, 2002) suggested the hybrid DT approach for growing a binary DT. A feed-forward neural network is used to subsequently determine some early stopping rules. (Rokach *et. al.*, 2005) proposed the cluster-based concurrent decomposition (CBCD) algorithm. That algorithm first decomposes the training set into mutually exclusive sub-samples and then uses a voting scheme to combine these sub-samples for the classifier's predictions. Similarly, (Cohen *et. al.*, 2007) proposed an approach for building a DT by using a homogeneity criterion for splitting the space. However, the above approaches have a difficulty in choosing the threshold value for early termination. A value which is too high may result in underfitting models, while a too low threshold value may not be sufficient to overcome overfitting models.

Under the post-pruning approaches described in (Breiman *et. al.*, 1984) and (Quinlan, 1987), the pruning process eliminates some partitions of the tree. The reduction on the number of partitions makes the remaining tree more general. In order to help fix the main idea, we consider the simple example depicted in Figure 4. Suppose that Figure 4.(a) shows a DT inferred from some training examples. The pruning process eliminates some of the DT's nodes as depicted in Figure 4.(b). The remaining part of the DT, as shown in Figure 4.(c), implies some rules which are more general. For instance, the left most branch of the DT in Figure 4.(a) implies the rule "*if* $D \wedge A \wedge B \wedge C$*, then* ..." On the order hand, Figure 4.(c) implies the more general rule "*if* $D \wedge A$*, then* ..."

However, more generalization is not always required nor is it always beneficial. A more complex arrangement of partitions has been proved to increase the complexity of DTs in some applications. Furthermore, the treatment of generalization of a DT may lead to overgeneralization since pruning conditions are based on localized information.

Instead of the pruning methods, there have been some other developments to improve the accuracy of DTs. (Webb, 1996) attempted to graft additional leaves to a DT after its induction. This method does not leave any area of the instance space in conflict, because each data point belongs to only one class. Obviously, the overfitting problem may arise from this approach. (Mansour *et. al.*, 2000) proposed another way to deal with the overfitting problem by using the learning theoretical method. In that method, the bounds on the error rate for DTs depend both on the structure of the tree and on the specific sample. (Kwok and Carter, 1990), (Schapire, 1990), (Wolpert, 1992), (Dietterich and

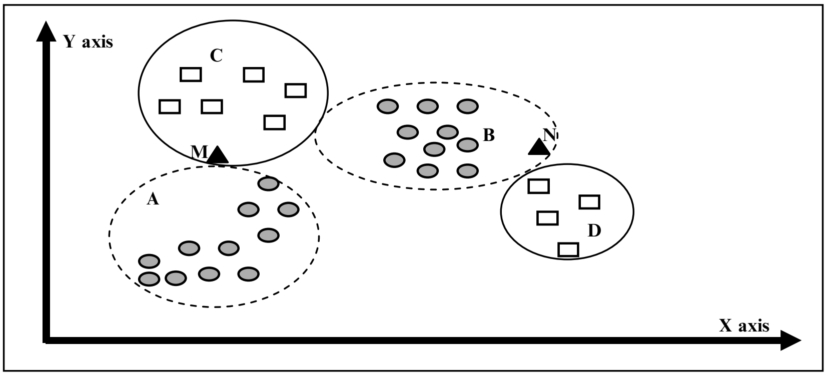


Fig. 3. An illustrative example of a better classification.

Bakiri, 1994), (Ali *et. al.*, 1994), (Oliver and Hand, 1995), (Nock and Gascuel, 1995) and (Breiman, 1996) allowed multiple classifiers used in a conjunction. This method is similar to using a Disjunctive Normal Form (DNF) Boolean function. Furthermore, (Breiman, 2001) also used the so-called random forest approach for multiple classifiers. However, the above approaches might create conflicts between the individual classifiers' partitions, as in the situation presented in C4.5 (Quinlan, 1993).

3.2 Rule-Based Classifiers

A rule-based classifier uses a collection of "if ... then ..." rules that identify key relationships between the attributes and the class values of a dataset. There are two methods which infer classification rules: direct methods which infer classification rules directly from the data, and indirect methods which infer classification rules from other classification methods such as DTs, SVMs, or ANNs and then they translate the final model into a set of classification rules (Tan *et. al.*, 2005). An extensive survey of rule-based methods can be found in (Triantaphyllou and Felici, 2006). A new rule-based approach, which is based on mathematical logic, is described in (Triantaphyllou, 2007).

A well-known algorithm of direct methods is the Sequence Covering algorithm and its later enhancement, the CN2 algorithm (Clark and Niblett, 1989). To control the balance of fitting and generalization while generating rules, these algorithms first use two strategies for growing the classification rules: general-to-specific or specific-to-general. Then, the rules are refined by using the pre and post-pruning methods mentioned in DTs.

Under the general-to-specific strategy, a rule is created by finding all possible candidates and use a greedy approach to choose the new conjuncts to be added into the rule antecedent part in order to improve its quality. This approach ends when some stopping criteria are met.

Under the specific-to-general strategy, a classification rule is initialized by randomly choosing one of the positive points as the initial step. Then, the rule is refined by removing one of its conjuncts so that this rule can cover more positive points. This refining approach ends when certain stopping criteria are met. A similar way exists for the negative points.

There are some related developments regarding these strategies. Such developments include a beam search approach (Clark and Boswell, 1991) which avoids the overgrowing of rules as result of the greedy behavior, the RIPPER algorithm (Cohen, 1995) which uses a rule induction algorithm. However, the use of the two strategies for growing classification rules has their drawbacks. The complexity for finding optimal rules is of exponential size of the search space. Although some rule pruning methods are used to improve their generalization error, they also leave drawbacks as mentioned in the case of DTs.

3.3 K-Nearest Neighbor Classifiers

While DTs and rule-based classifiers are examples of eager learners, K-Nearest Neighbor Classifiers (Cover, Hart, 1967) and (Dasarathy, 1979) are known as lazy learners. That is, this approach finds K training points that are relatively similar to attributes of a testing point to determine its class value.

The importance of choosing the right value for K directly affects the accuracy of this approach. A wrong value for K may lead to the overfitting or the overgeneralization problems (Tan *et. al.*, 2005). One way to reduce the impact of K is to weight the influence of the nearest neighbors according to their distance to the testing point. One of the most well-known schemes is the distance-weighted voting scheme (Dudani, 1976) and (Keller, Gray and Givens, 1985).

However, the use of K-nearest neighbor classifiers has their drawbacks. Classifying a test example can be quite expensive since we need to compute a similarity degree between the test point and each training point. They are unstable since they are based on localized information only. Finally, it is difficult to find an appropriate value for K to avoid model overfitting or overgeneralization.

3.4 Bayes Classifiers

This approach uses the modeling probabilistic relationships between the attribute set and the class variable for solving classification problems. There are two well known implementations of Bayesian classifiers: Naïve Bayes (NBs) and Bayesian Belief Networks (BBNs).

NBs assume that all the attributes are independent of each other and then they estimate by using the class conditional probability. This independence assumption, however, is obviously problematic because often times in many

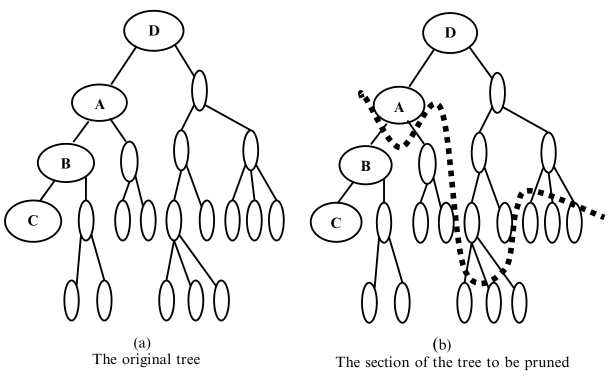


Fig. 4. An illustrative example of the DT pruning.

real applications there are strong conditional dependencies between the attributes. Furthermore, when using the independence assumption, NBs may suffer of overfitting since they are based on localized information.

Instead of requiring all attributes to be conditionally independent given a class, a BBN (Duda and Hart, 1973) allows only for pairs of attributes to be conditionally independent. We introduce this approach by discussing an illustrative example. Suppose that we have a training dataset consisting of the attributes: age, occupation, income, buy (i.e., buy some product X), and interest (i.e., "interest in purchasing insurance for this product"). The attributes age, occupation and income may determine if a customer will buy some product X. Given is a customer who has bought product X. There is an interest in buying insurance when we assume this is independent of age, occupation, and income. These constraints are presented by the BBN depicted in Figure 5. Thus, for a certain data point described by a 5-tuple (age, occupation, income, buy, interest), its probability based on the BBN should be:

P(age, occupation, income, buy, interest) =

$$\text{P(age)} \times \text{P(occupation)} \times \text{P(income)} \times \text{P(buy} \mid \text{age, occupation, income)} \times \text{P(interest} \mid \text{buy)}.$$

There was a lot of effort which has focused on improving BBNs. This effort follows two general approaches: selecting a feature subset (Langley and Sage, 1994), (Pazzani, 1995), and (Kohavi and John, 1997) and relaxing the independence assumptions (Kononenko, 1991) and (Friedman *et. al.*, 1997). However, these developments have the following drawbacks:

They require a large amount of effort when constructing the network.

They quietly degrade to overfitting because they combine probabilistically the data with prior knowledge.

3.5 Artificial Neural Networks (ANNs)

Recall that an ANN is a model that is an assembly of inter-connected nodes and weighted links. The output node sums up each of its input values according to the weights of its links. The output node is compared against a threshold value t. Such a model is illustrated in Figure 6. The ANN in this figure consists of the three input nodes X_1, X_2, and X_3 which correspond to the weighted links w_1, w_2, and w_3, respectively, and one output node Y. The sum of the input nodes can be $Y = \text{sign}\sum_i (X_i w_i - t)$, called the perceptron model (Abdi, 2003).

In general, an ANN has a set of input nodes $X_1, X_2, \ldots, X_m$ and one output node Y. Given are n values for the m-tuple (X_1, X_2, ..., X_m). Let $\overset{\wedge}{Y_1}$, $\overset{\wedge}{Y_2}$, ..., $\overset{\wedge}{Y_n}$ be the predicted outputs and $Y_1, Y_2, \ldots, Y_n$ be the expected outputs from the n values, respectively. Let $E = \sum_{i=1}^{n} [Y_i - \overset{\wedge}{Y_i}]^2$denote the total sum of the squared differences between the expected and the predicted outputs. The goal of the ANN is to determine a set of the weights in order to minimize the value of E. During the training phase of an ANN, the weight parameters are adjusted until the outputs of the perceptron become consistent with the true outputs of the training points. In the weight update process, the weights should not be changed too drastically because E is computed only for the current training point. Otherwise, the adjustments made during earlier iterations may be undone.

In order to avoid overgeneralization or overfitting, the design for an ANN must be considered. A network that is not sufficiently complex may fail to fully detect the input in a complicated dataset, leading to overgeneralization. On the other hand, a network that is too complex may not only fit the input but also the noisy points, thus leading to overfitting. According to (Geman, Bienenstock, and Doursat, 1992) and (Smith, 1996), the complexity of a network is related both to the number of the weights and to the size of the weights. Geman and Smith were directly or indirectly concerned with the number and size of the weights. That is, the number of the weights relates to the number of hidden units and layers. The more weights there are, relative to the number of the training cases, the more overfitting amplifies noise in the classification systems (Moody, 1992). Reducing the size of the weights may reduce the effective number of the weights leading to weight decay (Moody, 1992) and early stopping (Weigend, 1994). In summary, ANNs have the following drawbacks:

It is difficult to find an appropriate network topology for a given problem in order to avoid model overfitting and overgeneralization.

It takes lots of time to train an ANN when the number of hidden nodes is large.

3.6 Support Vector Machines (SVMs)

Another classification technique that has received considerable attention is known as SVMs (Vapnik, 1995). The basic idea behind SVMs is to find a maximal margin hyperplane, θ, that will separate points considered as vectors in an m-dimensional space. The maximum margin hyperplane can be essentially represented as a linear combination of the training points. Consequently, the decision function for classifying new data points with respect to the hyperplane only involves dot products between data points and the hyperplane.

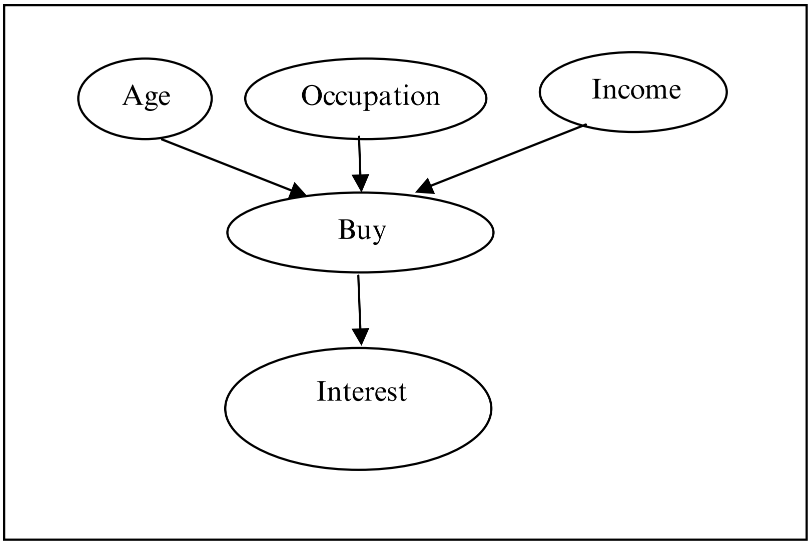


Fig. 5. An illustrative example of a BBN (Rada, 2004).

In order to help fix ideas, we consider the simple illustrative example depicted in Figure 7. Suppose that we have a training dataset defined on two given classes (represented by the squares and circles) in 2-D. In general, the approach can find many hyperplanes, such as B_1 or B_2, separating the training dataset into the two classes. The SVM, however, chooses B_1 to classify this training dataset since B_1 has the maximum margin. Roughly speaking it is in the middle of the distance between the two groups of training examples.

Decision boundaries with maximal margins tend to lead to better generalization. Furthermore, SVMs attempt to formulate the learning problem as a convex optimization problem in which efficient algorithms are available to find a global solution. For many datasets, however, an SVM may not be able to formulate the learning problem as a convex optimization problem because

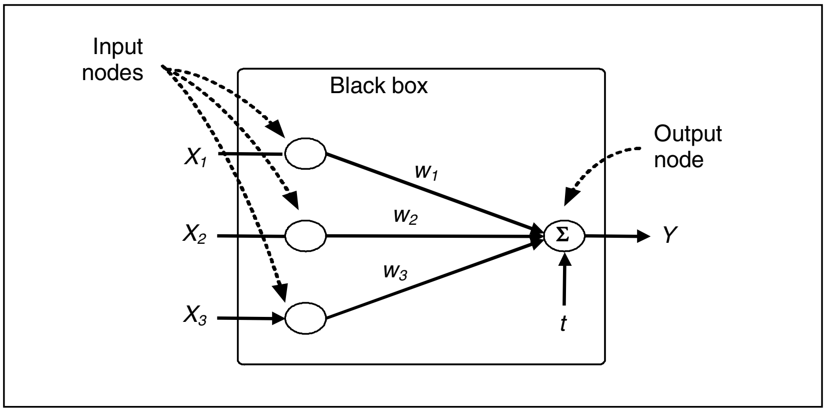


Fig. 6. An illustrative example of an ANN (Tan et. al., 2005).

it may be the cause of too many misclassifications. Thus, the attempts for formulating the learning problem may lead to overgeneralization.

4 Proposed Methodology – The Homogeneity-Based Algorithm (HBA)

4.1 Some Key Observations

In order to help motivate the proposed methodology, we first consider the situation depicted in Figure 8.(a). This figure presents two inferred patterns. These are the circular areas that surround groups of training data (shown as small circles). Actually, these data are part of the training data shown earlier in Figure 1 (please recall that the circles in Figure 1 represent positive points). Moreover, in Figure 8.(a) there are two additional data points shown as small triangles and are denoted as points P and Q. At this situation, it is assumed that we do not know the actual class values of these two new points. We would like to use the available training data and inferred patterns to classify these two points. Because points P and Q are covered by patterns A and B, respectively, both of these points may be assumed to be positive examples.

Let us look more closely at pattern A. This pattern covers regions of the state space that are not adequately populated by positive training points. Such regions, for instance, exist in the upper left corner and the lower part of pattern A (see also Figure 8.(a)). It is possible that the unclassified points which belong to such regions are erroneously assumed to be of the same class as the positive training points covered by pattern A. Point P is in one of these sparely covered regions under pattern A. Thus, the assumption that point P is a positive point may not be very accurate.

On the other hand, pattern B does not have such sparely covered regions (see also Figure 8.(a)). Thus, it may be more likely that the unclassified points covered by pattern B are more accurately assumed to be of the same class as the positive training points covered by the same pattern. For instance, the assumption that point Q is a positive point may be more accurate.

The above simple observations lead one to surmise that the accuracy of the inferred systems can be increased if the derived patterns are, somehow, more compact and homogenous.

According to the Wikipedia Dictionary (2007), given a certain class (i.e., positive or negative), a homogenous set describes a steady or uniform distribution of a set of distinct points. That is, within the pattern there are no regions (also known as *bins*) with unequal concentrations of classifiable (i.e., either positive or negative) and unclassified points. In other words, if a pattern is partitioned into smaller bins of the same unit size and the density of these bins is almost equal to each other (or, equivalently, the standard deviation is small enough), then this pattern is a homogenous set. An axiom and a theorem are derived from the definition of a homogenous set as follows:

Axiom 1. Given is an inferred pattern C of size one. Then, C is a homogenous set.

This axiom is used later in Section 4.4.

Theorem 1. *Let us consider a homogenous set C. If C is divided into two parts, C_1 and C_2, then the two parts are also homogenous sets.*

Proof. We prove Theorem 1 by using contradiction. Since C is a homogenous set, there is a uniform random variable Z that represents the distribution of points in C. Similarly, Z_1 and Z_2 are the two random variables that represent the distribution of points in C_1 andC_2, respectively. Obviously, Z is the sum of Z_1 and Z_2. Assume that either Z_1 or Z_2 is a non homogenous set. Thus, Z_1 + Z_2 is not a uniform random variable. This contradicts the fact that Z is a uniform random variable.

The pattern which is represented by the non homogenous A can be replaced by two more homogenous sets denoted as A_1 and A_2 as in Figure 8.(b). Now the regions covered by the two new smaller patterns A_1 and A_2 are more homogenous than the area covered by the original pattern A. Given these considerations, point P may be assumed to be an unclassifiable point while point Q is still a positive point.

As presented in the previous paragraphs, the homogenous property of patterns may influence the number of misclassification cases of the inferred classification systems. Furthermore, if a pattern is a homogenous set, then the number of training points covered by this pattern may be another factor which affects the accuracy of the overall inferred systems. For instance, Figure 9 shows the case discussed in Figure 8.(b) (i.e., pattern A has been replaced by

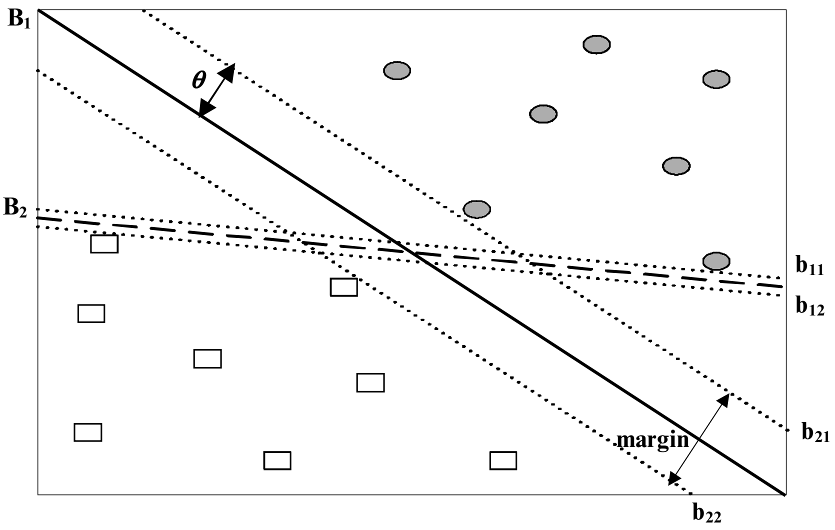


Fig. 7. An illustrative example of an SVM (Tan et. al., 2005).

two more homogenous sets denoted as A_1 and A_2). Suppose that all patterns A_1, A_2 and B are homogenous sets and a new point S (indicated as a triangle) is covered by pattern A_1.

A closer examination of this figure shows that the number of points in B is higher than those in A_1. Although both points Q and S are covered by homogenous sets, the assumption that point Q is a positive point may be more accurate than the assumption that point S is a positive point. The above simple observation leads one to surmise that the accuracy of the inferred systems may also be affected by a *density* measure. Such a density could be defined as the number of points in each inferred pattern per unit of area or volume. Therefore, this density will be called the *homogeneity degree.*

In summary, a fundamental assumption here is as follows: if an unclassified point is covered by a pattern that is a homogenous set which also happens to have a high homogeneity degree, then it may be more accurately assumed to be of the same class as the points covered by that pattern. On the other hand, the accuracy of the inferred systems may be increased when their patterns are more homogenous and have high homogeneity degrees.

4.2 Non Parametric Density Estimation

Please recall that a pattern C of size n is a homogenous set if the pattern can be partitioned into smaller bins of the same unit size h and the density of these bins is almost equal to each other (or, equivalently, the standard deviation is small enough). In other words, if C is superimposed by a hypergrid of unit size h and the density of the bins inside C is almost equal to each other, then C is a homogenous set.

As seen in the above, the density estimation of a typical bin plays an important role in determining whether a set is a homogenous set or not. According to (Duda *et. al.*, 2001), the density estimation is the construction of an estimate, based on the observed data and on an unobservable underlying probability density function. There are two basic approaches to the density estimation:

Parametric in which we assume a given form of the density function (i.e., Gaussian, normal, and so on) and its parameters (i.e., its mean and variance) such that this function may optimally fit the model to the dataset.

Non parametric where we cannot assume a functional form for the density function, and the density estimates are driven entirely by the available training data.

The following sections will use the non parametric density estimation. That is, the approach divides pattern C into a number of small bins of unit size h. The density at the center $x$of each bin can be approximated by the fraction of points in C that fall into the corresponding bin and the volume of the bin. For instance, a bin in 3-D can be a cube of unit size h as depicted in Figure 10. Let n be the number of points in C and d(x) denote the x's density, then:

$$d(x) = \frac{1}{n}[\frac{the\, number\ of\ examples\ falling\ in\ the\ bin\ with\ center\ x}{volume\ of\ the\ bin}]. \quad (2)$$

The basic idea behind computing $d(x)$ relies on the probability p that a data point x, drawn from a distribution function, will fall in bin R.By using this idea we arrive at the following obvious estimate for $d(x)$:

$$d(x) \approx \frac{k}{n \times V}, \quad (3)$$

where x is a point within $R; k$ is the number of points which fall in R; and V is the volume enclosed by R.

The Parzen Windows approach (Duda and Hart, 1973) was introduced as the most appropriate approach for the density estimation. That is, it temporarily assumes that the region R is a D-dimensional hypercube of unit size h. To find the number of points that fall within this region, the Parzen Windows approach defines a kernel function $\varphi(u)$ as follows:

$$\varphi(u) = \begin{cases} 1, & |u| \le 1/2. \\ 0, & otherwise. \end{cases} \quad (4)$$

It follows that the quantity $\varphi(\frac{x-x_i}{h})$ is equal to unity if the point x_i is inside the hypercube of unit size h and centered at x, and zero otherwise. Therefore, k, the number of points in the hypercube is given by:

$$k = \sum_{i=1}^{n} \varphi(\frac{x - x_i}{h}) \tag{5}$$

In the D-dimensional space, the kernel function can be presented as follows:

$$\varphi(\frac{x - x_i}{h}) = \prod_{m=1}^{D} \varphi(\frac{x^m - x_i^m}{h}). \tag{6}$$

By using (6) in Equation (3), one gets:

$$d(x) \approx \frac{1}{n \times h^D} \sum_{i=1}^{n} \prod_{m=1}^{D} \varphi(\frac{x^m - x_i^m}{h}). \tag{7}$$

Usually, but not always, $\varphi(u)$ will be radically symmetric. Thus, the unimodal probability density function, for instance the multivariate Gaussian density function, may be used to compute $\varphi(u)$:

$$\varphi(u) = \frac{1}{(2 \times \pi)^{\frac{D}{2}}} \exp(-\frac{1}{2} u^t u). \tag{8}$$

Choosing a value for h plays the role of a smoothing parameter in the density estimation. That is, if $h \to \infty$, then the density at point x in C, $d(x)$, approaches a false density. As $h \to 0$, then the kernel function approaches the Dirac Delta Function and $d(x)$ approaches to the true density (Bracewell, 1999).

Suppose that we determine all distances between all possible pairs formed by taking any two points from pattern C. For easy illustration, assume that for pattern C which contains 5 points these distances are as follows: 6, 1, 2, 2, 1, 5, 2, 3, 5, 5. Then, we define S as a set of the distances which have the highest frequency. For the previous illustrative example, we have set S equal to $\{2, 5\}$ as both distances 2 and 5 occur with frequency equal to 3. By using the concept of the previous set S, Heuristic Rule 1 proposes a way for finding an appropriate value for h when we estimate the density $d(x)$. In particular, it uses the minimum value in S (which is equal to 2 in this illustration) as follows:

Heuristic Rule 1: *If h is set equal to the minimum value in set S and this value is used to compute d(x) by using Equation (7), then d(x) approaches to a true density.*

This heuristic rule is reasonable for the following reason. In practice, since pattern C has a finite number of points the value for h cannot be made arbitrarily small. Obviously, an appropriate value for h is between the maximum and the minimum distances that are computed by all pairs of points in pattern C.If the value for h is the maximum distance, then C would be inside a single bin. Thus, $d(x)$ approaches to a false density. In contrast, if the value for h is the minimum distance, then the set of the bins would degenerate to the set of the single points in C. This situation also leads to a false density.

According to (Bracewell, 1999), as $h \rightarrow 0$, then $d(x)$ approaches to the true density. Furthermore, a small value for h would be appropriate to approach to the true density (Duda *et. al.*, 2001). Thus, the value for h described in Heuristic Rule 1 is a reasonable selection because it is close to the minimum distance but simultaneously the bins would not degenerate to the single points in C.

4.3 The Proposed Approach

Recall that in optimizing the total misclassification cost as defined in Equation (1) for classification algorithms, one cannot separate the control of fitting and generalization into two independent studies. Instead of this, the key idea of the proposed methodology is to simultaneously balance both fitting and generalization by adjusting the inferred systems through the use of the concept of homogenous sets and the homogeneity degree. The proposed methodology can be summarized in terms of the following three phases:

- **Phase #1**: Apply a classification approach (such as a DT, ANN, or SVM) to infer the two classification systems (i.e., the positive and the negative classification systems). Suppose that each classification system consists of a set of patterns. Next, break the inferred patterns into hyperspheres.
- **Phase #2**: Determine whether the hyperspheres derived in Phase #1 are homogenous sets or not. If so, then go to Phase #3. Otherwise, break a non homogenous set into smaller hyperspheres. Repeat Phase #2 until all of the hyperspheres are homogenous sets.
- **Phase #3**: For each homogenous set, if its homogeneity degree is greater than a certain breaking threshold value, then expand it. Otherwise, break it into smaller homogenous sets. The approach stops when all of the homogenous sets have been processed.

Suppose that given is a homogenous set C. Let *HD(C)* denote its homogeneity degree. There are five parameters which are used in the proposed methodology:

- Two expansion threshold values α^{+} and α^{-} to be used for expanding the positive and the negative homogenous sets, respectively.
- Two breaking threshold values β^{+} and β^{-} to be used for breaking the positive and the negative patterns, respectively.
- A density threshold value γ to be used for determining whether either a positive or a negative hypersphere is approximately a homogenous set or not.

These three phases are also described in Algorithm 1 where they lead to the formulation of six sub-problems as follows:

Algorithm 1: The main algorithm.

```
Input: The training sets with the positive and the negative points.
    A given classification algorithm.
    Values of the control parameters $\alpha^+$, $\alpha^-$, $\beta^+$, $\beta^-$, and $\gamma$.
Output: New positive and negative classification systems.
 1: Call Sub-Problem #1. {Phase #1}
 2: Call Sub-Problem #2.
 3: for all hypersphere $C$ do {Phase #2}
 4:    Call Sub-Problem #3 with inputs $C$ and $\gamma$.
 5:    if $C$ is a non homogenous set then
 6:       Call Sub-Problem #4
 7:       Go To Step 3
 8:    end if
 9: end for
10: Sort the homogeneity degrees in decreasing order.
11: for all homogenous set $C$ do {Phase #3}
12:    if $HD(C) \geq \beta^+$ (for positive sets) or $HD(C) \geq \beta^-$ (for negative sets) then
13:       Call Sub-Problem #5 with inputs $HD(C)$ and $\alpha^+$ or $\alpha^-$.
14:    else
15:       Call Sub-Problem #6.
16:    end if
17: end for
```

- **Sub-Problem #1:** Apply a data mining approach (such as a DT, ANN, SVM) to infer the two classification systems.
- **Sub-Problem #2:** Break the inferred patterns into hyperspheres.
- **Sub-Problem #3:** Determine whether a hypersphere is a homogenous set or not. If so, then its homogeneity degree is estimated.
- **Sub-Problem #4:** If a hypersphere is not a homogenous set, then break it into smaller hyperspheres.
- **Sub-Problem #5:** Expand a homogenous set C by using the notion of its homogeneity degree *HD(C)* and the corresponding expansion threshold value plus some stopping conditions.
- **Sub-Problem #6:** Break a homogenous set C into smaller homogenous sets.

To solve Sub-Problem #1, one simply applies a classification algorithm and then derives the classification patterns. Furthermore, a solution to Sub-Problem #2 is similar to solutions for Sub-Problem #4. Therefore, the following sections present some procedures for solving Sub-Problems #2, #3, #5, and #6.

4.4 Solving Sub-Problem # 2

In order to help motivate the solution to Sub-Problem #2, we first consider the situation depicted in Figure 11.(a). This figure presents a set of positive

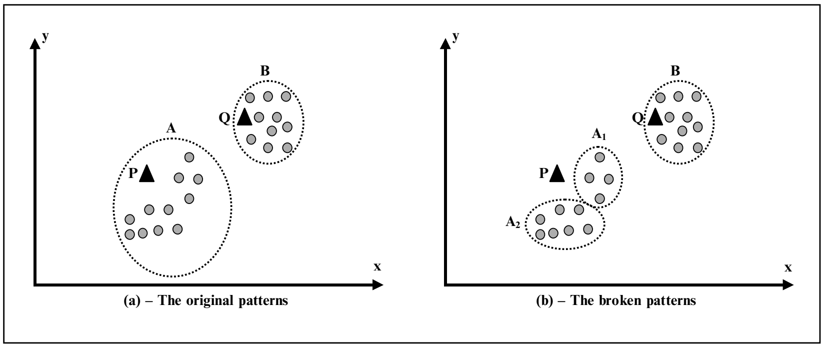


Fig. 8. Pattern B is a homogenous set while pattern A is a non homogenous set. Pattern A can be replaced by the two homogenous sets A_1 and A_2 as shown in part (b).

training points and a set of negative training points in 2-D. Suppose that Sub-Problem #1 has applied a DT algorithm on these sample data to infer a decision tree as depicted in Figure 11.(b). This decision tree separates the training data into the four groups described by the two solid lines depicted in Figure 11.(c).

Next for each pattern somehow Sub-Problem #2 finds the minimum number of hyperspheres which cover all the points in the original patterns. For instance, the above situation is depicted in Figure 11.(d) in which the positive patterns and the bottom negative pattern are covered by the circles (please note that in 2-D hyperspheres are circles): B, D and C, respectively. The top negative pattern is covered by the two circles A and E.

The problem of finding the minimum number of hyperspheres that can cover a pattern C of size N is similar to a form of the s*et cover problem,* an NP-complete problem (Karp, 1972). In this research, a heuristic algorithm is proposed as depicted in Algorithm 2.

The algorithm starts by first estimating the densities of the N points by using Equation (7). Assume that the value for K is going from 1 to N. The algorithm will pick K points in C with the highest densities. Next, it uses these K points as centroids in the K-means clustering approach. If the K hyperspheres which are obtained from the clustering approach cover C, then the algorithm will stop. Otherwise, we repeat the algorithm with the value for K increased by one. Obviously, the algorithm will stop after some iterations because of Axiom 1. For instance, in Figure 11.(d) the algorithm determines at least two circles which can cover the two positive patterns while it uses three circles for the two negative patterns.

Recall that Sub-Problem #4 is to decompose a non homogenous set C into smaller hyperspheres in order to minimize the number of the hyperspheres which cover pattern C. We can use a similar algorithm as the one depicted in Algorithm 2.

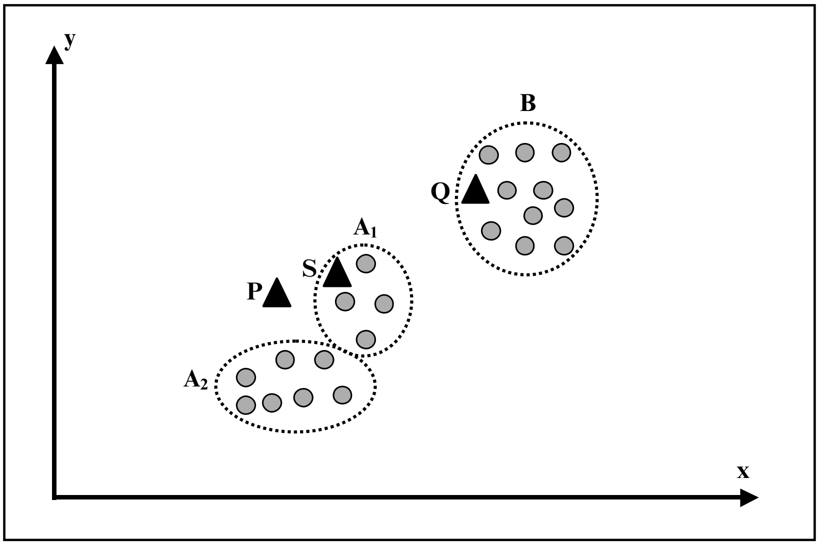


Fig. 9. An illustrative example of homogenous sets.

Algorithm 2: The algorithm for Sub-Problem 2

```
Input: Pattern $C$ of size $N$.
Output: $K$ hyperspheres.
 1: Estimate the densities of the $N$ points by using Equation (7).
 2: for $K$=1 to $N$ do
 3:    Pick $K$points in $C$ with the highest densities.
 4:    Use the $K$-means clustering approach to find $K$ hyperspheres.
 5:    if the $K$ hyperspheres cover $C$ then
 6:       STOP
 7:    else
 8:       $K = K + 1$
 9:    end if
10: end for
```

4.5 Solving Sub-Problem #3

Let consider some hypersphere C. Sub-Problem #3 determines whether or not hypersphere C is a homogenous set. By using the idea of the non parametric density estimation described in Section 4.2, C is divided into a number of small bins of unit size h and approximates the density at the center $x$of each bin. If the densities at the centers are approximately equal to each other, then C is a homogenous set.

In order to help motivate the algorithm for Sub-Problem #3, we first consider the situation depicted in Figure 12. The left side of this figure presents two positive circles, called A and B in 2-D.

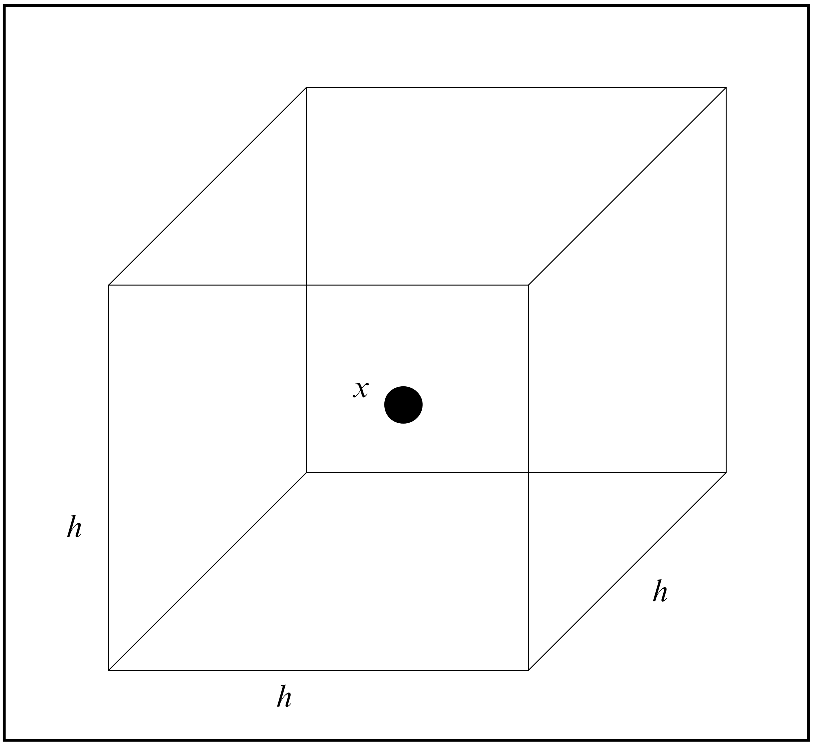


Fig. 10. A bin of unit size h and the center x in 3-D.

Algorithm 3: The algorithm for Sub-Problem 3

```
Input: Hypersphere $C$ and density threshold value $\gamma$.
Output: Decide whether or not hypersphere $C$ is a homogenous set.
 1: Compute the distances between all pairs of points in $C$.
 2: Let $h$ be the distance mentioned in Heuristic Rule 1.
 3: Superimpose $C$ into hypergrid $V$ of unit size $h$.
 4: Approximate the density at the center $x$of each bin.
 5: Compute the standard deviation of the densities at the centers of the bins.
 6: if the standard deviation is $\leq$ess than or equal to $\gamma$, then
 7:     $C$ is a homogenous set and its homogeneity degree HD(C) is computed by
        using Equation (9).
 8: else
 9:     $C$ is not a homogenous set.
10: end if
```

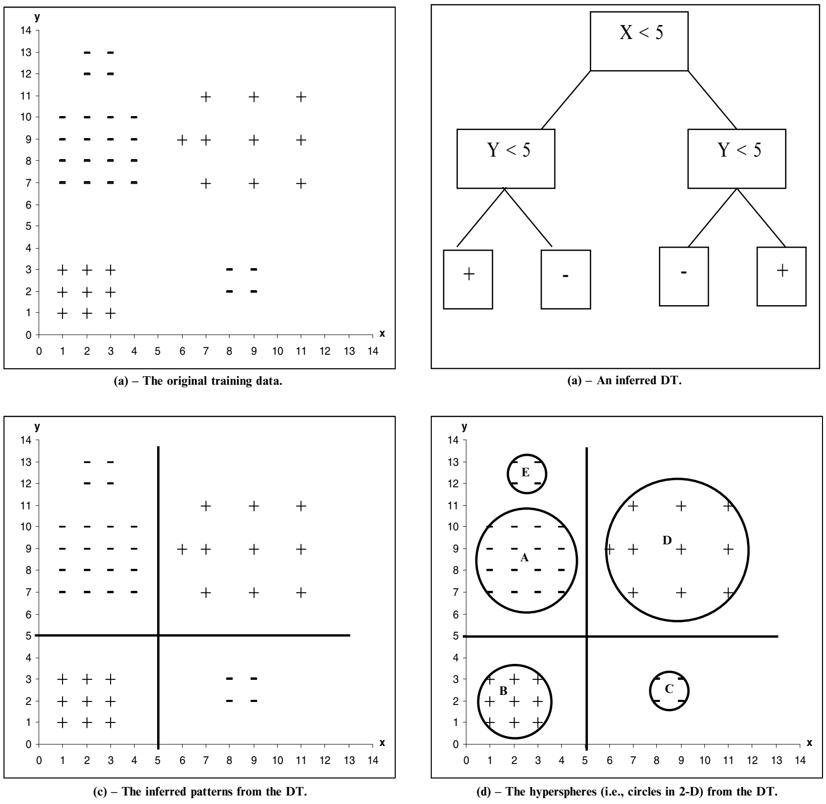

(a) – The original training data.
X < 5
Y < 5
Y < 5
(a) – An inferred DT.
(c) – The inferred patterns from the DT.
A
B
C
D
E
(d) – The hyperspheres (i.e., circles in 2-D) from the DT.

Fig. 11. An Illustrative example of Phase 1

Suppose that both circles A and B are superimposed by the same hypergrid V of unit size h equal to one. This situation is depicted in the right side of Figure 12. By using Equation (7), the right figures show that all bins in circle A are of the same density equal to $\frac{1}{16\times 1^2}$ =0.0625. In contrast, the density of some of the bins in circle B is equal to $\frac{0}{16\times 1^2}$ =0. Thus, circle A is a homogenous set while circle B is not.

Furthermore, instead of the strict condition which requires the same density at the centers of the bins, we may apply a softer condition. That is, if the standard deviation of the densities at the centers of the bins is approximately less or equal to γ, say for $\gamma = 0.01$, then hypersphere C may be considered to be a homogenous set. The algorithm for Sub-Problem #3 is given in Algorithm 3.

As mentioned in Section 4.1, the homogeneity degree *HD(C)* is a factor that may affect the total misclassification cost of the inferred classification systems. If an unclassified point is covered by a homogenous set C which has a higher homogeneity degree, then it may more accurately be assumed to be

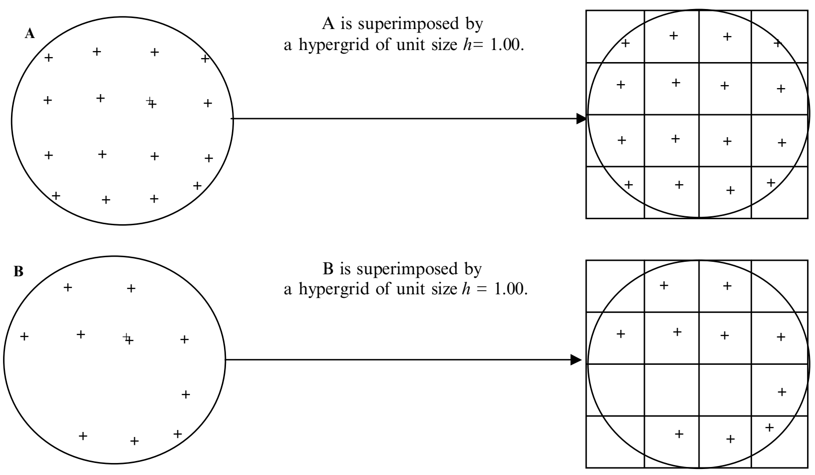

A
A is superimposed by a hypergrid of unit size h= 1.00.
B
B is superimposed by a hypergrid of unit size h = 1.00.

Fig. 12. Illustrative examples of the homogenous set (at the top part) and the non homogenous set (at the bottom part).

of the same class as the points covered by the homogenous set C. Thus, a definition for *HD(C)* is an important step in improving the accuracy of the classification systems.

As discussed in Section 4.1, the concept of the homogeneity degree *HD(C)* is defined as the number of points inside the homogenous set C per unit of C's volume. This definition, however, has its drawbacks. For instance, let us look at circles A and E as the one depicted in Figure 11. According to the above definition, *HD(*A) is equal to $\frac{16}{2\times1.5^2\times\pi} \approx 1.1318$, while *HD(*E) is equal to$\frac{4}{2\times0.5^2\times\pi} \approx 2.5465$. This means that pattern E is denser than pattern A. This is an apparent contradiction since in reality pattern A has more points and covers a wider region than pattern E. Thus, we need to find an appropriate definition for the homogeneity degree.

Intuitively, *HD(C)* depends on the value h defined in Heuristic Rule 1 and the number of points in C, denoted by n_C. If n_C increases, then *HD(C)* would slightly increase since the volume of C does not change and C has more points. Furthermore, if h increases, then the average distance between pairs of points in homogenous set C increases. Obviously, this leads to;$D(C)$ decreases. Hence, *HD(C)* is inversely proportional to h while is directly proportional to n_C. We use the function $\ln(n_C)$ to show the slight effect of n_C to *HD(C)*.

$$HD(C) = \frac{\ln(n_C)}{h}. \tag{9}$$

For instance, *HD(*A) as depicted in Figure 12 is equal to $\frac{\ln(16)}{1} \approx 2.77$. Let us consider the illustrative example depicted in Figure 11. Now we have

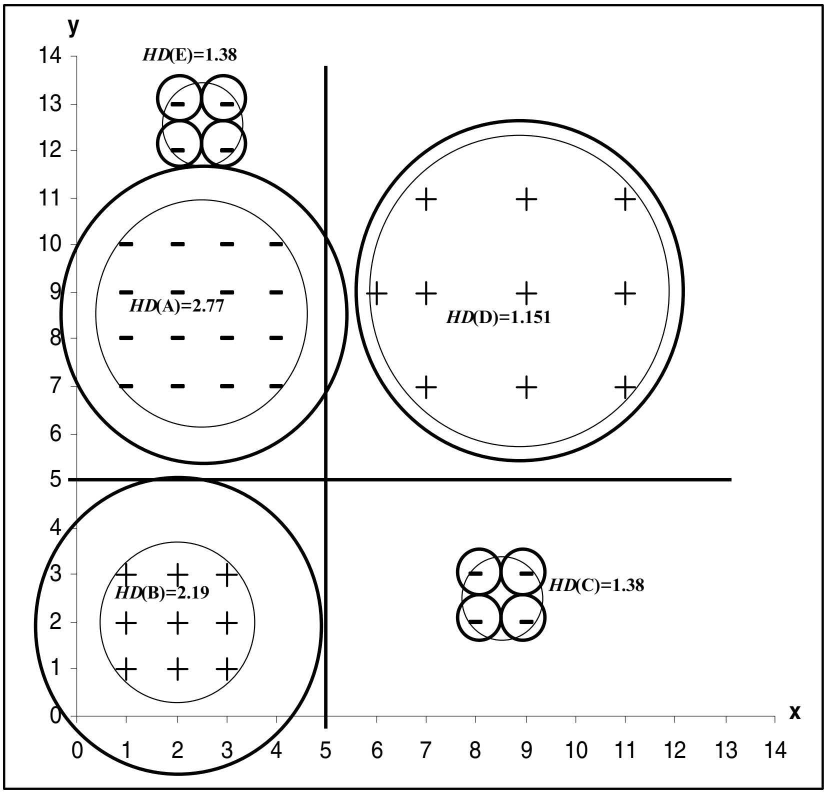


Fig. 13. An illustrative example of Sub-Problem 5.

*HD(*A) equal to $\frac{\ln(16)}{1} \approx 2.77$, *HD(*B) equal to $\frac{\ln(9)}{1} \approx 2.19$, *HD(*C) = *HD(*E) equal to $\frac{\ln(4)}{1} \approx 1.38$, and *HD(*D) equal to $\frac{\ln(10)}{2} \approx 1.151$.

4.6 Solving Sub-Problem #5

Recall that the control of fitting and generalization for classification systems may be achieved by expanding or breaking the inferred homogenous sets by using their homogeneity degrees. Suppose that we are given a positive homogenous set F with its homogeneity degree *HD(F)*, the breaking threshold value β^+, and the expansion threshold value α^+. A similar definition exists for a negative homogenous set. According to the main algorithm depicted in Algorithm 1, if *HD(F)* is greater than or equal to β^+, then the homogenous set F will be expanded by using the expansion threshold value α^+. Otherwise, we will break the homogenous set F into smaller hyperspheres.

In order to help motivate this stage, we consider the example depicted in Figure 11. Please recall that the homogeneity degrees of circles A, B, C, D,

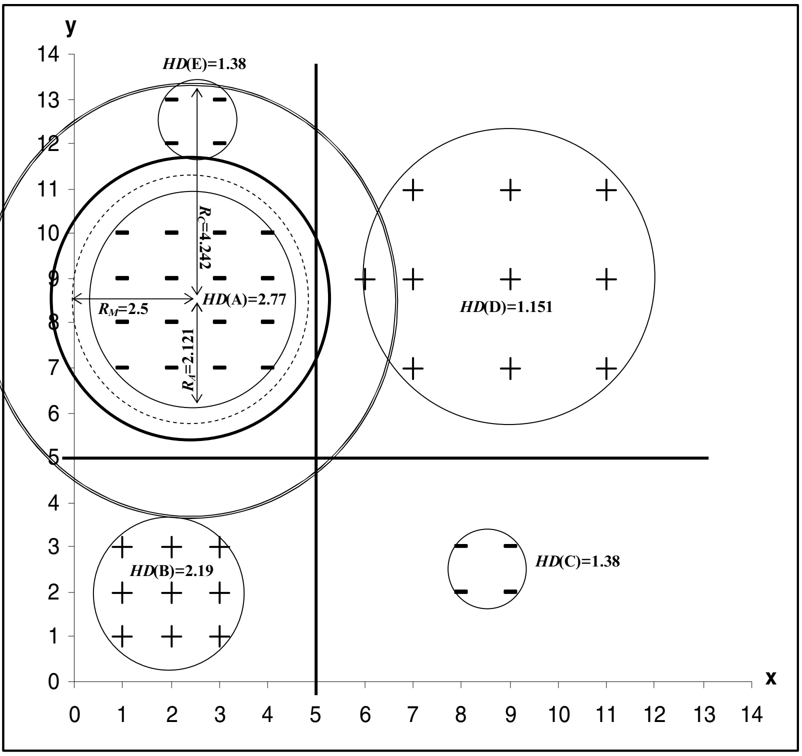


Fig. 14. An illustrative example of the radial expansion.

and E are *HD(*A)=2.77, *HD(*B)=2.19, *HD*(C)=1.38, and *HD(*D)=1.15, and *HD(*E)=1.38, respectively. Suppose that the two breaking threshold values β^+ and β^- are equal to 1.00 and 1.50, respectively. Furthermore, let the two expansion threshold values α^+ and α^- be equal to 2.00. As depicted in Figure 13, the homogenous sets A, B, and D are expanded (the expanded regions are indicated by the solid line circles), while C and E are broken into four smaller circles (the broken regions are indicated by the small solid line circles). Please note that the breaking
approach, i.e., Sub-Problem #6, is described in Section 4.7.

There are two types of expansion: a radial expansion in which a homogenous set F is expanded in all directions and a linear expansion in which a homogenous set F is expanded in a certain direction. For instance, in Figure 13 the homogenous sets A, B, and D have used the radial expansion approach. The following sections discuss in detail these two expansion types.

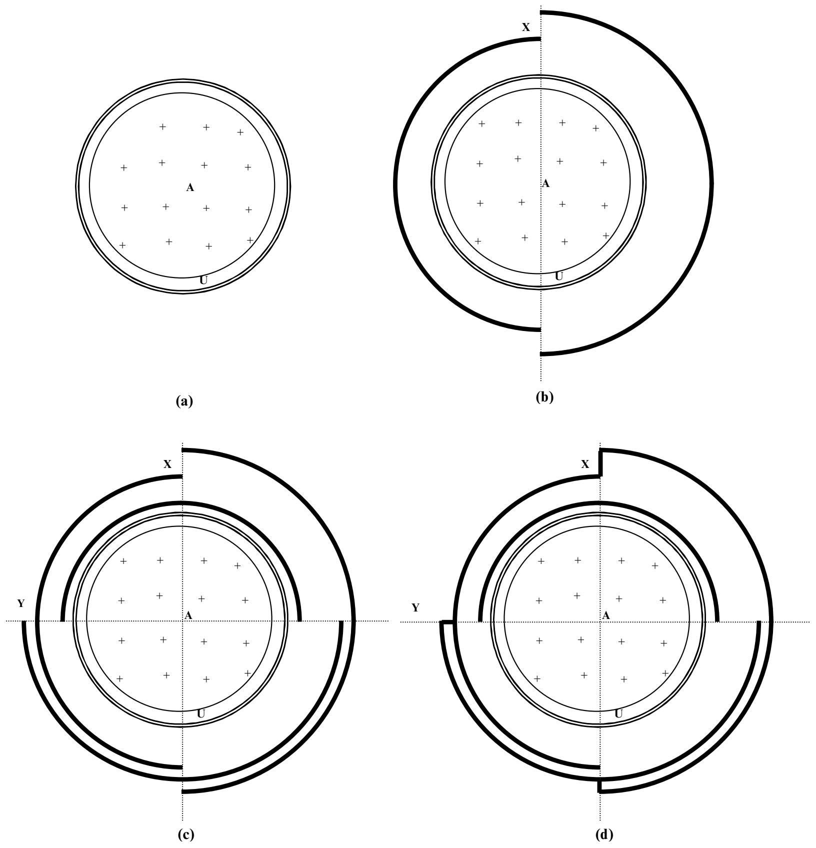


Fig. 15. An illustrative example of the linear expansion.

Radial Expansion

In the radial type, a homogenous set$F$ is expanded in all directions. Let M be a region expanded from F. Let R_F and R_M denote the radiuses of F and M, respectively.In the radial expansion approach R_F is increased by a certain amount denoted as T, called a *step-size increase.* Thus, one gets:

$$R_M = R_F + T \tag{10}$$

Following a dichotomous search methodology, we assume that there exists a hypersphere G which covers the homogenous set F. Furthermore, without loss of generality, let us assume that the radius $R_G$may be computed by:

$$R_G = 2 \times \quad R_F \tag{11}$$

By using R_G and R_F, we can derive the step-size increase$T$. That is, T must depend on the difference between R_G and R_F. One of the ways that T may be determined is as follows:

$$T = \frac{R_G - R_F}{2}. \tag{12}$$

At the same time, T should depend on *HD(F)* because of the dichotomous search methodology. That is, if *HD(F)* gets higher, then T should get smaller. This means that *HD(F)* is inversely proportional to T. We may use a threshold value L to ensure that *HD(F)* is always greater than one. Thus, the value for Tmay be defined as follows:

$$T = \frac{R_G - R_F}{2} \times \frac{1}{L \times HD(F)}. \tag{13}$$

If we substitute back into Equation (10), R_M becomes:

$$R_M = R_F + \frac{R_G - R_F}{2} \times \frac{1}{L \times HD(F)}. \tag{14}$$

In order to help motivate the radial expansion algorithm, we consider the example indicated in Figure 14. This example uses the same hypothetical data as the ones depicted in Figure 11.(d). Assume that L is equal to one. A closer examination of Figure 14 indicates that the hypersphere A (i.e., the one-line circle with R_A=2.121) is covered by the three circles: a double-line circle which depicts circle G with $R_G = 2.121 \times 2 = 4.242$, a solid line circle which shows the final expanded region, and a dotted line circle which presents the hypersphere M whose radius is computed as follows:
$R_M = R_F + \frac{R_G - R_M}{2} \times \frac{1}{L \times w(\mathrm{A})} = 2.121 + \frac{4.242 - 2.121}{2} \times \frac{1}{1 \times 2.77} \approx 2.5.$

Similarly, Equation (14) computes the following values for R_M in four iterations: 2.8, 3.06, 3.23, and 3.25, respectively, until R_M satisfies the stopping conditions mentioned next in Section 4.6. The final expanded region is the solid line circle depicted in Figure 14. Furthermore, this figure also shows that a part of the state space which has been inferred as a positive region by the DT algorithm. However, now it is derived as a negative region after using the HBA. This illustration indicates that the HBA may derive better classification systems. The radial expansion algorithm is depicted in Algorithm 4.

Linear Expansion

The linear approach expands a homogenous set F in a certain direction. There is a difference between the method presented in the previous section and the one presented in this section (i.e., linear vs. radial). That is, now the homogenous set F is first expanded to hypersphere M by using the radial expansion. Then, hypersphere M is expanded in a given direction by using

Algorithm 4: The algorithm for the radial expansion.

```
Input: Homogenous set $F$ with $HD(F)$, $R_F$, and $\alpha^+$
Output: An expanded region $E$.
 1: Set $M = F$ (i.e., $R_F = R_M$).
 2: Set hypersphere $G$ covering $M$ with radius $R_G = 2 \times R_M$.
 3: repeat
 4:    Set $E = M$ (i.e., $R_E = R_M$).
 5:    Expand $M$ by using Equation (14).
 6: until $R_M$ satisfies stopping conditions discussed in Section 4.6 or $R_M = R_G$.
 7: if $R_M$ satisfies stopping conditions then
 8:    STOP.
 9: else
10:    go to Step 2.
11: end if
```

the radial approach until it satisfies the stopping conditions mentioned next in Section 4.6. The final region is the union of all the expanded regions.

In order to help motivate the linear expansion approach, we consider the homogenous set A depicted in Figure 15. Suppose that by using the radial expansion for the homogenous set A with the expansion threshold value equal to 2.00, we get the hypersphere U (i.e., the two-line circle depicted in Figure 15.(a)). Next, we divide the hypersphere U in the X axis into two parts. The radial expansion approach would expand each one of the parts as the solid lines depicted in Figure 15.(b). A similar approach exists for the Y axis depicted in Figure 15.(c). The final expanded region is the region which is defined by the union of the solid lines depicted in Figure 15.(d).

Description of the Stopping Conditions

This section presents the stopping conditions for the radial expansion approach for expanding a homogenous set F. That is, the stopping conditions must satisfy the following requirements:

Depend on the homogeneity degree. This has been mentioned in the fundamental assumption of the proposed approach.

Stop when an expanded region reaches other patterns. We can use a softer condition in which the expanded region can accept several noisy data points. If the homogeneity degree is high, then the expanded region can accept more noisy data.

To address the first stopping condition, an upper bound for R_M should be directly proportional to the homogeneity degree $HD(F)$, the expansion threshold value α^+, and the original radius R_F. The second stopping condition can be determined while expanding. Furthermore, an upper bound on the number of noisy points should be directly proportional to $HD(F)$ and the size of F, which is denoted as n_F. The stopping conditions are summarized as follows (a similar way exists for the expansion threshold value α^-):

$$R_M \leq HD(F) \times R_F \times \alpha^+ \text{ and } \textit{the number of noisy points} \leq \frac{HD(F) \times \alpha^+}{n_F} \tag{15}$$

4.7 Solving Sub-Problem #6

Suppose that given is a positive homogenous set F. Recall that if its homogeneity degree *HD(F)* is less than β^+, then the homogenous set F is broken into sub-patterns. According to Theorem 1, the sub-patterns are also homogenous sets. Thus, they can be expanded or broken down even more.

In order to help motivate this problem, we consider the example depicted in Figure 13. In this figure the two threshold values β^+ and β^- are equal to 1.00 and 1.50, respectively. Therefore, the homogenous sets C and E are broken down into four smaller circles for each set. Then, these smaller circles are considered to be homogenous sets with their homogeneity degrees equal to zero. Thus, they should not be expanded.

5 Some Computational Results

5.1 Datasets and Parametric Analysis

Please recall that this chapter aims at better understanding the performance of the HBA in balancing both fitting and generalization by adjusting the inferred systems through the use of the concept of homogenous sets and the homogeneity degree. The balance will target at minimizing the total misclassification costs, TC, of the final system:

$$TC = \min\left(C_{FP} \times RATE\_FP + C_{FN} \times RATE\_FN + C_{UC} \times RATE\_UC\right).$$

Please note that the penalty costs: C_{FP}, C_{FN}, and C_{UC} depend on each individual application. In the following experiments, we used some 2-D synthetic datasets which were divided into a training set and a testing set as described in Table 1.These data points were determined as follows. At first the map of VietNam was considered. Next some data points were generated randomly in 2-D. A data point would be a positive point, if it fell inside the map of VietNam. Otherwise, that point was defined as a negative point. The HBA attempted to use the training set to infer the map of VietNam (i.e., the positive and the negative systems). Then, we used the inferred map to test the testing set. The four parameters used in the HBA are as follows:

- Two expansion threshold values α^+ and α^- to be used for expanding the positive and the negative homogenous sets, respectively.

Table 1. Characteristics of the 2-D synthetic datasets

Name	Number of training points	Number of testing points
D_1	63	16
D_2	89	28
$D_3 = D_1 \cup D_2$	144	44

- Two breaking threshold values β^+ and β^- to be used for breaking the positive and the negative patterns, respectively.

Furthermore, it was also assumed that β^+ and β^- were in [0, 2] while α^+ and α^- were in [0, 10]. Given is a certain 3-tuple of the penalty costs (C_{FP}, C_{FN}, C_{UC}). By using exhaustive search in the above ranges the HBA found the optimal combinations of α^+, α^-, β^+,and β^- in order to minimize the TC value.On the other hand, given are different values for the 3-tuple (C_{FP}, C_{FN}, C_{UC}). We expect that the value for TC after controlling the fitting and generalization problems would be less than or at most equal to what was achieved by the original algorithms.

5.2 Experimental Results

The experiments were ran on a PC with 2.8GHZ speed and 1GB RAM under the Windows XP operating system. The original classification algorithms used in these experiments are based on SVMs, ANNs, and DTs. There were thirteen experiments done on the three datasets D_1, D_2, and $D_3$with different values for the 3-tuple (C_{FP}, C_{FN}, C_{UC}). Furthermore, we used the libraries in Neural Network Toolbox 6.0 and Statistics Toolbox 6.0 (Matlab, 2004) for implementing the classification algorithms, the K-means clustering algorithm, and the density estimation approach. The experimental details are as follows:

Case 1: At first we studied the case of a 3-tuple (C_{FP}, C_{FN}, C_{UC}) in which the application would penalize much more for the false-positive cases than for the other types of error. Thus, the objective function in this case was assumed to be:

$$TC = 6\times RATE\_FP + 3\times RATE\_FN + RATE\_UC.$$

Next, we ran the HBA on D_1 with β^+ and β^- divided into {0, 1, 2} and α^+ and α^- divided into {0, 2, 4, 6, 8, 10}. Recall that *RATE_FP*, *RATE_FN*, and *RATE_UC* are the false-positive, the false-negative, and the unclassifiable rates, respectively. Table 2 shows these three rates and the value of TC obtained from the algorithms. The notation "SVM-HBA" means that the HBA used the classification models first obtained by using the SVM algorithm before controlling the fitting and generalization problems. The two similar notations exist for DT-HBA (the Decision Tree algorithm and the HBA) and

Table 2. Results for minimizing TC = 6×RATE_FP + 3×RATE_FN + RATE_UC on D_1.

Algorithm	*RATE_FP*	*RATE_FN*	*RATE_UC*	*TC*
SVM	1	0	7	13
DT	3	0	5	23
ANN	1	0	7	13
SVM-HBA	0	1	5	8
DT-HBA	0	1	5	8
ANN-HBA	0	1	5	8

ANN-HBA (the Artificial Neural Network algorithm and the HBA). Table 2 presents that SVM-HBA, DT-HBA, and ANN-HBA found the optimal *TC* to be equal to 8. This value was less than the value of *TC* achieved by the original algorithms (i.e., the SVM, DT, and ANN) by about 39%.

Table 3 presents information for the four specific parameter values when SVM-HBA found the optimal *TC*. The execution time in this case was approximately equal to 1 hour and 3 minutes.

Table 3. Values for the four parameters when the SVM-HBA ran on D_1 and found the optimal TC.

β^+	α^-	β^-	α^+
1	10	2	4
1	10	2	6
1	10	2	8
1	10	2	10

An even lower *TC* was found once we divided β^+ and β^- into {0, 1, 2} and divided α^+ and α^- into {0 to10}. These results are presented in Table 4.

Table 4. Results for minimizing TC = 6×RATE_FP + 3×RATE_FN + RATE_UC on D_1 with the smaller ranges.

Algorithm	*RATE_FP*	*RATE_FN*	*RATE_UC*	*TC*
SVM	1	0	7	13
DT	3	0	5	23
ANN	1	0	7	13
SVM-HBA	0	0	7	7
DT-HBA	0	0	7	7
ANN-HBA	0	0	7	7

Table 4 shows that if we split the four parameters into smaller ranges, then the HBA could find a lower *TC*. This may lead to a new strategy in which one can develop an approach for determining optimal combinations of the four parameter values by successively considering higher resolution.

Case 2: Now we consider a case in which the application would penalize much more for the unclassifiable cases than for the other types of error. Thus, the objective function in this case was assumed to be:

$$TC = RATE\_FP + 3 \times RATE\_FN + 6 \times RATE\_UC.$$

We ran the HBA on D_1 with β^+ and β^- divided into {0, 1, 2} and α^+ and α^- divided into {0 to 10}. Table 5 shows that SVM-HBA, DT-HBA, and ANN-HBA found an optimal *TC* which was less than the value of *TC* achieved by the original algorithms by about 53%.

Table 5. Results for minimizing TC = RATE_FP + 3×RATE_FN +6×RATE_UC on D_1.

Algorithm	*RATE_FP*	*RATE_FN*	*RATE_UC*	*TC*
SVM	1	0	7	43
DT	3	0	5	33
ANN	1	0	7	43
SVM-HBA	1	1	4	28
DT-HBA	2	1	2	17
ANN-HBA	1	1	4	28

Case 3: Now we consider a case in which the application would penalize the same way for the false-positive, the false-negative, and the unclassifiable cases. Thus, the objective function in this case was assumed to be:

$$TC = 3.3 \times RATE\_FP + 3.3 \times RATE\_FN + 3.3 \times RATE\_UC.$$

We ran the HBA on D_1 with β^+ and β^- divided into {0, 1, 2} and α^+ and α^- divided into {0, 2, 4, 6, 8, 10}. Table 6 shows that SVM-HBA, DT-HBA, and ANN-HBA found two possible cases for each algorithm where the optimal value for *TC* was less than the value of *TC* achieved by the original algorithms by about 33%.

A similar result for *TC* once we ran the HBA on D_3, which had more training points, also divided β^+ and β^- into {0, 1, 2}, and α^+ and α^- into {0, 2, 4, 6, 8, 10}. These results are presented in Table 7.

Table 7 shows that SVM-HBA, DT-HBA, and ANN-HBA found the optimal *TC* which was less than the value of *TC* achieved by the original algorithms by about 47%. The execution time in this case was approximately equal to 12 hours and 50 minutes.

Table 6. Results for minimizing TC = 3.3×RATE_FP + 3.3×RATE_FN +3.3×RATE_UC on D_1.

Algorithm	*RATE_FP*	*RATE_FN*	*RATE_UC*	*TC*
SVM	1	0	7	26.4
DT	3	0	5	26.4
ANN	1	0	7	26.4
SVM-HBA	0	1	5	19.8
	1	1	4	19.8
DT-HBA	1	1	3	16.5
	2	1	2	16.5
ANN-HBA	0	1	5	19.8
	1	1	4	19.8

Table 7. Results for minimizing TC = 3.3×RATE_FP + 3.3×RATE_FN + 3.3RATE_UC on D_3

Algorithm	*RATE_FP*	*RATE_FN*	*RATE_UC*	*TC*
SVM	5	3	26	112.2
DT	8	3	24	115.5
ANN	5	2	27	112.2
SVM-HBA	4	7	7	59.40
DT-HBA	7	7	9	75.90
ANN-HBA	4	7	7	59.40

Case 4: Now we consider a case in which the application would penalize much more for the false-negative cases than for the other types of error. Furthermore, the penalty cost for unclassifiable cases was equal to zero Thus, the objective function in this case was assumed to be:

$$TC = 2\times RATE\_FP + 20\times RATE\_FN + 0\times RATE\_UC.$$

We ran the HBA on D_2 with β^+ and β^- divided into {0, 1, 2} and α^+ and α^- divided into {0, 2, 4, 6, 8, 10}. Table 8 shows that SVM-HBA, DT-HBA, and ANN-HBA found an optimal TC equal to 0. This value was equal to the value of TC achieved by the original algorithms. However, SVM-HBA achieved an unclassifiable rate of 21 versus 28 for the original algorithms. A similar result existed for DT-HBA and ANN-HBA. The execution time in this case was approximately equal to 5 hours and 24 minutes.

We also experimented with the following different objective functions on the dataset D_1:

$$TC = 6\times RATE\_FP + 2\times RATE\_FN + 2\times RATE\_UC,$$

$$TC = 4\times RATE\_FP + 2\times RATE\_FN + 4\times RATE\_UC, \text{ and}$$

Table 8. Results for minimizing TC = 2×RATE_FP +20×RATE_FN on D_2.

Algorithm	*RATE_FP*	*RATE_FN*	*RATE_UC*	*TC*
SVM	0	0	28	0
DT	0	0	28	0
ANN	0	0	28	0
SVM-HBA	0	0	21	0
DT-HBA	0	0	24	0
ANN-HBA	0	0	22	0

$$TC = 3\times RATE\_FP + 6\times RATE\_FN + 1\times RATE\_UC.$$

Similarly, we experimented with the following different objective functions on the dataset D_2:

$$TC = 2\times RATE\_FP + 20\times RATE\_FN + 0\times RATE\_UC,$$

$$TC = 6\times RATE\_FP + 3\times RATE\_FN + 1\times RATE\_UC, \text{ and}$$

$$TC = 50\times RATE\_FP + 60\times RATE\_FN + 1\times RATE\_UC.$$

We also experimented with the following different objective functions on the dataset D_3:

$$TC = RATE\_FP + 3\times RATE\_FN + 6\times RATE\_UC, \text{ and}$$

$$TC = 20\times RATE\_FP + 2\times RATE\_FN + 0\times RATE\_UC.$$

In all these tests we concluded that the HBA always found the optimal combinations of α^+, α^-, β^+,and β^- in order to minimize the value of *TC*. Furthermore, the value for *TC* in all these cases was significantly less than or at most equal to what was achieved by the original algorithms.

6 Conclusions

The performance of a classification method in terms of the false-positive, the false-negative, and the unclassifiable rates may be totally unpredictable and depend on the application at hand. Attempts to minimize one of the previous rates, lead to increases on the other two rates. The root to the above critical problems is the overfitting and overgeneralization behaviors of a given classification approach when it is processing a particular dataset. This chapter identified a gap between fitting and generalization with current algorithms and also defined the desired goal as an optimization problem. Next, it provided a new approach, called the Homogeneity-Based Algorithm (HBA), which appears to be very promising. There are some future research goals. For example, the HBA needs to be tested with higher dimensions and more data. This is ongoing research by our group. Currently we are implementing a GA (Genetic Algorithm) for finding the optimal values of the controlling threshold values α^+, α^-, β^+,and β^-. Some preliminary results seem to suggest that by using the GA one can achieve even better values for the various objectives functions at a fraction of the original CPU time (often times by spending between 50% to 80%).;

References

Abdi, H., (2003), "*A neural network primer,*" Journal of Biological Systems, vol. 2, pp. 247-281.

Ali, K., C. Brunk, and M. Pazzani, (1994), "*On learning multiple descriptions of a concept,*" Proceedings of Tools with Artificial Intelligence, New Orleans, LA, USA, pp. 476-483.

Artificial Neural Network Toolbox 6.0 and Statistics Toolbox 6.0, Matlab Version 7.0, website: http://www.mathworks.com/products/

Boros, E., P. L. Hammer, and J. N. Hooker, (1994), "*Predicting Cause-Effect Relationships from Incomplete Discrete Observations,*" Journal on Discrete Mathematics, vol. 7, no. 4, pp. 531-543.

Bracewell, R., (1999), "*The Impulse Symbol,*" Chapter 5 in The Fourier Transform and Its Applications, 3rd ed. New York: McGraw-Hill, pp. 69-97.

Breiman, L., (1996), "*Bagging predictors,*" Journal of Machine Learning, vol. 24, pp. 123-140.

Breiman, L., (2001), "*Random forests,*" Journal of Machine Learning, vol. 45, no. 1, pp. 5–32.

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone, (1984), "*Classification and Regression Trees,*" Chapman Hall/CRC Publisher, pp. 279-293.

Byvatov, E., and G. Schneider, (2003), "*Support vector machine applications in bioinformatics,*" Journal of Application Bioinformatics, vol. 2, no.2, pp. 67-77.

Clark, P., and R. Boswell, (1991), "*Rule induction with CN2: Some recent improvements,*" Y. Kodratoff, editor, Machine Learning - EWSL-91, Berlin, Springer-Verlag, pp. 151-163.

Clark, P., and T. Niblett, (1989), "*The CN2 Algorithm,*" Journal of Machine Learning, vol. 3, pp. 261-283.

Cohen S., L. Rokach, O. Maimon, (2007), "*Decision-tree instance-space decomposition with grouped gain-ratio,*", Information Science, Volume 177, Issue 17, pp. 3592-3612.

Cohen, W. W., (1995), "*Fast effective rule induction,*" Machine Learning: Proceedings of the Twelfth International Conference, Tahoe City, CA., USA, pp. 115-123.

Cortes, C., and V. Vapnik, (1995), "*Support-vector networks,*" Journal of Machine Learning, vol. 20, no. 3, pp. 273-297.

Cover, T. M., and P. E. Hart, (1967), "*Nearest Neighbor Pattern Classification,*" Institute of Electrical and Electronics Engineers Transactions on Information Theory, vol. 13, no. 1, pp. 21-27.

Cristianini, N., and S. T. John, (2000), "*An Introduction to Support Vector Machines and other kernel-based learning methods,*" Cambridge University Press.

Dasarathy, B. V., and B. V. Sheela, (1979), "*A Composite Classifier System Design: Concepts and Methodology,*" Proceedings of the IEEE, vol. 67, no. 5, pp. 708-713.

Dietterich, T. G., and G. Bakiri, (1994), "*Solving multiclass learning problems via error-correcting output codes,*" Journal of Artificial Intelligence Research, vol. 2, pp. 263-286.

Duda, R. O., and P. E. Hart, (1973), "*Pattern Classification and Scene Analysis,*" Wiley Publisher, pp. 56-64.

Duda. O. R., E. H. Peter, G. S. David , (2001), "*Pattern Classification,*" Chapter 4: Nonparametric Techniques in Wiley Interscience Publisher, pp. 161-199.

Dudani, S., (1976), "*The Distance-Weighted k-Nearest-Neighbor Rule,*" IEEE Transactions on Systems, Man and Cybernetics, vol. 6, no. 4, pp. 325-327.

Friedman, N., D. Geiger, and M. Goldszmidt, (1997), "*Bayesian Network Classifiers,*" Journal of Machine Learning, vol. 29, pp. 131-161.

Geman, S., E. Bienenstock, and R. Doursat, (1992), "*Neural Networks and the Bias/Variance Dilemma,*" Journal of Neural Computation, vol. 4, pp. 1-58.

Hecht-Nielsen, R., (1989), "*Theory of the Backpropagation neural Network,*" International Joint Conference on neural networks, Washington, DC, USA, pp. 593-605.

Huzefa, R., and G. Karypis, (2005), "*Profile Based Direct Kernels for Remote Homology Detection and Fold Recognition,*" Journal of Bioinformatics, vol. 31, no. 23, pp. 4239-4247.

Karp, R. M., (1972), "*Reducibility Among Combinatorial Problems,*" Proceedings of Sympos. IBM Thomas J. Watson Res. Center, Yorktown Heights, New York: Plenum, pp. 85-103.

Keller, J. M., M. R. Gray, and J. A. Givens, Jr, (1985), "*A Fuzzy K-Nearest Neighbor Algorithm,*" Journal of IEEE Transactions on Systems, Man, and Cybernetics, vol. 15, no. 4, pp. 580-585.

Kohavi R., (1996), "*Scaling up the accuracy of naive-Bayes classifiers: a decision-tree hybrid,*" Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Portland, OR, USA, pp. 202-207.

Kohavi, R., and G. John, (1997), "*Wrappers for Feature Subset Selection,*" Journal of Artificial Intelligence: special issue on relevance, vol. 97, no. 1-2, pp. 273-324.

Kokol, P., M. Zorman, M. M. Stiglic, and I. Malcic, (1998), "*The limitations of decision trees and automatic learning in real world medical decision making,*" Proceedings of the 9th World Congress on Medical Informatics MEDINFO'98, vol. 52, pp. 529-533.

Kononenko, I., (1991), "*Semi-naïve Bayesian classifier,*" Y. Kodratoff Editor, Proceedings of sixth European working session on learning, Springer-Verlag, pp. 206-219.

Kwok, S., and C. Carter, (1990), "*Multiple decision trees: uncertainty,*" Journal of Artificial Intelligence, vol.4, pp. 327-335.

Langley, P., and S. Sage, (1994), "*Induction of Selective Bayesian Classifiers,*" Proceedings of UAI-94, Seattle, WA, USA, pp. 399-406.

Mansour, Y., D. McAllester, (2000), "*Generalization Bounds for Decision Trees,*" Proceedings of the 13th Annual Conference on Computer Learning Theory, San Francisco, Morgan Kaufmann, USA, pp. 69–80.

Moody, J. E., (1992), "*The Effective Number of Parameters: An Analysis of Generalization and Regularization in Nonlinear Learning Systems,*" Journal of Advances in Neural Information Processing Systems, vol. 4, pp. 847-854.

Nock, R., and O. Gascuel, (1995), "*On learning decision committees,*" Proceedings of the Twelfth International Conference on Machine Learning, Morgan Kaufmann, Taho City, CA., USA, pp. 413-420.

Oliver, J. J., and D. J.Hand, (1995), "*On pruning and averaging decision trees,*" Proceedings of the Twelfth International Conference on Machine Learning, Morgan Kaufmann, Taho City, CA., USA, pp. 430-437.

Pazzani, M.J., (1995), "*Searching for dependencies in Bayesian classifiers,*" Proceedings of AI STAT'95, pp. 239-248.

Podgorelec, V., P. Kokol, B. Stiglic, and I. Rozman, (2002), "*Decision trees: an overview and their use in medicine,*" Journal of Medical Systems, Kluwer Academic/Plenum Press, vol. 26, no. 5, pp. 445-463

Quinlan, J. R., (1987), "*Simplifying decision trees,*" International Journal of Man-Machine Studies, vol. 27, pp. 221-234.

Quinlan, J. R., (1993), "*C4.5: Programs for Machine Learning,*" Morgan Kaufmann Publisher San Mateo, CA., USA, pp. 35-42.

Rada, M., (2004), "*Seminar on Machine Learning,*" a presentation of a course taught at University of North Texas.

Rokach L., O. Maimon, O. Arad, (2005), "*Improving Supervised Learning by Sample Decomposition,*" Journal of Computational Intelligence and Applications, vol. 5, no. 1, pp. 37-54.

Sands D., (1998), "*Improvement theory and its applications,*" Gordon A. D., and A. M. Pitts Editors, Higher Order Operational Techniques in Semantics, Publications of the Newton Institute, Cambridge University Press, pp. 275-306.

Schapire, R. E, (1990), "*The strength of weak learnability,*" Journal of Machine Learning, vol. 5, pp. 197-227.

Shawe-Taylor. J., and C. Nello, (1999), "*Further results on the margin distribution,*" Proceedings of COLT99, Santa Cruz, CA., USA, pp. 278-285.

Smith, M., (1996), "*Neural Networks for Statistical Modeling,*" Itp New Media Publisher, ISBN 1-850-32842-0, pp. 117–129.

Spizer, M., L. Stefan, C. Paul, S. Alexander, and F. George, (2006), "*IsoSVM – Distinguishing isoforms and paralogs on the protein level,*" Journal of BMC Bioinformatics, vol. 7:110,

website: http://www.biomedcentral.com/content/pdf/1471-2105-7-110.pdf.

Tan, P. N., S. Michael, and K. Vipin, (2005), "*Introduction to Data Mining,*" Chapters 4 and 5, Addison-Wesley Publisher, pp. 145-315.

Triantaphyllou, E., (2007), "*Data Mining and Knowledge Discovery Via a Novel Logic-Based Approach,*" A monograph, Springer, Massive Computing Series, 420 pages, (in print).

Triantaphyllou, E., and G. Felici, (Editors), (2006), "*Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques,*" Springer, Massive Computing Series, 796 pages.

Triantaphyllou, E., L. Allen, L. Soyster, and S. R. T. Kumara, (1994), "*Generating Logical Expressions From Positive and Negative Examples via a Branch-and-Bound approach,*" Journal of Computers and Operations Research, vol. 21, pp. 783-799.

Vapnik, V., (1998), "*Statistical Learning Theory,*" Wiley Publisher, pp. 375-567.

Webb, G. I., (1996), "*Further experimental evidence against the utility of Occam's razor,*" Journal of Artificial Intelligence Research, vol. 4, pp. 397-417.

Webb, G. I., (1997), "*Decision Tree Grafting,*" Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97), vol. 2, pp. 23-29.

Weigend, A., (1994), "*On overfitting and the effective number of hidden units,*" Proceedings of the 1993 Connectionist Models Summer School, pp. 335-342.

Wikipedia Dictionary, (2007), website: http://en.wikipedia.org/wiki/Homogenous.

Wolpert, D. H, (1992), "*Stacked generalization,*" Journal of Neural Networks, vol. 5, pp. 241-259.

Zavrsnik, J., P. Kokol, I. Maleiae, K. Kancler, M. Mernik, and M. Bigec, (1995), "*ROSE: decision trees, automatic learning and their applications in cardiac medicine,*" MEDINFO'95, Vancouver, Canada, pp. 201-206.

Zhou Z. and C. Chen, (2002), "*Hybrid decision tree,*" Journal of Knowledge-Based Systems, vol. 15, pp. 515 - 528.

Index