Nikhil R. Pal and Lakhmi Jain (Eds)

Advanced Techniques in Knowledge Discovery and Data Mining

With 101 Figures

Springer

Nikhil R. Pal
Electronics and Communication Sciences Unit, Indian Statistical Institute, India

Lakhmi Jain
KES Center, University of South Australia, Australia

Series Editors
Xindong Wu
Lakhmi Jain

British Library Cataloguing in Publication Data
Advanced techniques in knowledge discovery and data mining.
— (Advanced information and knowledge processing)
1. Data mining
I. Pal, Nikhil R. II. Jain, L. C.
006.3'12

ISBN 1852338679

Library of Congress Cataloging-in-Publication Data
Advanced techniques in knowledge discovery and data mining / Nikhil R. Pal and Lakhmi Jain (eds.).
p. cm. — (Advanced information and knowledge processing)
Includes bibliographical references and index.
ISBN 1-85233-867-9 (alk. paper)
1. Knowledge acquisition (Expert systems) 2. Data mining. I. Pal, Nikhil R. II. Jain, L. C. III. Series.
QA76.76.E95A335 2004
006.3'31—dc22 2004050953

Apart from any fair dealing for the purposes of research or private study, or criticism or review, as permitted under the Copyright, Designs and Patents Act 1988, this publication may only be reproduced, stored or transmitted, in any form or by any means, with the prior permission in writing of the publishers, or in the case of reprographic reproduction in accordance with the terms of licences issued by the Copyright Licensing Agency. Enquiries concerning reproduction outside those terms should be sent to the publishers.

AI&KP ISSN 1610-3947

ISBN 1-85233-867-9
Springer Science+Business Media
springeronline.com

© Springer-Verlag London Limited 2005

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

The publisher makes no representation, express or implied, with regard to the accuracy of the information contained in this book and cannot accept any legal responsibility or liability for any errors or omissions that may be made.

Typesetting: Electronic text files prepared by editors
Printed and bound in the United States of America
34/3830-543210 Printed on acid-free paper SPIN 10966099

Advanced Information and Knowledge Processing

Series Editors
Professor Lakhmi C. Jain
Xindong Wu

Also in this series

Gregoris Mentzas, Dimitris Apostolou, Andreas Abecker and Ron Young
Knowledge Asset Management
1-85233-583-1

Michalis Vazirgiannis, Maria Halkidi and Dimitrios Gunopulos
Uncertainty Handling and Quality Assessment in Data Mining
1-85233-655-2

Asunción Gómez-Pérez, Mariano Fernández-López, Oscar Corcho
Ontological Engineering
1-85233-551-3

Arno Scharl (Ed.)
Environmental Online Communication
1-85233-783-4

Shichao Zhang, Chengqi Zhang and Xindong Wu
Knowledge Discovery in Multiple Databases
1-85233-703-6

Jason T.L. Wang, Mohammed J. Zaki, Hannu T.T. Toivonen
and Dennis Shasha (Eds)
Data Mining in Bioinformatics
1-85233-671-4

C.C. Ko, Ben M. Chen and Jianping Chen
Creating Web-based Laboratories
1-85233-837-7

Manuel Graña, Richard Duro, Alicia d'Anjou and Paul P. Wang (Eds)
Information Processing with Evolutionary Algorithms
1-85233-886-0

Colin Fyfe
Hebbian Learning and Negative Feedback Networks
1-85233-883-0

Yun-Heh Chen-Burger and Dave Robertson
Automating Business Modelling
1-85233-835-0

Dirk Husmeier, Richard Dybowski and Stephen Roberts (Eds)
Probabilistic Modeling in Bioinformatcs and Medical Informatics
1-85233-778-8

Ajith Abraham, Lakhmi Jain and Robert Goldberg (Eds)
Evolutionary Multiobjective Optimization
1-85233-787-7

K.C. Tan, E.F. Khor and T.H. Lee
Multiobjective Evolutionary Algorithms and Applications
1-85233-836-9

Nikhil R. Pal and Lakhmi Jain (Eds)
Advanced Techniques in Knowledge Discovery and Data Mining
1-85233-867-9

Contents

Preface

Data mining and knowledge discovery (DMKD) is a rapidly expanding field in computer science. It has become very important because of an increased demand for methodologies and tools that can help the analysis and understanding of huge amounts of data generated on a daily basis by institutions like hospitals, research laboratories, banks, insurance companies, and retail stores and by Internet users. This explosion is a result of the growing use of electronic media.

But what is data mining (DM)? A Web search using the Google search engine retrieves many (really many) definitions of data mining. We include here a few interesting ones.

One of the simpler definitions is: "As the term suggests, data mining is the analysis of data to establish relationships and identify patterns" [1]. It focuses on identifying relations in data. Our next example is more elaborate:

An information extraction activity whose goal is to discover hidden facts contained in databases. Using a combination of machine learning, statistical analysis, modeling techniques and database technology, data mining finds patterns and subtle relationships in data and infers rules that allow the prediction of future results. Typical applications include market segmentation, customer profiling, fraud detection, evaluation of retail promotions, and credit risk analysis [2].

This one suggests that data mining tries to find useful "information" from data that can help predict the future. These definitions do not explicitly emphasize a large volume of data, an issue in the next definition: "The process of analyzing large amounts of data in order to extract new kinds of useful information (such as implicit relationships between different pieces of information)" [3].

Based on these definitions one may get the impression that data mining is just pattern recognition (PR). In our view, traditional pattern recognition is a subset of data mining. We shall elaborate on our views later. All these definitions focus on finding useful relations. Traditional pattern recognition does the same. So what is the difference between PR and DM? In PR we typically deal with data sets of moderate size, whereas in a typical DM application, we are concerned with data sets that are large in terms of dimension and number of clusters. And the large size has resulted in new challenges. For example, many traditional clustering algorithms, such as c-means or their fuzzy and other relatives, become computationally infeasible for very large data sets. Similarly, most traditional analysis techniques are difficult to use with very large-dimensional (on the order of thousand) data sets. In traditional PR, data are typically collected with some goal in mind and we know the questions we are trying to answer using the data. But in many data mining applications, data are collected without any specific goal in mind and the miner tries to find "interesting information" from. Because of the explosion in the use of electronic media, some new exploratory data analysis problems have arisen. For example, analyzing patterns of Web access, try to create user profiles that can help e-marketing and other applications (known as web mining). This is a new type of problem, although one can argue that it is a clustering problem!

The term knowledge discovery (KD) is often used with data mining. Fayyad et al. [4] defined KD as the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. Knowledge discovery may not be viewed as synonymous with data mining, but they are intimately related. We may look at them as follows: data mining is the cause and knowledge discovery is the effect (results) – mining the data leads to knowledge discovery.

Some of the important problems that the DM and KD deal with are: rule extraction, identification of association, feature analysis, linguistic summarization, clustering, classifier design, and novelty/anomaly detection.

Typically, we want DM algorithms that are scalable and whose outcome (knowledge) is interpretable. DM problems can be solved using statistical techniques computational intelligence (CI) tools such as neural networks (NN), fuzzy logic, rough sets, and genetic algorithms (GA). CI tools have some advantages, for example, fuzzy systems are easy to understand and so are neuro-fuzzy systems. Neural networks are adaptive systems that posses parallelism, which could be useful for dealing with large data sets. Evolutionary algorithms, such as GA, can be used to solve difficult optimization problems, and genetic programming (GP) can be used to find programs that can solve a given problem.

This volume gives a balanced mixture of both traditional- and computational-intelligence-based techniques for data mining and knowledge discovery. It deals with both methodologies and applications ranging from the semiconductor industry to medical diagnosis.

The opening chapter, “Trends in Data Mining and Knowledge Discovery” by Cios and Kurgan, provides a nice introduction to recent trends in data mining and knowledge discovery. Huge amount of data are readily available in various databases, warehouses, and data repositories. Researchers and the business and industrial communities are interested in clear and simple methodologies for extracting the knowledge hidden in the data. In Chapter 1, the authors discuss an integrated DMKD process model based on emerging technologies like XML, PMML, SOAP, UDDI, and OLE BD-DM. These technologies help us to design flexible, semiautomated, easy-to-use DMKD models. They enable us to build knowledge repositories and they allow for communication among data mining tools, databases, and knowledge repositories. They also enable integration and automation of DMKD tasks. The various technologies described in the chapter will play an important role in the design of next-generation DMKD systems. The six-step process model is expected to help readers and practitioners, particularly in the business and industrial communities, to design data mining and knowledge discovery systems.

Modern semiconductor manufacturing processes feature an increasing number of processing steps with an increasing complexity of the steps themselves. They generate a flood of multivariate monitoring data. This exponentially increasing complexity and the associated information processing and productivity impose stringent requirements, which are difficult to meet with state-of-the-art monitoring and analysis methods and tools. We cannot overemphasize that the optimization of the manufacturing processes in the semiconductor industry is important and has a significant economic impact. The next chapter, “Advanced Methods for the Analysis of Semiconductor Manufacturing Process Data” by Konig and Gratz,

presents several methods for detecting anomaly or novelty and discovery of nonobvious multivariate dependencies of the parameters involved in a semiconductor manufacturing process. Such discovery of knowledge can be used to significantly improve process control. Data visualization is very effective for anomaly and novelty detection. In this context, the authors talk about several interesting tools for dimensionality reduction and data visualization. The authors emphasize applications of soft computing tools, such as neural networks, for the problem at hand. This chapter provides a very useful taxonomy of dimensionality reduction methods and presents a wide spectrum of visualization tools for high-dimensional data. It is clearly demonstrated how such tools can be used for knowledge discovery. Although they exhibit specific applications to semiconductor manufacturing, the methods are applicable to many other applications that demand knowledge discovery.

One of the major applications of data mining to the retail industry is transaction analysis, including clustering of market baskets. It poses a very challenging problem because the dimensionality of the data is very high; sometimes it can have thousands of features and often the data are sparse. Moreover, this particular application demands easily interpretable and actionable results. Obviously the "curse of dimensionality" is an important issue while clustering such data sets. Ghosh and Strehl present methods for clustering and visualization of retail market baskets in Chapter 3 using a relational approach. The high-dimensional data are converted into a similarity relation and the rest of the work, such as clustering and visualization, is done using the relation, thereby avoiding dealing with the high-dimensional data. As we mentioned earlier, that visualization is very important for mining and knowledge discovery; the authors use the clustering result to reorder the data to produce useful two-dimensional visual displays of the similarity relation. They use efficient and scalable graph-partitioning-based clustering techniques on the computed relation. The visual display produced by the algorithm can be used very effectively as a cluster validity tool. The OPOSSUM (Optimal Partitioning of Sparse Similarities Using Metis) algorithm is a similarity-based clustering technique tailored to market basket data. It differs from other graph-based clustering techniques by application-driven balancing of clusters (which is very important) and visualization-driven heuristics for finding an appropriate value for the number of clusters. The visualization tool CLUSION gives a relationship-centered view, as contrasted with common projective techniques, such as the selection of dominant features or optimal linear projections, which are object-centered. In CLUSION, the actual features are transparent, and all pairwise relationships are displayed.

The detection of changes in a time series is a difficult but important problem that can also be viewed as a kind of novelty detection. Typically data mining algorithms assume that the historic data are the best estimator of what is going to happen in the future. But this is not necessarily true. As more data are accumulated in a database, we need to examine whether the new data agree with the model induced from previous instances. This is the theme of Chapter 4 by Zeira et al. Once all points associated with changes are detected, a data stream can be represented as a series of nonoverlapping segments. The authors present a new methodology for change detection and segmentation based on a set of statistical

estimators. Unlike traditional segmentation methods, which typically analyze univariate time series, the method proposed here detects statistically significant changes in incrementally built classification models of data mining. The authors demonstrate the utility of the method on real-world data sets from two distinct domains, education and finance. The quality of segmentation obtained by the proposed method is compared with that of alternative segmentations.

Instance/prototype selection is very important for knowledge discovery. Data mining and knowledge discovery situations typically involve very large data sets. This makes data reduction a very important step in DMKD. Extraction of prototypes is also important because prototypes are typical objects, so they can easily be converted to rules. Chapter 5, "Instance Selection Using Evolutionary Algorithms: An Experimental Study" by Cano et al., deals with these issues. They present an interesting study on the performance of four representative evolutionary algorithm models for prototype selection and the training set selection for data reduction in knowledge discovery. The stratified approach to training set selection appears to be an effective scheme when the data set size is very large. The authors have done an excellent job of comparing these algorithms with other nonevolutionary instance selection algorithms. The evolutionary instance selection algorithms are found to consistently outperform (in terms of higher data reduction and better classification accuracy) the nonevolutionary ones considered in this chapter.

Chapter 6 by Wong et al. also deals with evolutionary algorithms, but for learning Bayesian networks more efficiently. Bayesian networks, formal knowledge representation tools, support reasoning under uncertainty and have many applications including data mining, information retrieval, and various diagnostic systems. Bayesian networks are useful, but automatic construction of the network from data is a difficult problem. The authors use cooperative coevolution for this problem. One of the main criticisms of evolutionary computation for learning is that it is very slow. The authors propose a hybrid framework, the cooperative coevolution genetic algorithm (CCGA), for Bayesian network learning. The proposed learning algorithm has two phases: the conditional independence test (CIT) phase and the search phase. In the CIT phase, a dependency analysis is done to reduce the search space. In the search phase, model searching is done using cooperative coevolution. These two phases make the algorithm more efficient. Comparison with other algorithms demonstrates that the proposed algorithm performs faster than other algorithms. Also CCGA is found to discover better (more accurate) Bayesian networks.

Extraction of human interpretable rules from medical databases is an important and challenging problem of knowledge discovery and data mining. It is a challenging task because of deficits in patients' medical records and the existence of contradictory information. In Chapter 7, Ichimura et al. describe three methods of extracting *if-then* rules from neural networks. In particular they propose a knowledge-based artificial neural network (KBANN) with structure level adaptation (SLA) of NN. This method can determine network structure to represent knowledge structure of medical experts. The authors also describe a genetic programming–based rule extraction scheme using automatically defined groups (ADG). The ADG extracts *if-then* rules by generating cooperative

behaviors of agents. It is then applied to obtain a diagnostic system for hepatobiliary disorders. The third approach is based on immune multiagent neural networks (IMANN). The basic idea behind the IMANN is the cooperation and competition mechanism found in biological immune systems. It is an interesting architecture that combines the ideas of multiagent system and neural networks and makes a two-stage classification by planar lattice neural networks (PLNN) and Darwinian neural networks (DNN). The effectiveness of the system is demonstrated using the intensive care unit (ICU) database.

In recent times an enormous amount of satellite data have been routinely generated, posing a great challenge to researchers and practitioners. Extraction of rules from such databases is difficult because satellite images usually contain many complex structures. Consequently, a high recognition rate is not easy to attain. Lin et al. in Chapter 8 discuss a hybrid neural fuzzy network that combines unsupervised and supervised learning to design classifier systems. Although with higher resolution the visual quality of the images becomes better, the analysis of the same becomes more difficult because pixel values in isolation do not contain much discriminative power. So authors take a feature-based approach that combines spatial, statistical, and spectral features. The authors also devised a mechanism to reduce the computational complexity of the cooccurrence matrix. The cascaded system uses Kohonen's self-organizing feature map for dimensionality reduction. The lower-dimensional inputs are then classified by a neural fuzzy network (called SONFIN). Such a system has many applications, including land-cover classification, crop yield estimation, soil erosion, and land usage. The proposed system is a novel general algebraic system identification method, which can be used for knowledge discovery from data in many applications. One of the main advantages of this system over other neural networks is that it is interpretable in terms of fuzzy *if-then* rules.

The last chapter, by Tsumoto, presents knowledge discovery methods using rough sets for medical applications. One of the important problems associated with rule induction methods is that the extracted rules may not represent information exactly the same way an expert represents and uses it for decision making. To deal with this problem, the author discusses the characteristics of medical reasoning. The concept of positive and negative rules is introduced and their importance is emphasized. Two search algorithms are proposed for induction of positive and negative rules. The proposed rule induction method is evaluated on several medical databases. Authors have clearly demonstrated that these rough sets–based schemes can extract rules that correctly represent experts' knowledge. They also discovered several interesting patterns. In this context, the characteristics of two measures, classification accuracy and coverage, are discussed.

We are grateful to the authors and reviewers for their contributions. Thanks are due to Feng-Hsing Wang for his help during the preparation of the manuscript. The editorial help by the Springer is acknowledged.

References

[1] practice.findlaw.com/glossary.html

[2] www.twocrows.com/glossary.htm

[3] www.rlg.org/redlightgreen/glossary.html

[4] Fayyad, U. M.; Piatetsky-Shapiro, G.; and Smyth, P., From data mining to knowledge discovery: An overview, in *Advances in Knowledge Discovery and Data Mining*, AAAI Press and the MIT Press, chapter 1, 1–34, 1996.

1. Trends in Data Mining and Knowledge Discovery

Krzysztof J. Cios[1,2,3] and Lukasz A. Kurgan[4]

[1] University of Colorado at Denver and Health Sciences Center, Department of Computer Science and Engineering, Campus Box 109, Denver, CO 80217-3364, U.S.A.; email: Krys.Cios@cudenver.edu

[2] University of Colorado at Boulder, Department of Computer Science, Boulder, CO, U.S.A.;

[3] 4cData, LLC, Golden, CO 80401

[4] University of Alberta, Department of Electrical and Computer Engineering, ECERF 2nd floor, Edmonton, AB T6G 2V4, Canada; email: lkurgan@ece.ualberta.ca

Data mining and knowledge discovery (DMKD) is a fast-growing field of research. Its popularity is caused by an ever increasing demand for tools that help in revealing and comprehending information hidden in huge amounts of data. Such data are generated on a daily basis by federal agencies, banks, insurance companies, retail stores, and on the WWW. This explosion came about through the increasing use of computers, scanners, digital cameras, bar codes, etc. We are in a situation where rich sources of data, stored in databases, warehouses, and other data repositories, are readily available but not easily analyzable. This causes pressure from the federal, business, and industry communities for improvements in the DMKD technology. What is needed is a clear and simple methodology for extracting the knowledge hidden in the data. In this chapter, an integrated DMKD process model based on technologies like XML, PMML, SOAP, UDDI, and OLE BD-DM is introduced. These technologies help to design flexible, semiautomated, and easy-to-use DMKD models to enable building knowledge repositories and allowing for communication between several data mining tools, databases, and knowledge repositories. They also enable integration and automation of the DMKD tasks. This chapter describes a six-step DMKD process model and its component technologies.

1.1 Knowledge Discovery and Data Mining Process

Knowledge discovery (KD) is a nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns from large collections of data [30]. One of the crucial KD steps is a data mining (DM) step. DM is concerned with the actual extraction of knowledge from data, in contrast to the KD process, which is concerned with many other activities. We want to stress this distinction, although people often use the terms DM, KD and DMKD as synonymous.

The design of a framework for a knowledge discovery process is an important issue. Several researchers described a series of steps that constitute the KD process; they range from very simple models, incorporating few steps that usually include data collection and understanding, data mining, and implementation, to more sophisticated models like the nine-step model proposed by Fayyad et al. [31]. In this chapter we describe the six-step DMKD process model [18], [19]. The advantage of this model is that it is based on the industry-initiated study that led to the development of an industry- and tool-independent DM process model [26] and has been successfully used in medical applications [18], [44], [47], [61].

1.1.1. XML: Key to Unlocking Data Mining and Knowledge Discovery

One of the technologies that can help in carrying out the DMKD process is XML (eXtensible Markup Language); a standard proposed by the WWW Consortium [12]. It is a subset of SGML that uses custom-defined tags [42]. XML allows for description and storage of structured or semistructured data and their relationships. One of the most important features of XML is that it can be used to exchange data in a platform-independent way. XML is easy to use with many off-the-shelf tools available for automatic processing of XML. From the DMKD process point of view XML is a crucial technology as it helps to:

- standardize communication between diverse DM tools and databases. This may result in a new generation of DM tools that can communicate with a number of different database products.
- build standard data repositories that share data between different DM tools and work on different software platforms. This may help to consolidate the DMKD market and open it for new applications.
- implement communication protocols between the DM tools. This may result in development of DM toolboxes [45] that consist of different DM tools, developed by different companies, but that are able to communicate and provide protocols to extract consolidated, more understandable, accurate, and easily applicable knowledge.
- provide a framework for integration of and communication between different DMKD steps. For instance, the information collected during the domain and data understanding steps can be stored as XML documents. They can then be used in the data preparation and data mining steps as a source of information that can be accessed automatically, across platforms and across tools. In addition, the extracted knowledge can be stored using XML and PMML (Predictive Model Markup Language) documents. This may enable automation of sharing of discovered knowledge between diverse domains and tools that use it, as long as they are XML- and PMML-compliant.

Because the DMKD is a very complex process, which includes DM as one of its steps, the importance of XML's utility in automating and consolidating the DMKD process cannot be overstated; it makes it platform- and tool-independent. A number of other XML goals defined by the W3C, like support of a variety of

applications, ease of writing programs that process XML documents, human-comprehensibility of XML documents, quick design of XML documents, all support usefulness of the XML technology. XML is currently revolutionizing a number of fields, including the DMKD process.

In this chapter we describe a six-step DMKD process, provide examples, and discuss its relation to other DMKD models. We also describe new technologies like XML, XML-RPC, PMML, SOAP, UDDI, OLE BD-DM, and DM methods and tools. The differences between the methods and tools of the DMKD process are also discussed.

1.1.2. Why Data Mining and Knowledge Discovery?

DMKD history goes back to 1989 IJCAI Workshop on Knowledge Discovery in Databases (KDD) [54]. The workshops continued until 1994; in 1995 the International Conference on Knowledge Discovery and Data Mining became the most important event for the DMKD community [55], [30]. DMKD conferences like ACM SIGKDD, SPIE, PKDD, and SIAM, and journals like *Data Mining and Knowledge Discovery Journal* (started in 1997), *Journal of Knowledge and Information Systems* (started in 1999), and *IEEE Transactions on Knowledge and Data Engineering* (started in 1989) have also become an integral part of the DMKD field.

It is not easy to describe the current status of the DMKD field because it changes very quickly. We attempt to describe it using Web-based online research service Axiom® [6], which provides access to INSPEC, Compendex®, PageOne™, and the Derwent World Patents Index databases. It finds research papers using a user-specified set of keywords, and a time frame. To analyze the DMKD field we performed several queries, which are summarized in Figures 1.1 and 1.2.

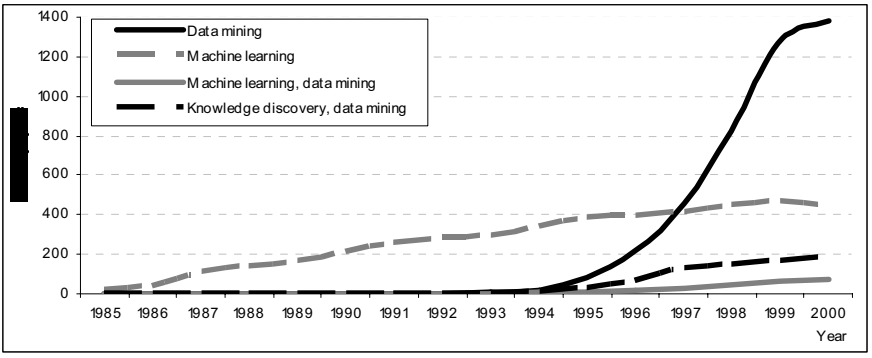
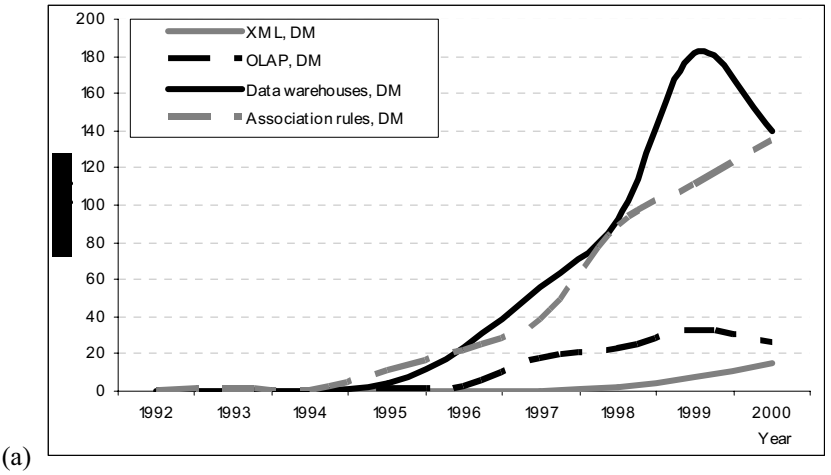
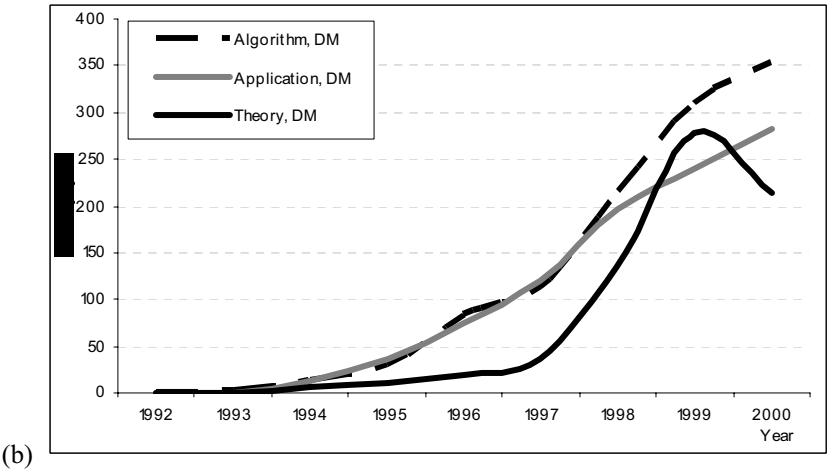


Fig. 1.1. Evolution of data mining and data mining and knowledge discovery fields.



(a)



(b)

Fig. 1.2. (a) Trends in data mining, (b) data mining theory and applications.

As said above, the DM revolution started in the mid-1990's. It was characterized by fast growth, as evidenced by the increase over a 5-year period in the number of DM papers from about 20 to about 1270. One of the reasons for that growth was the incorporation of existing tools and algorithms into the DM framework. Many DM tools, like machine learning (ML), were already well established. Figure 1.1 shows the number of ML papers in the context of DM papers. The number of papers covering both ML and DM grew slowly; in 2000 there were 74 such papers, which constituted 6% of the entire DM research and in 2000 it constituted 15%. The data are misleading, however, since many people still treat DM and DMKD as being the same.

Recent trends in DMKD include OLAP, data warehousing, association rules, high-performance DMKD systems, visualization techniques, and applications of DM. The first three are summarized in Fig. 1.2(a). The research interest in association rules follows a pattern generally similar to that of DM. On the other

hand, the research in OLAP (on-line analytical processing) and data warehouses peaked around 1999. Some of the trends that initially had the greatest impact on the DM field began to decline because most of the issues concerned with those areas may have been solved, and thus the attention shifted toward new areas and applications. Moreover, new trends emerged that have great potential to benefit the DMKD field, like XML and related technologies, database products that incorporate DM tools, and new developments in the design and implementation of the DMKD process. Among these, XML technology may have the greatest influence since it helps to tie DM with other technologies like databases or e-commerce. XML can also help to standardize the I/O procedures, which will help to consolidate the DM market and carry out the DMKD process, see Fig. 1.2(a).

The other important DMKD issue is the relationship between theoretical DM research and DM applications; see Fig. 1.2(b). The number of DM application papers increased rapidly over the last few years. The growth rate of theoretical research was slower initially but accelerated around 1998 and then started to slow down. This trend may be interpreted that more attention has been given to practical applications, possibly because of the increased funding levels. This situation, however, calls for a more balanced approach because applications need to be well grounded in theory if real progress is to be made. We need new DM tools that can handle huge amounts of textual data generated by the Internet, tools to extract knowledge from hypertext and images as often encountered in biology and medicine.

In short, the DMKD is an exponentially growing field with strong emphasis on applications.

1.2 Six-Step Knowledge Discovery and Data Mining Process

The goal of designing a DMKD process model is to come up with a set of processing steps that can be followed by practitioners when they execute their DMKD projects. Such a process model should help to plan, work through, and reduce the cost by detailing procedures to be performed in each of the steps. The DMKD process model should provide a complete description of all the steps, from problem specification to deployment of the results.

A useful DMKD process model must be validated in real-life applications. One such initiative was taken by the CRISP-DM (CRoss-Industry Standard Process for Data Mining) group [72], [26]. Their design was based on the study supported by several European companies (automotive, aerospace, telecommunication, consultancy, insurance, data warehouse, developer of DM tools). The project included two inseparable ingredients of any DMKD process: databases and DM tools. Two companies (OHRA and DaimlerChrysler) provided large-scale applications to validate the DMKD process model. The goal of the project was to develop a DMKD process that would help to save project costs, shorten project time, and adopt DM as a core part of the business. As a result, the six-step DM process was developed: business understanding, data understanding, data

preparation, modeling, evaluation, and deployment. They called the entire process a data mining process, which is different from the term's understanding in the United States.

Another six-step DMKD process model [18], described below, is based on the CRISP-DM model but differs from it in the following:

- the entire process is called the DMKD process, thus resolving the confusing use of the term DM (it is just a step, not the process).
- the DM step is used instead of the modeling step. The DM step is concerned with modeling and actual extraction of knowledge from data.
- several new feedback mechanisms were introduced. The CRISP-DM model has only three major feedback sources but our experience shows that the new feedback mechanisms are as important as the three [18], [44], [47], [61]; see Fig. 1.3.
- the model is able to communicate with other domains; the knowledge discovered for a domain may be applied to other domains.

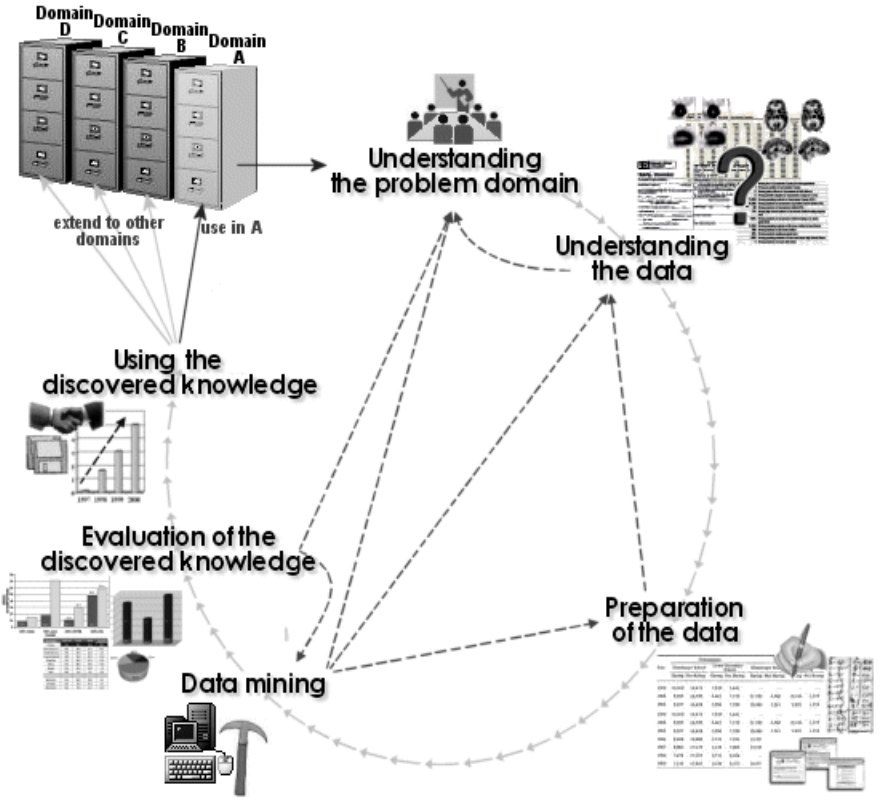


Fig. 1.3. The six-step DMKD process model.

1. **Understanding the problem domain**
   In this step one works closely with domain experts to define the problem and determine the project goals, identify key people, and learn about current solutions to the problem. It involves learning domain-specific terminology. A description of the problem, including its restrictions, is done. The project goals must be translated into the DMKD goals and may include initial selection of potential DM tools.
2. **Understanding the data**
   This step includes collection of sample data and deciding which data will be needed, including its format and size. If background knowledge exists, some attributes may be ranked as more important. Next, we need to verify usefulness of the data in respect to the DMKD goals. Data need to be checked for completeness, redundancy, missing values, plausibility of attribute values, and the like.
3. **Preparation of the data**
   This is the key step on which the success of the entire knowledge discovery process depends; it usually consumes about half of the entire project effort. In this step, we decide which data will be used as input to the data mining tools in Step 4. It may involve sampling of data, running correlation and significance tests, cleaning data like checking for completeness of data records and correcting for noise. The cleaned data can be further processed by feature selection and extraction algorithms (to reduce dimensionality), by derivation of new attributes (say by means of discretization), and by summarization of data (data granularization). The result is new data records, meeting specific input requirements for the planned, to-be-used DM tools.
4. **Data mining**
   This is another key step in the knowledge discovery process. Although it is the data mining tools that discover new information, their application usually takes less time than data preparation. This step involves usage of the planned data mining tools, and selection of the new ones if needed. Data mining tools include many types of algorithms, such as rough and fuzzy sets, Bayesian methods, evolutionary computing, machine learning, neural networks, clustering, and preprocessing techniques. Detailed descriptions of these algorithms and their applications can be found in [17]. Description of data summarization and generalization algorithms can be found in [22]. This step involves the use of several DM tools on data prepared in Step 3. First, however, the training and testing procedures need to be designed and the data model is constructed using one of the chosen DM tools; the generated data model is then verified using testing procedures.

   One of the major difficulties in this step is that many commonly used tools may not scale up to a huge volume of data. Scalable DM tools are characterized by a linear increase of their run time with the increase of the number of data points within a fixed amount of available memory. Most of the DM tools are not scalable but there are examples of tools that scale well with the size of the input data: clustering [11], [32], [78]; machine learning [34], [63]; association rules [2], [3], [70]. An overview of scalable DM tools

is given in [33]. One approach for dealing with scalability of DM tools is connected with the notion of meta-mining. Meta-mining generates meta-knowledge from knowledge generated by data mining tools [67]. It is done by dividing data into subsets, generating data models for these subsets, and generating meta-knowledge from these data models. In this approach small data models are processed as input data instead of huge amounts of the original data, which greatly reduces computational overhead [46], [48].

5. Evaluation of the discovered knowledge

This step includes understanding the results by owners of the data who check whether the new information is truly novel and interesting, and checking the impact of the discovered knowledge. Only the approved models (results of applying many data mining tools and preprocessing methods) are kept. The entire DMKD process may be revisited to identify which alternative actions could be taken to improve the results.

6. Using the discovered knowledge

This step is entirely in the hands of the owners of the database. It consists of planning where and how the discovered knowledge will be used. The application area in the current domain should be extended to other domains within an organization. A plan to monitor the implementation of the discovered knowledge should be created and the entire project documented.

The six-step DMKD process model described above is visualized in Fig. 1.3. Important parts of the process are its iterative and interactive aspects. The feedback loops are necessary since any changes and decisions made in one of the steps can result in changes in later steps. The model uses several such feedback mechanisms:

- from Step 2 to Step 1 because additional domain knowledge may be needed to better understand the data.
- from Step 3 to Step 2 because additional or more specific information about the data may be needed before choosing specific data preprocessing algorithms (for instance, data transformation or discretization).
- from Step 4 to Step 1 when the selected DM tools do not generate satisfactory results, and thus the project goals must be modified.
- from Step 4 to Step 2 in a situation when data was misinterpreted, causing the failure of a DM tool (e.g., data were misrecognized as continuous and discretized in Step 3). The most common scenario is when it is unclear which DM tool should be used because of poor understanding of the data.
- from Step 4 to Step 3 to improve data preparation because of the specific requirements of the used DM tool, which may not have been known during the data preparation step.
- from Step 5 to Step 1 when the discovered knowledge is not valid. There are several possible sources of such a situation: incorrect understanding or interpretation of the domain or incorrect design or understanding of problem restrictions, requirements, or goals. In these cases the entire DMKD process needs to be repeated.
- from Step 5 to Step 4 when the discovered knowledge is not novel, interesting,

or useful. In this case, we may choose different DM tools and repeat Step 4 to extract new and potentially novel, interesting, and thus useful knowledge.

The feedback paths are shown as dashed lines in Fig. 1.3. It should be understood that the described feedback paths are by no means exhaustive.

Table 1.1 compares the six-step DMKD process model with the nine-step DMKD process model [31] and the five-step model [16].

The common steps for the three models are domain understanding, data mining, and evaluation of the discovered knowledge. The nine-step (Fayyad's) model is very detailed, and although it provides the most guidance, it performs Steps 5 and 6 too late in the process. We think that these steps should be performed during the steps of understanding the domain and understanding the data, to guide the process of data preparation. In other words, the goal of data preparation is to prepare the data to be used with the already chosen DM tools, while their model suggests that the DM tool is selected in Step 6, depending on the outcome of data preparation. This may cause problems when choosing a DM tool because the prepared data may not be suitable for the given tool. Thus, an unnecessary feedback loop may be needed to change data preparation in Steps 2, 3, and 4. The five-step (Cabena's) model is very similar to the six-step (Cios's) model, except that the data understanding step is missing. The incompleteness of the Cabena model was pointed out in [39] where the author used it in a business domain, and one of the conclusions was the necessity of adding one more step

Table 1.1. Comparison of three DMKD process models.

6 Step DMKD Process [18]	**9 Step DMKD Process** [31]	**5 Step DMKD Process** [16]
1. Understanding the domain	1. Understanding application domain, identifying the DMKD goals	1. Business objective determination
2. Understanding the data	2. Creating target data set	2. Data preparation
3. Preparation of the data	3. Data cleaning and preprocessing	
	4. Data reduction and projection	
	5. Matching goal to particular data mining method	
	6. Exploratory analysis, model and hypothesis selection	
4. Data mining	7. Data mining	3. Data mining
5. Evaluation of the discovered knowledge	8. Interpreting mined patterns	4. Analysis of results
6. Using the discovered knowledge	9. Consolidating discovered knowledge	5. Knowledge assimilation

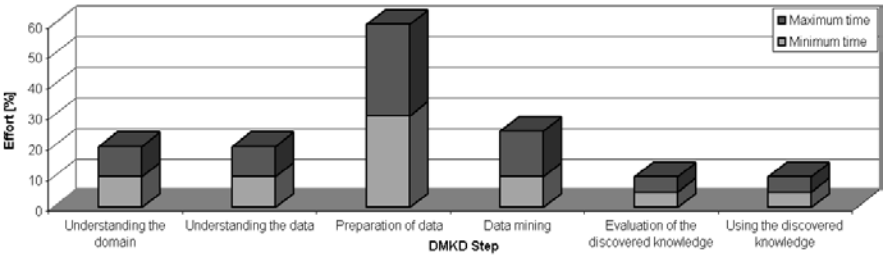


Fig. 1.4. Relative effort spent on each of the DMKD steps.

between data preparation and data mining, which he called data audit. The six-step model has the advantage of being similar to the CRISP-DM model that was validated on large business applications. The model has also been used in several projects like a system for diagnoses of SPECT bulleye images [18], creating and mining a database of cardiac SPECT images [61], creating an automated diagnostic system for cardiac SPECT images [44], and mining clinical information concerning cystic fibrosis patients [47].

The important characteristic of the DMKD process is relative time spent on completing each of the steps. Reference [16] estimates that about 20% of the effort is spent on business objective determination, about 60% on data preparation, and about 10% for data mining and analysis of knowledge and knowledge assimilation steps, respectively. On the other hand, [10] shows that about 15 to 25% of the project time is spent on the DM step. Usually it is assumed that about 50% of the project effort is spent on data preparation. There are several reasons why this step requires so much time: data collected by enterprise companies consist of about 1 to 5% errors, often the data are redundant (especially across databases) and inconsistent, also companies may not collect all the necessary data [57]. These serious data quality problems contribute to the extensive requirements for data preprocessing step. In a study at a Canadian fast-food company [39], it was shown that the DM step took about 45% of the total project effort, while data preparation took only about 30%. Thus, it is better to use time ranges rather than fixed times for estimating the steps requirements, see Fig. 1.4.

A very important issue is how to carry out the DMKD process without extensive background knowledge, without manual data manipulation, and without manual procedures to exchange data between different DM applications. The next two sections describe technologies that may help in automating the DMKD process, thus making its implementation easier.

1.3 New Technologies

Automating or, more realistically, semiautomating the DMKD process is a very complex task. User input is always necessary to perform the entire DMKD task because only domain experts have the necessary knowledge about the domain and data. In addition, evaluation of the results, at each DMKD step, is needed. To

semiautomate the DMKD process several technologies are necessary:

- a data repository that stores the data, background knowledge, and models;
- protocols for data and information exchange between data repositories and DM tools and between different DM tools; and
- standards for describing the data and models.

XML technology has been studied and used extensively over the last years, along with other technologies built on top of XML, like PMML, XML-RPC, SOAP, and UDDI. Together they can provide solutions to the problem of semiautomating the DMKD process. In what follows, these technologies are introduced and their applications within the DMKD process are described. In addition, technologies like OLAP and OLE-DB DM and their impact on DMKD process are also discussed.

1.3.1. XML

XML is a markup language for documents that contain structured information. Structured information consists of content (numbers, character strings, images, etc.) and information of what role that content plays, i.e., context of the information (e.g., a rule is built out of selectors, and a selector is a pair of attributes (name and value)). XML defines a standard to add markup or to identify structures in documents.

XML is primarily used to create, share, and process information. XML enables users to define tags (element names) that are specific to a particular purpose. XML tags are used to describe the meaning or context of the data in a precisely defined manner. It is the information modeling features of XML that made it popular. Thanks to these features, processing of XML documents can be performed automatically.

XML technology is widely used in industry to transfer and share information. One of the most important properties of XML is that the current database management systems (DBMS) support the XML standard. From the DMKD point of view this means that XML can be used as a transport medium between DM tools and XML-based knowledge repositories, which are used to store discovered knowledge and information about the data and the DBMS that store the data.

There are two major kinds of DBMS that can handle XML documents: XML-native DBMS, and XML-enabled DBMS:

- The majority of *XML-native DBMS* are based on the standard DB physical storage model, like relational, object-relational, or object-oriented, but they use XML documents as the fundamental storage unit, just as relational DBMS uses tuples as its fundamental storage unit. Their main advantage lies in the possibility of storing an XML document and then retrieving the same document without losing any information, both on structural and data levels (not yet possible using the XML-enabled DBMS). The two well-known XML-native DBMS are: Lore [49] and Tamino [62]. XML-native DBMSs can be divided into two groups: created over the relational model (examples include DBDOM, eXist, Xfinity, and XML Agent) and created over the

object-oriented model (examples include eXcelon, X-Hive, and ozone). There are also XML-native DBMS, that are not built on either relational or object-oriented models. They are schema-independent, information-centric, and characterized by treating context as flexibly as the data. Example of such a DBMS is the NeoCore's XML database [50].

- The *XML-enabled DBMS* incorporates the XML document into the traditional database technology. Examples of commercial XML-enabled DBMSs (all use the relational model) are: Oracle 8i [7], DB2 [23], Informix [41], Microsoft SQLServer 2000 [68], and Microsoft Access2002 [74]. Because these systems are used on a large scale in the business world they may become a dominant method for storing XML documents.

  However, there are several problems associated with using XML-enabled DBMS. First, the existing models of storing XML documents do not fully preserve the context of the XML documents (e.g., the order of tags is lost). Second, some content, like comments or processing instructions of the XML document, is also lost. In contrast, any native XML DBMS preserves that information. The researchers already developed some solutions to this problem by proposing new schemas for storing XML documents within both relational and object-relational DBMS, which either use [9], [64], or do not use the Document Type Definition (DTD) documents [65].

Another advantage of XML is the ability to query it to retrieve and manipulate data stored in the document. A number of query languages have been developed, including Lorel [1], Quilt [20], UnQL [14], XDuce [40], XML-QL [27], XPath [24], XQL [60], Xquery [21], and YaTL [25]. XPath and Xquery are two query languages that received the W3C recommendation.

1.3.2. XML-RPC

XML-RPC (XML-Remote Procedure Call) is a protocol that allows software running on disparate operating systems and in different environments to make procedure calls over the Internet [75]. It uses HTTP as the transport and XML for the encoding. XML-RPC is designed to be as simple as possible to allow for the transmittal, processing, and return of complex data structures. XML-RPC implementations are available for virtually all operating systems, programming languages, dynamic and static environments, which include implementations in Perl, Python, Java, Frontier, C/C++, Lisp, PHP, Microsoft .NET, Rebol, Real Basic, Tcl, Delphi, WebObjects, and Zope.

1.3.3. SOAP

SOAP (Simple Object Access Protocol) is another XML/HTTP-based protocol for accessing services, objects, and servers on the Internet [66]. It is platform-independent. It consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses. SOAP is derived from XML-

RPC, and it is a superset of XML-RPC, but they are not compatible.

From the DMKD point of view, both XML-RPC and SOAP can be used as protocols to communicate between DM tools to create DM toolboxes. Such toolboxes would use multiple DM tools, choosing ones that are suitable to work with the supplied data and provide the user with combined results without the necessity of running the data separately using all chosen DM tools [45]. Using these protocols, the DM toolbox can access the DM tools over the Internet; as a result distributed and user-customized toolboxes can be easily built.

1.3.4. PMML

PMML (Predictive Model Markup Language) is an XML-based language used to define predictive data models and share them between compliant applications [56]. PMML was designed by the Data Mining Group (DMG) [29]. DMG is an independent vendor-led group that develops data mining standards; its members include IBM, Oracle, SPSS Inc., Angoss, and MineIt Software Ltd.. PMML is supported by products from IBM, Oracle, SPSS, NCR, Magnify, Angoss, and other companies.

PMML defines the vendor-independent method for defining models. It removes the issues of incompatibility between applications and proprietary formats. This, in turn, enables exchanging models between applications. For example, it allows users to generate data models using one vendor application and then to use another vendor application to analyze, still another to evaluate the models, and yet another to visualize the model. This is yet another very important element that would enable building DM toolboxes. Previous solutions to the problem of sharing data models were incorporated into custom-built systems, and thus exchange of models with an application outside of the system was virtually impossible.

The PMML currently supports the following DM models: decision trees, naive Bayes models, regression models, sequence and association rules, neural networks, and center- and distribution-based clustering algorithms [29]. The PMML describes the models using eight modules: header, data schema, DM schema, predictive model schema, definition for predictive models, definition for ensemble of models, rules for selecting and combining models and ensembles of models, and rules for exception handling [36]. The PMML supports not only several DM models but also the ensemble of models and mechanisms for selecting and combining the models.

1.3.5. UDDI

Universal Description Discovery and Integration (UDDI) is a platform-independent framework for describing, discovering, and integrating services using the Internet and operational registry [71]. The framework uses XML, SOAP, HTTP, and Domain Name System (DNS) protocols. Currently more than 220 companies use the UDDI. The UDDI can help solve problems like finding the correct service among millions available or interfacing with a service using Web

Services Description Language (WSDL), which is the XML-based format for describing network services [73]. At the same time, because of benefits like reaching new customers, expanding offerings, and market reach, it is almost certain that service providers will register their services using UDDI.

From the DMKD point of view, services can be DM tools that are published as online services. DM toolboxes can be implemented as clients that can use those services [45]. The DM toolbox would then check the availability of the online DM tools using UDDI and invoke the ones that can provide meaningful results for the currently processed data. The DM tools (services) then would take the data provided by the DM toolbox, process it, and return results to the toolbox. Using this protocol, a DM toolbox can dynamically access and use several DM tools, which process data and generate results. The toolbox would collect the results, process them, present them to the user, and finally store them in the knowledge base. This simple mechanism, powered by dynamic online DM tools, can be used to build flexible and widely applicable DM toolboxes.

These technologies will certainly help in semiautomating the DMKD process. XML can be used to store data and PMML to store data models. SOAP and XML-RPC can be used for platform-independent communication between different DM applications, and UDDI can be used to find DM services offered by DM companies. A more detailed description of how to incorporate the technologies into the DMKD process is given later. A big advantage of these technologies is that they are open source and thus can be freely downloaded and used.

1.3.6. OLAP

OLAP (online analytical processing) is a relatively old DMKD technology. Its main purpose is to provide users with multidimensional views of aggregate data for quick access to the needed information for further analysis. OLAP gives fast, consistent, interactive access to a variety of views of any information. OLAP and data warehouses (DW) are complementary technologies. A DW stores and manages data whereas OLAP transforms the data into possibly strategic information. OLAP services range from basic navigation and browsing (called slice and dice) data, to analyses such as time series processing. OLAP gives the user some decision-making power. The most common applications of OLAP are marketing, promotions, customer analysis, sales forecasting, and market and customer segmentation. OLAP has the following characteristics:

- *multidimensional views*, which help in analytical processing of the data through flexible access to information. Users can analyze data in any dimension and at any level of aggregation.
- *time intelligence*, which means that OLAP systems can deal with the sequential nature of time. The notion of time should be built as an integral feature to any analytical package.
- *complex calculations*, which give the user a tool to, for instance perform share calculations (percentage of the total), allocations (which use hierarchies from a top-down perspective); they use trend algorithms such as moving averages and percentage growth.

One advantage of OLAP systems is that they can be evaluated using a standardized set of benchmarks; for example, the OLAP Council APB-1 performance benchmark simulates a realistic OLAP business situation [4]. The goal of APB-1 is to measure overall OLAP performance rather than the performance of specific tasks. The operations performed during the APB-1 test include: bulk loading of data, incremental loading of data, aggregation of data along hierarchies, calculation of new data based on business models, time series analysis, queries with a high degree of complexity, drill-down through hierarchies, ad hoc queries, and multiple online sessions. In short, OLAP provides fast data summarization and basic data processing. It can be used as one of the preprocessing tools during the DMKD process, to make it more efficient and easier to perform. Also, OLAP technology can be directly integrated with a majority of other DM algorithms like association rules, classification, prediction, and clustering [38].

OLAP is well coupled with DW because the data warehouses are designed differently from traditional relational DBMS. DW is a central data repository that defines integrated data models for data normally stored in a number of different locations. It incorporates a subject-oriented read-only historical data. This not only guarantees stability of the data but also gives flexibility to effectively query the data stored in a warehouse.

1.3.7. OLE DB-DM

OLE DB-DM (OLE DB for Data Mining) is an extension of the SQL query language that allows users to train and test DM models [51]. Its primary use is to integrate different DM tools using a common API. The OLE DB-DM supports all of the most popular DM tools and applies DM analysis directly against a relational database. OLE DB-DM consists of these elements:

- a *data mining model* (DMM) is modeled by a relational table, except that it contains columns used for training and predictions. After the data are inserted into the table, a DM algorithm processes them and the resulting DM model is saved. Thus, the DMM can be browsed, refined, or used.
- *prediction join operation*, an operation that does a join query between a trained DM model and data to generate a prediction result that can be sent to the user's application as either an OLE DB row set or an ADO (active data objects) record set.
- *OLE DB-DM schema row sets*, which allow user applications to find available DM services and models and the model contents.

One of the advantages of the OLE DB-DM is its support of standard DM data types by using flags, instead of using only the SLQ data types. The following data types are supported:

- *key* – discrete attribute that is a key.
- *continuous* – attribute with continuous values.
- *discrete* – attribute with discrete values.
- *discretized* – attribute is continuous and should be discretized.

- *ordered* – attribute with discrete values that are ordered.
- *cyclical* – attribute with discrete values that are ordered and cyclical, e.g., weekdays.
- *sequence time* – attribute containing time measurement units.
- *sequence* – attribute containing the sorting key of the related attributes.

The OLE DB-DM supports the following DM models:

- classification when the predicted attribute is categorical.
- regression when the predicted attribute is continuous.
- clustering.
- association (data summarization) including association rules.
- sequence and deviation analysis.
- dependency modeling used to identify dependencies among attributes.

Two main advantages of the OLE DB-DM are:

- it can interface with PMML, because all of the structure and content of a DMM may be expressed as an XML string in PMML format
- it can interface with the OLAP technology.

The technologies described above can be used to integrate and semiautomate the DMKD process, on the level of manipulation and sharing data, and on the level of data models. XML-based technologies can be used to store data and DM data models and to provide communication protocols between DM tools. OLAP can be used during the data preprocessing step, and OLE DB-DM can be used to integrate DM tools with relational DBMSs.

1.4 Future of Data Mining and Knowledge Discovery?

IDC, a well-known provider of technology intelligence and industry analysis, estimates that the data mining tools market will reach $1.85 billion in 2006. In 1998, Simoudis of IBM predicted that "within five years, [data mining] will be as important to running a business as the business systems are today;" his prediction has proven to be already correct (2004). On the other hand, many business managers are willing to conduct DMKD on their data but they are not sure where to start [8].

The DMKD community developed several successful DM methods over the last few years. A survey of software implementations of DM methods presents a comparison of 43 existing implementations of DM methods [35]. Unfortunately, just having a variety of DM methods does not solve the problems of DMKD, like the necessity of integrating DM methods, integrating them with the DBMS, and providing support for novice users.

To provide a framework to address these issues we start by defining DM methods and DM tools. A DM method is simply an implementation of a DM algorithm; a DM tool is a DM method that can communicate and operate in the DMKD environment. Development of DM tools or upgrading the existing

methods to tools, as well as improving integration of the entire DMKD process, may help a lot to address the existing problems. XML and XML-based technology provide tools for transforming DM methods into DM tools, combining them into DM toolboxes, and, most importantly, semiautomating the DMKD process. The DMKD research community recognizes importance of XML technology for data preparation and as a medium to store, retrieve, and use the domain knowledge via the use of PMML [15].

XML provides a universal format for storing structured data. Because it is supported by current DBMSs it is becoming a standard not only for data transport but also for data storage. The PMML language can be used to transmit and store metadata. It is one of the technologies that can substantially simplify the design of complete DMKD systems and increase their flexibility [36]. Hence we predict the creation of metadata repositories (knowledge repositories) that would use the PMML format to store their content. SOAP and XML-RPC are two communication protocols that are not only platform-independent but that also eliminate the need for direct API calls, make the communication easy, and support compatibility between applications that exchange data. Because these protocols are loosely coupled, one can communicate in a developer- and user-friendly manner; say, between applications written in C++ on the Linux operating system and another application written in COBOL on the Windows system. Traditional communication protocols based on COM, DCOM, and CORBA models are tightly coupled, which makes development of the integration procedures not only very difficult, but also inefficient and costly [5]. On the other hand, the SOAP communication protocol is seamless in terms of implementation because most of the software development packages already offer libraries that support this technology. As a result it is very easy to communicate between DM tools and the DM toolbox using these protocols. The UDDI is another technology that enables building flexible DM toolboxes. By using it we can build online toolboxes that can dynamically search, access, and use DM tools that are published as Web services. OLE DB-DM is the technology that allows the use of DM algorithms within the existing DBMS products while avoiding problems of interfacing between DM tools and the DBMSs.

These technologies can, and we think will, be used to support all stages of the DMKD process. Figure 1.5 shows the DMKD model based on these technologies, which supports semiautomation of the DMKD process.

The database and knowledge database can be stored using a single DBMS that supports XML, because the PMML used to store the knowledge complies with the XML format. We separate the two to underscore the difference in format and functionality of the information they store. The database is used to store and query the data. All of the DMKD steps, however, can store information and communicate using the knowledge database. The advantages of implementing the knowledge database are: automation of knowledge storage and retrieval, sharing of the discovered knowledge between different domains, and supporting semiautomation of two DMKD steps: understanding the data and preparation of the data. The architecture shown in Fig. 1.5 has the advantage of supporting the iterative and interactive aspects of the DMKD process. It simply makes sense to support the entire DMKD process rather than only a single DM step.

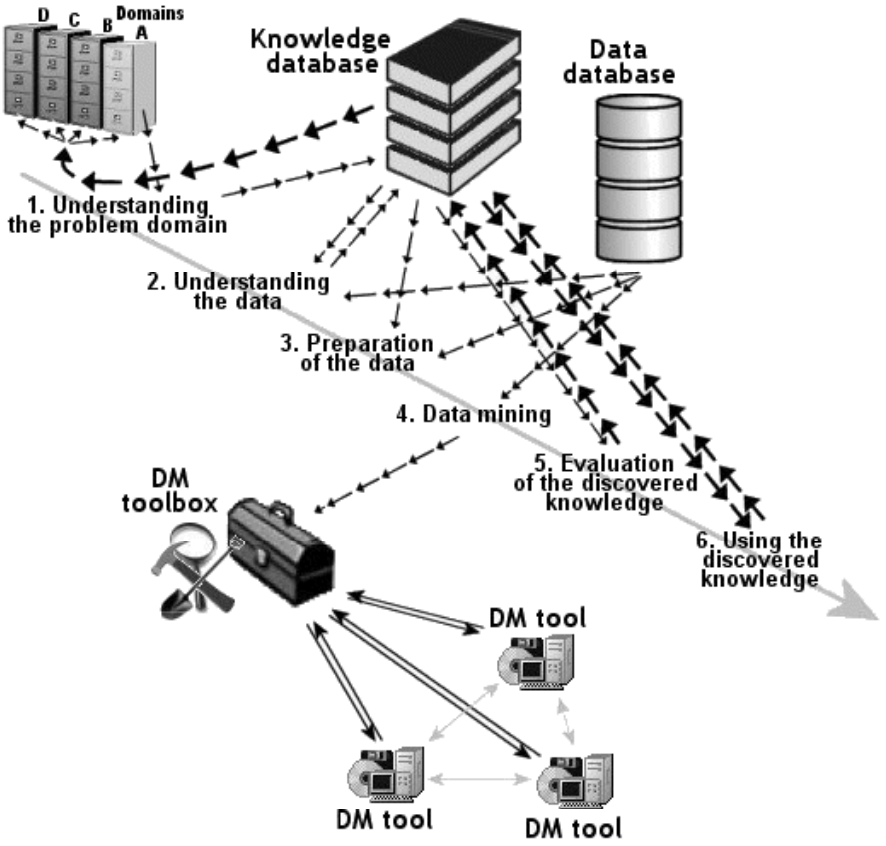


Fig. 1.5. The automation of the DMKD process using XML based technologies.

A DM step may use a DM toolbox that integrates multiple DM tools [45]. The DM toolbox architecture, based on XML, is shown in Fig. 1.6.

The idea of implementing DM toolboxes arises from a simple observation that no single DM tool performs well on all types of data. XML and XML-based technology like SOAP and UDDI make the implementation of such toolboxes easy. First, the DM tools are registered as Web services using the UDDI registry. The DM toolbox performs a series of steps to generate knowledge from data. It loads the data from a database, and then using UDDI and WSDL descriptions it scans for DM tools that are available and suitable for particular analysis. Next, it communicates with selected DM tools, provides them with the data, and receives the results of analyses. Finally, it processes the results and stores them in the knowledge database.

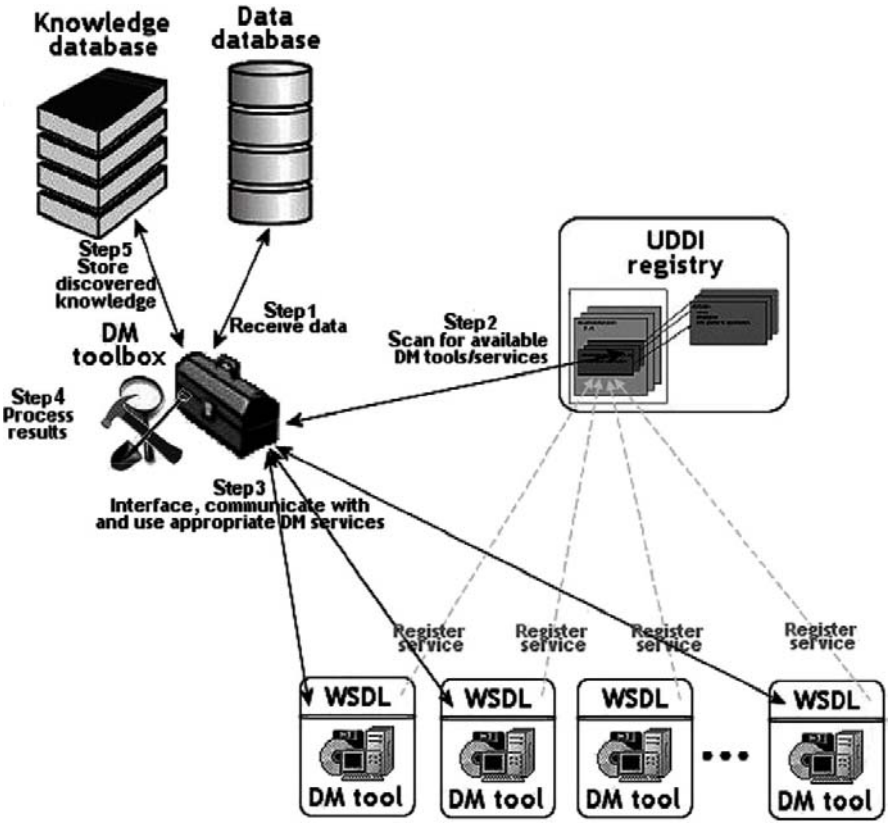


Fig. 1.6. DM toolbox architecture using the Internet via HTTP, XML, SOAP, or XML-RPC and UDDI.

The business community already tries to integrate the DMKD process. During the last few years businesses showed growing interest in DMKD. The biggest DBMS vendors like IBM, Microsoft, and Oracle integrated some of the DM tools into their commercial systems. Their goal is to make use of DM methods easier, in particular for users that use their DBMS products. IBM's DM tool, called Intelligent Miner, which integrates with DB2, consists of three components: Intelligent Miner for Data, Intelligent Miner for Text, and Intelligent Miner Scoring [58], [59]. Intelligent Miner for Data uses clustering based on Kohonen neural network, factor analysis, linear and polynomial regression, and decision trees to find associations and patterns in data [17], [28], [43]. Intelligent Miner for Text includes a search engine, Web access tools, and text analysis tools. Intelligent Miner Scoring is the DM component designed to work in real time. Intelligent Miner incorporates some data preprocessing methods like feature selection, sampling, aggregation, filtering, cleansing, and data transformations like principal component analysis [17]. It also supports the PMML format. Microsoft's SQLServer2000 incorporates two DM algorithms: decision trees and clustering [69]. The implementation is based on the OLE DB-DM specification. Oracle has a

DM tool called Oracle Darwin®, which is a part of the Oracle Data Mining Suite [52]. It supports DM algorithms like neural networks, classification and regression trees, memory-based reasoning (based on *k*-nearest neighbor approach), and clustering (based on *k*-means algorithm) [13], [17]. Their solution integrates with the Oracle 9i DBMS.

These products provide tools to automate several steps of the DMKD process like preparation of the data and DM. However, they only partially solve the issue of semiautomation of the entire DMKD process because they do not provide an overall framework for carrying out the DMKD process.

1.5 Conclusions

Currently the DMKD "industry" is quite fragmented. It consists of research groups and field experts that do not work closely with decision makers. This is caused by a situation where the DMKD community generates new solutions that are not widely accessible to a broader audience and are difficult to use. Because of that, and the high cost of performing the DMKD process, DMKD projects are undertaken by companies which can afford them and urgently need to analyze large amounts of data they constantly collect, but many other businesses reject it because of the costs involved. To solve this problem we need to semiautomate the DMKD process, provide integrated DM tools and services, thus making the DMKD process easier and less expensive to use by the end user.

The technologies described in this chapter (XML, XMP-RPC, SOAP, PMML, UDDI, OLAP, and OLE DB-DM) will play a significant role in the design of the such next-generation DMKD process framework. These technologies will make it possible to build DM toolboxes that span multiple DM tools; to build knowledge repositories; to communicate and interact between DM tools, DBMSs, and knowledge repositories; and most importantly, to semiautomate the entire DMKD process. These technologies also can be used to deploy the DMKD process that will include elements that run on different platforms because they are platform-independent. Another advantage of these technologies is that they will bring the DMKD industry to a new level of usability. New users, who will follow these standards, in spite of their lack of deep knowledge of DMKD, will be exposed to and attracted by DMKD applications.

In addition to the design and implementation of a new DMKD framework, a more traditional course of action will need to be undertaken [37]. It includes design and implementation of a new generation of high-performance DM systems that incorporate multiple DM methods [76] and are capable of mining heterogeneous sources of knowledge like multimedia data [77], can visualize the results, and can handle huge amounts of complex data. One of the goals in designing such systems should be the design of better user interfaces. This will result in a wider acceptance of the products, particularly by midsize and small companies with limited technical skills. Another important issue is to learn about the user perception of the novelty, understandability, and simplicity of the knowledge generated by the DMKD process. We also should take into account the

human cognitive processes and learn how people assimilate new knowledge to increase the usefulness of the new generation of DMKD tools [53]. Such studies would greatly help to increase acceptance of DMKD tools.

In a nutshell, being able to model real problems using easy-to-follow procedure is the major reason for designing the integrated DMKD process. Having such a process will help organizations to respond more quickly to market opportunities and threats, and to increase revenues and operational efficiencies.

Acknowledgments

The authors thank Dr. Harry Direen and Mr. Michael Trombley for helping to improve the readability of the chapter.

References

[1] Abiteboul, S., Quass, D., McHugh, J., Widom, J., and Wiener, J., The Lorel query language for semistructured data, *International Journal on Digital Libraries*, 1:1, pp. 68–88, 1997.

[2] Agrawal, R., and Srikant, R., Fast algorithms for mining association rules, *Proceedings of the 20th International Conference on Very Large Databases*, Santiago, Chile, 1994.

[3] Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., and Verkamo, A. I., Fast discovery of association rules, *Advances in Knowledge Discovery and Data Mining*, chapter 12, AAAI/MIT Press, 1995.

[4] APB-1, *OLAP Council APB-1 OLAP Benchmark Release II*, http://www.olapcouncil.org/ research/bmarkco.htm, 1998.

[5] Apps, E., New mining industry standards: Moving from monks to the mainstream, *PC AI*, 14:6, pp.46–50, 2000.

[6] AXIOM, http://axiom.iop.org/, 2001.

[7] Banerjee, S., Krishnamurthy, V., Krishnaprasad, M., and Murthy, R., Oracle8i - The XML enabled data management system, *Proceedings of the 16th International Conference on Data Engineering*, pp. 561–8, San Diego, CA, 2000.

[8] Bauer C. J., Data mining digs in special advertising recruitment supplement to *The Washington Post*, March 15, 1998.

[9] Bourret, R., Bornhvd, C. and Buchmann, A. P., A generic load/extract utility for data transfer between XML documents and relational databases, *Proceedings of the 2nd International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems*, pp.134–143, San Jose, CA, June 2000.

[10] Brachman, R., and Anand, T., The process of knowledge discovery in databases: A human-centered approach, in Fayyad, U.M., Piatesky-Shapiro, G., Smyth, P., and Uthurusamy, R. (eds), *Advances in Knowledge Discovery and Data Mining*,

AAAI/MIT Press, 1996.

[11] Bradley, P., Fayyad, U., and Reina, C., Scaling clustering algorithms to large databases, *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, Menlo Park, CA, pp. 9–15, 1998.

[12] Bray, T., Paoli, J., and Maler E., *Extensible Markup Language (XML) 1.0* (second edition), W3C Recommendation, http://www.w3.org/TR/2000/REC-xml-20001006, October 2000.

[13] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J., *Classification and Regression Trees*, Wadsworth Int. Group, Belmont, CA, 1984.

[14] Buneman, P., Fernandez, M. F., Suciu, D., UnQL: A query language and algebra for semistructured data based on structural recursion, *Very Large Databases Journal*, 9(1), pp. 76–110, 2000.

[15] Büchner, A. G., Baumgarten, M., Mulvenna, M. D., Böhm, R., and Anand, S. S., Data mining and XML: Current and future issues, *Proceedings of the 1st International Conference on Web Information Systems Engineering (WISE'00)*, pp. 127–31, Hong Kong, 2000.

[16] Cabena, P., Hadjinian, P., Stadler, R., Verhees, J., and Zanasi, A., *Discovering Data Mining: From Concepts to Implementation*. Prentice Hall, 1998.

[17] Cios, K. J., Pedrycz, W., and Swiniarski, R., *Data Mining Methods for Knowledge Discovery*, Kluwer, 1998.

[18] Cios, K. J., Teresinska, A., Konieczna, S., Potocka, J., and Sharma, S., Diagnosing myocardial perfusion from PECT bull's-eye maps - A knowledge discovery approach, *IEEE Engineering in Medicine and Biology Magazine*, Special issue on Medical Data Mining and Knowledge Discovery, 19:4, pp. 17–25, 2000.

[19] Cios, K. J. (ed.), *Medical Data Mining and Knowledge Discovery*, Springer, 2001.

[20] Chamberlin, D., Robie, J., and Florescu, D., Quilt: An XML query language for heterogeneous data sources, *Proceedings of the3rd International Workshop on the Web and Databases (WebDB 2000)*, Dallas, TX, 1997 of *Lecture Notes in Computer Science*, 2000.

[21] Chamberlin, D., Clark, J., Florescu, D., Robie, J., Siméon, J., and Stefanescu, M., *XQuery 1.0: An XML Query Language*, W3C Working Draft, http://www.w3.org/TR/xquery/, 2001.

[22] Chen, M. S., Han, J., and Yu, P. S., Data mining: An overview from database perspective, *IEEE Transactions on Knowledge and Data Engineering*, 8:6, pp. 866–883, 1996.

[23] Cheng, J., and Xu, J., IBM DB2 extender, *Proceedings of the Sixteenth International Conference on Data Engineering*, pp. 569–73, San Diego, CA, 2000.

[24] Clark, J., and DeRose, S., *XPath: XML Path Language* (Version 1.0), W3C Recommendation, http://www.w3.org/TR/xpath, 1999.

[25] Cluet, S., and Simeon, J., *YATL: A Functional and Declarative Language for XML*, technical report, INRIA Rocquencourt and Bell Laboratories, 2000.

[26] CRISP-DM, *CRoss-Industry Standard Process for Data Mining*, www.crisp-dm.org, 2001.

[27] Deutsch, A., Fernandez, M., Florescu, D., Levy, A., and Suciu, D., *XML-QL: A Query Language for XML*, W3C Note, http://www.w3.org/TR/NOTE-xml-ql/, 1998.

[28] Dillon, W. R., and Goldstein, M., *Multivariate Analysis: Methods and Applications*, New York: Wiley, 1984.

[29] DMG, *The Data Mining Group*, http://www.dmg.org/, 2001.

[30] Fayyad, U. M., Piatesky-Shapiro, G., Smyth, P., and Uthurusamy, R., *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press, 1996.

[31] Fayyad, U. M., Piatetsky-Shapiro, G., and Smyth, P., Knowledge discovery and data mining: Towards a unifying framework, *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD96)*, Portland, OR, AAAI Press, 1996.

[32] Ganti, V., Ramakrishnan, R., Gehrke, J., Powell, A. L., and French, J. C., Clustering large datasets in arbitrary metric spaces. *Proceedings of the 15th International Conference on Data Engineering (ICDE)*, pp. 502–11, 1999.

[33] Ganti, V., Gehrke, J., and Ramakrishnan, R., Mining very large databases, *IEEE Computer*, 32:8, pp. 38–45, 1999.

[34] Gehrke, J., Ramakrishnan, R., and Ganti, V., RainForest – a framework for fast decision tree construction of large datasets, *Proceedings of the 24th International Conference on Very Large Data Bases*, San Francisco, pp. 416–27, 1998.

[35] Goebel, M., and Gruenwald, L., A survey of data mining software tools, *SIGKDD Explorations*, 1:1, pp. 20–33, 1999.

[36] Grossman, R. L., Bailey, S., Ramu, A., Malhi, B., Hallstrom, P., Pulleyn, I., and Qin, X., The management and mining of multiple predictive models using the predictive modeling markup language, *Information and Software Technology*, 41:9, pp 589–95, 1999.

[37] Han, J., Data mining: Where is it heading? (Panel abstract), *Proceedings of the 1997 International Conference on Data Engineering* (ICDE'97), Birmingham, England, pp. 462, 1997.

[38] Han, J., OLAP mining: An integration of OLAP with data mining, in Spaccapietra, S., and Maryanski, F., (eds.), *Data Mining and Reverse Engineering: Searching for Semantics*, Chapman & Hall, pp. 3–20, 1998.

[39] Hirji, K. K., Exploring data mining implementation, *Communications of the ACM*, 44:7, pp. 87–93, July 2001.

[40] Hosoya, H., and Pierce, B. C., XDuce: A typed XML processing language, *Proceedings of the 3rd International Workshop on the Web and Databases (WebDB 2000)*, Dallas, TX, 1997 of *Lecture Notes in Computer Science*, pp. 226–44, 2000.

[41] Informix Object Translator, http://www.informix.com/idn-secure/webtools/ot/, 2001.

[42] ISO, ISO 8879:1986. *Information Processing – Text and Office Systems – Standard Generalized Markup Language* (SGML), 1986.

[43] Kohonen, T., *Self-Organisation and Associative Memory*, Springer-Verlag, (Berlin) Springer Series in Information Sciences 8, 1989.

[44] Kurgan, L., Cios, K. J., Tadeusiewicz, R., Ogiela, M., and Goodenday, L. S., Knowledge discovery approach to automated cardiac SPECT diagnosis, *Artificial Intelligence in Medicine*, 23:2, pp. 149–69, 2001.

[45] Kurgan, L., Cios, K. J., and Trombley, M., The WWW based data mining toolbox architecture, *Proceedings of the 6th International Conference on Neural Networks and Soft Computing*, pp. 855–60, Zakopane, Poland, 2002.

[46] Kurgan, L., & Cios, K., Meta Mining Architecture for Supervised Learning, *Proceedings of the 7th International Workshop on High Performance and Distributed Mining*, in conjunction with the *4th International SIAM Conference on Data Mining*, pp. 18–26, Lake Buena Vista, FL, 2004.

[47] Kurgan, L., Cios, K. J., Sontag, M., and Accurso, F. J., Mining a cystic fibrosis database, in: Zurada, J., and Kantardzic, M. (eds.), *Novel Applications in Data Mining*, IEEE Press, in print 2004.

[48] Kurgan, L., *Meta Mining System for Supervised Learning*, Ph.D. dissertation, University of Colorado at Boulder, Department of Computer Science, 2003.

[49] McHugh, J., Abiteboul, S., Goldman, R., Quass, D., and Widom, J., Lore: A database management system for semistructured data, *SIGMOD Record*, 26:3, pp.54–66, 1997.

[50] Native XML DBMS, http://www.rpbourret.com/xml/XMLDatabaseProds.htm, 2001.

[51] *OLE DB for Data Mining Specification*, version 1.0, Microsoft Corporation, http://www.microsoft.com/data/oledb/dm.htm, July 2000.

[52] Oracle Data Mining Suite, Oracle Darwin®, http://technet.oracle.com/products/datamining/htdocs/datasheet.htm, 2001.

[53] Pazzani, M. J., Knowledge discovery from data? *IEEE Intelligent Systems*, pp.10–3, March/April 2000.

[54] Piatesky-Shapiro, G., Knowledge discovery in real databases: A report on the IJCAI-89 Workshop, *AI Magazine*, 11:5, pp. 68–70, Jan. 1991.

[55] Piatetsky-Shapiro, G., and Frawley, W., (eds), *Knowledge Discovery in Databases*, AAAI/MIT Press, 1991.

[56] PMML*, 2nd Annual Workshop on the Predictive Model Markup Language*, San Francisco, CA, August 2001.

[57] Redman, T. C., The impact of poor data quality on the typical enterprise, *Communications of the ACM,* 41:2, pp.79–81, 1998.

[58] Rennhackkamp, M., IBM's intelligent family, *DBMS Magazine*, http://www.dbmsmag.com/9808d17.html, August 1998.

[59] Reinschmidt, J., Gottschalk, H., Kim, H., and Zwietering, D., *Intelligent Miner for Data: Enhance Your Business Intelligence*, IBM International Technical Support Organization (IBM Redbooks), IBM Corporation, 1999.

[60] Robie, J., Lapp J., and Schach D., XML query language (XQL), *The Query*

Languages Workshop (QL 1998), Boston, 1998.

[61] Sacha, J. P., Cios, K. J., and Goodenday, L. S., Issues in automating cardiac SPECT diagnosis, *IEEE Engineering in Medicine and Biology Magazine*, special issue on Medical Data Mining and Knowledge Discovery, 19:4, pp. 78–88, 2000.

[62] Schoening, H., Tamino–a DBMS designed for XML, *Proceedings of the 17th IEEE International Conference on Data Engineering*, pp.149–54, Los Alamos, CA, 2001.

[63] Shafer, J., Agrawal, R., and Mehta, M., SPRINT: A scalable parallel classifier for data mining, *Proceedings of the 22nd International Conference on Very Large Data Bases*, San Francisco, pp. 544–55, 1996.

[64] Shanmugasundaram, J., Gang, H., Tufte, K., Zhang, C., DeWitt, D. J., Naughton, J. F., Relational databases for querying XML documents: Limitations and opportunities. *Proceedings of the 25th International Conference on Very Large Databases*, pp. 302–14, 1999.

[65] Shimura, T., Yoshikawa, M., and Uemura, S., Storage and retrieval of XML documents using object-relational databases, *Proceedings of the 10th International Conference on Database and Expert Systems Applications*, Lecture Notes in Computer Science, 1677, Springer-Verlag, pp. 206–17, Aug.-Sep. 1999.

[66] SOAP 1.1, W3C Note, http://www.w3.org/TR/SOAP/, May 2000.

[67] Spiliopoulou, M., and Roddick, J. F., Higher order mining: Modelling and mining the results of knowledge discovery, *Data Mining II – Proceedings of the 2nd International Conference on Data Mining Methods and Databases*, Cambridge, UK, pp. 309–20, 2000.

[68] *SQL Server Magazine: The XML files*, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsqlmag2k/html/TheXMLFiles.asp, 2000.

[69] Tang, Z., and Kim, P., *Building Data Mining Solutions with SQL Server 2000, DM Review*, White Paper Library, http://www.dmreview.com/whitepaper/wid292.pdf, 2001.

[70] Toivonen, H., Sampling large databases for association rules, *Proceedings of the 22nd International Conference on Very Large Data Bases*, San Francisco, pp. 134–45, 1996.

[71] Universal Description, Discovery, and Integration (UDDI) specification, version 2.0, http://www.uddi.org/, 2001.

[72] Wirth, R., and Hipp, J., CRISP-DM: Towards a standard process model for data mining, *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, pp. 29–39, Manchester, UK, 2000.

[73] Web Services Description Language (WSDL) 1.1, W3C Note, http://www.w3.org/TR/wsdl, March 2001.

[74] XML and Access2002, *Exploring XML and Access 2002*, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnacc2k2/html/odc_acxmllnk.asp, 2001.

[75] XML-RPC, *UserLand Software*, Inc., http://www.xmlrpc.com/, 2001.

[76] Yaginuma, Y., High-performance data mining system, *Fujitsu Scientific and*

Technical Journal, special issue on *Information Technologies in the Internet Era*, 36:2, pp. 201–10, 2000.

[77] Zaïane, O. R., Han, J., Li, Z. N., Hou, J., Mining multimedia data, *Proceedings of the CASCON'98: Meeting of Minds*, Toronto, Canada, 1998, pp. 83–96, 1998.

[78] Zhang, T., Ramakrishnan, R., Livny, and M., BIRCH: An efficient data clustering method for very large databases, *Proceedings of the SIGMOD Conference*, pp.103–14, 1996.

2. Advanced Methods for the Analysis of Semiconductor Manufacturing Process Data

Andreas König[1] and Achim Gratz[2]

[1] Technische Universität Kaiserslautern, Kaiserslautern D-67663, Germany; email: koenig@eit.uni-kl.de

[2] Infineon Technologies Dresden GmbH & Co. OHG, D-01076 Dresden, Germany.

The analysis, control, and optimization of manufacturing processes in the semiconductor industry are applications with significant economic impact. Modern semiconductor manufacturing processes feature an increasing number of processing steps with an increasing complexity of the steps themselves to generate a flood of multivariate monitoring data. This exponentially increasing complexity and the associated information processing and productivity demand impose stringent requirements, which are hard to meet using state-of-the-art monitoring and analysis methods and tools. This chapter deals with the application of selected methods from soft computing to the analysis of deviations from allowed parameters or operation ranges, i.e., anomaly or novelty detection, and the discovery of nonobvious multivariate dependencies of the involved parameters and the structure in the data for improved process control. Methods for online observation and offline interactive analysis employing novelty classification, dimensionality reduction, and interactive data visualization techniques are investigated in this feasibility study, based on an actual application problem and data extracted from a CMOS submicron process. The viability and feasibility of the investigated methods are demonstrated. In particular, the results of the interactive data visualization and automatic feature selection methods are most promising. The chapter introduces to semiconductor manufacturing data acquisition, application problems, and the regarded soft-computing methods in a tutorial fashion. The results of the conducted data analysis and classification experiments are presented, and an outline of a system architecture based on this feasibility study and suited for industrial service is introduced.

2.1 Introduction

The exponential increase of available computational resources leads to an explosive growth in the size and complexity of application-specific databases. In fact, today's industrial sites can produce so much data per day that the evaluation of potentially beneficial information and even complete storage become close to impossible. The monitoring of complex processes, for instance, in industrial manufacturing, however, requires online monitoring and decision making as well as ensuing extraction of nonobvious information and

structure of the data. This procedure of knowledge discovery and the online decision making serve to control the respective complex processes, e.g., for quality assurance purposes, keeping the process in a multivariate window of allowed parameter tolerances.

One important instance of this general problem class with a significant commercial impact and stringent information processing demands is represented by the analysis, control, and optimization of manufacturing processes in the semiconductor industry. Typical aims are the centering of the process in a so-called process window and the assurance of an optimum yield based on functional and electrical tests. For instance, in [2.53] a good general introduction to the topic can be found. In this particular work, decision trees are applied to determine significant individual variables or groups of variables. A more focused example is given in [2.3], where data mining and various classification techniques are applied to a single processing step dealing with wafer cleaning. Leading-edge technology and the corresponding manufacturing lines have reached an unprecedented complexity in terms of both required machinery and the required process monitoring, control, and optimization demands. Thus, modern semiconductor manufacturing processes feature an increasing number of processing steps with an increasing complexity of the steps themselves from initial wafer preparation to final passivation. Due to the continued validity of Moore's exponential growth law (see, e.g., the SIA ITRS roadmap [2.2]) the complexity of the processes will continue to increase at a rapid pace. In Section 2.2.2, a brief introduction to this part of the presented work will be given. Consequently, a tremendous amount of monitoring data are generated by the manufacturing line. The generated data have to be analyzed with regard to the required process specification or qualification, i.e., whether the process remains in the process window (see Fig 2.1). In simple models, the process window can be described, e.g., by a multiparameter or multivariate bounding box with thresholds in each parametric dimension. Exceeding the threshold makes overt that the process is going out of specification for one or several of the involved parameters. This approach neglects multivariate dependencies and higher-order correlations of variable groups. Figure 2.2 depicts typical problems occurring, such as the process being off-centered or showing correlated parameters or multimodality. The same holds for the typ-

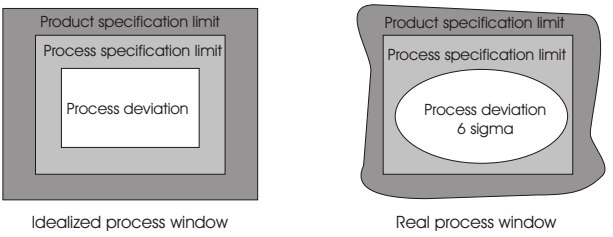


Fig. 2.1. Illustration of a process window.

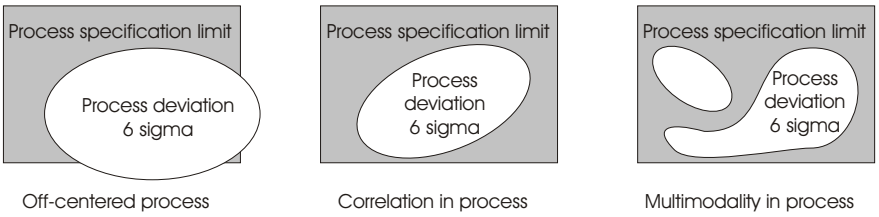


Fig. 2.2. Illustration of process window problems.

ical statistical analysis approach employed for the analysis and evaluation of process-related data. Individual parameters are checked for model consistency with regard to univariate, typically Gaussian assumptions. Further, methods like principal component analysis are used, which by its nature is a linear and parametric approach and, thus, is of limited applicability for nonlinear cases not obeying a multivariate Gaussian model. The significant economic potential of the data mining field in general and the field of semiconductor process data analysis in particular has triggered many activities. Numerous statistical tools with interactive visualization have recently become available. For instance, for the semiconductor industry, tools like dataPOWERsc [2.51], Knights' Yield Manager [2.52], or Q-Yield [2.7] are on the market. These tools dominantly apply parametric first order methods, i.e., methods based on the statistical information of a single variable or the correlation of two selected variables.

Thus, for the cases regarded earlier, advanced methods from soft computing originating from the fields of pattern recognition, neural networks, bio-inspired computing and statistics, and corresponding tool implementations provide improved leverage by multivariate, nonparametric, and nonlinear approaches. In Section 2.3.1, specific methods and their potential for advanced process window modeling and detection of deviation from the process window in (semi)automatic operation are briefly presented.

For the offline analysis of the multivariate process data as a baseline for ensuing process control and optimization, advanced methods for efficient multivariate data dimensionality reduction and interactive visualization can be salient. The benefit is given in terms of capturing multidimensional relations in the data, transparency as well as speed in the process of analysis, and knowledge extraction. In prior work of other groups, e.g., Goser's group in Dortmund [2.38] [2.14], Kohonen's self-organizing map (SOM) has been applied. In an enhancement of this work Rückert et al. [2.47] have developed the dedicated tool DANI for the analysis of semiconductor data of Robert Bosch GmbH. In this kind of application, the topology-preserving and dimensionality-reduction mapping properties of the SOM are exploited in conjunction with visualization enhancements, as, e.g., the U-Matrix of Ultsch [2.54]. The properties of the SOM and other neural networks have

also been employed in the *smart fabrication* project from 1995 to 2000 by a consortium including TEMIC, Siemens, and the University of Tübingen (Rosenstiel et al.). The detection of characteristic failure patterns [2.36] and yield prediction [2.37] were some of the pursued goals in this project.

In these and similar efforts, Kohonen's SOM has been employed with static visualization techniques. The advanced methods investigated here, however, differ in many ways and especially target on bringing improvements with regard to mapping speed, mapping error reduction, user convenience, and interactivity in the analysis process. The respective methods briefly browsed in Section 2.3.2 can serve to project data in a lower-dimensional space to make it amenable for interactive human perception–based analysis as well as automatic variable or variable group selection and pattern clustering. The objective of the current phase of the work and this chapter is to demonstrate the viability of the addressed methods for real process data extracted from a modern CMOS process. As a feasibility study, data with known but nonobvious information content prove that the methods can indeed help in rapidly detecting the desired information. In the second phase of the feasibility study, novel information and knowledge shall be extracted from additional process data by applying the proposed methods, e.g., interactive multivariate data visualization. In this regard, the chapter is as organized as follows. In the next section, the general data acquisition process and the chosen instance data for the conducted experiments are described. In the following section, the spectrum of applied methods and their tool implementations are covered. Then the conducted experiments and the achieved results are presented and discussed. Before concluding, the envisioned perspective of the work and the related information processing architecture for manufacturing process monitoring and optimization are introduced.

2.2 Semiconductor Manufacturing and Data Acquisition

2.2.1 Brief History of the IC

Semiconductor devices had a slow start as a curiosity that was not well understood. Still, they had important niche applications in radio communications, when vacuum tubes could not be used. As the understanding of their principles of operation grew, refinements to the manufacturing process first enabled military applications and then delivered the first commercially available devices in the form of single-pn-junction diodes and transistors in the early 1950s. The year 1958 marked the birth of the monolithic integrated circuit, now commonly just called IC. The invention of the IC is attributed to TI engineer Jack Kilby, but without the planar manufacturing process developed in the same year by Jean Hoerni and advanced by Robert Noyce and Gordon Moore at Fairchild,[3] it would likely have taken quite a bit longer for the idea

[3] R. Noyce and G. Moore left Fairchild to cofound Intel in 1968.

to take off. Meanwhile, also at Fairchild, a group of researchers[4] were getting a handle on manufacturing stable metal-oxide semiconductor (MOS) field effect transistors. They had actually been invented decades before the bipolar transistor, but irreproducible characteristics and fast degradation had prevented their application. The MOS transistor came back into focus because as a surface device it is a natural match to planar processing. In 1963 complementary MOS,[5] or CMOS, now the dominant technology for ICs, was invented. In the April 1965 issue of *Electronics* [2.40], Gordon Moore boldly predicted[6] that the number of components per IC would double each year at least through 1975. Depending on how you count components, the actual doubling interval turned out to be 18 months, but the general pattern of exponential growth has proven to be accurate for more than 40 years, with no end in sight. One of the important consequences is that the smallest feature F of an IC has to be halved about every three years. The diminishing of the feature size is commonly called technology scaling or shrinking, derived from the fact that at larger feature sizes it sufficed to simply draw the layout of an IC at a smaller scale to go from one technology generation to the next (provided the new technology was designed to be compatible with the old). As F becomes smaller, it becomes more difficult, if not impossible, to keep this strict compatibility between technology generations; however, there are design tools to "scale" IC layouts down to the new generation while making these differences transparent. Technologies with an F of 0.13 μm are in production right now, and the next technology generation with sub-100-nm structures is imminent. These ICs will integrate more than 100 million transistors.

2.2.2 IC Production Process

The prevalent technology for producing ICs today is CMOS on silicon. The silicon substrate (called the wafer) is sliced off of a single crystal of extremely pure silicon (the ingot) at a precise angle with respect to the cristallographic orientation. The wafers are then polished to achieve an atomically smooth surface and extreme flatness. Currently, wafer diameters of 200 mm are most common, while 300 mm wafers just being put into production.

The actual IC production process takes place in clean rooms, at the so-called fab floor. Clean rooms are classified by the number of particles larger than a certain size in a cubic meter of air. A laminar flow of air from the ceiling to the bottom is maintained to quickly remove any particles becoming airborne. The IC production process is roughly divided into the wafer or frontend processing, wafer test, and the back-end processing where the chips are singulated, packaged, and subjected to more tests. Commonly test and

[4] One of them was Andrew Grove, later to become Intel employee number 4.

[5] Thus far, MOS IC technology had employed only n-conducting devices, which led to the name NMOS technology.

[6] In various forms, this prediction is now known as Moore's law. Beyond that prediction, this article is an elucidating read even today, almost 40 years later.

packaging make up more than 50% of the production cost. Wafer processing takes places in a so-called wafer fab or manufacturing line and is often further divided into front-end-of-line (FEOL) and back-end-of-line (BEOL) processing. Simply speaking, the FEOL processing provides the active devices within the silicon, BEOL processing produces the connections between the devices, and the back-end processing provides the connections to the outside world as well as protective packaging. To simplify the fab logistics, wafers typically run in lots of 25, [7] although some tools demand batches (see Fig. 2.3) of up to six lots to be used effectively, while other tools can't process a complete lot, which will then be split into smaller batches or even single wafers.

All wafer processing, whether FEOL or BEOL, has the same general structure of producing so-called layers, one after another. The whole wafer is subjected to some processing, like producing a thin film of oxide or metal. Then a mask is transferred to the wafer, most commonly by optical lithography, to selectively protect parts of the wafer from the following process steps. Then the wafer is subjected to further processing, like etching or implantation of ionized dopants. Manual and automatic inspections are inserted at various stages (Fig. 2.4). Then the mask is removed and the next layer is processed. Layers vary widely in the number, complexity, and cost required to make them. This leads to a distinction between critical and uncritical layers. Modern technologies make use of 20 to 30 layers, and this number continues to go up. The number of layers that make up the actual devices stays relatively constant. However, as the minimum feature size F continues to shrink, the exponentially growing number of devices requires much more interconnect between them. For this reason, the number of interconnect or metal layers in the BEOL, another commonly cited characteristic of a technology, increases quite rapidly. In fact, the interconnect of the devices (the BEOL) is now more costly to produce then the devices themselves (the FEOL).

2.2.3 Data from the Fab – Inline Data

Historically, each lot was accompanied by a stack of paper, called the process record. Each sheet detailed one process step and the operator would set up the tool accordingly, run the process, sign off, note remarks and the result of any measurements taken, look up the next operation, and hand the lot over to the next operator. This is really where the term semiconductor manufacturing stems from. The process record has been replaced[8] by a database and the lots are moved to the next operation by automatic transport systems (Fig. 2.5) coupled to that database. The so-called process flow is defined by the layer sequence at the top level. This has to be broken down into individual process steps, often called moves. Each of the process steps is made up of a

[7] The lot size is somtimes reduced to 12 wafers for 300-mm wafers, as a lot of 25 wafers is too heavy to be handled manually.

[8] Some fabs still use printouts to accompany the lots.

Fig. 2.3. Batched 300-mm wafers ready to go into a vertical furnace (open furnace tube on the upper left).

sequence of operations (called a recipe) within the tool. It is now common to have so-called cluster tools comprising of multiple stations capable of running a variety of processes, so a recipe can be quite complex.

While the process record has been moved into an electronic database, it has also been expanded over time to contain more data. Measurement equipment will generally store results to a dedicated database before a result summary is attached to the process record. Additionally there are separate databases dedicated to certain tools or tool groups for recipe repositories and recording events and in situ measurements during processing. The trail of

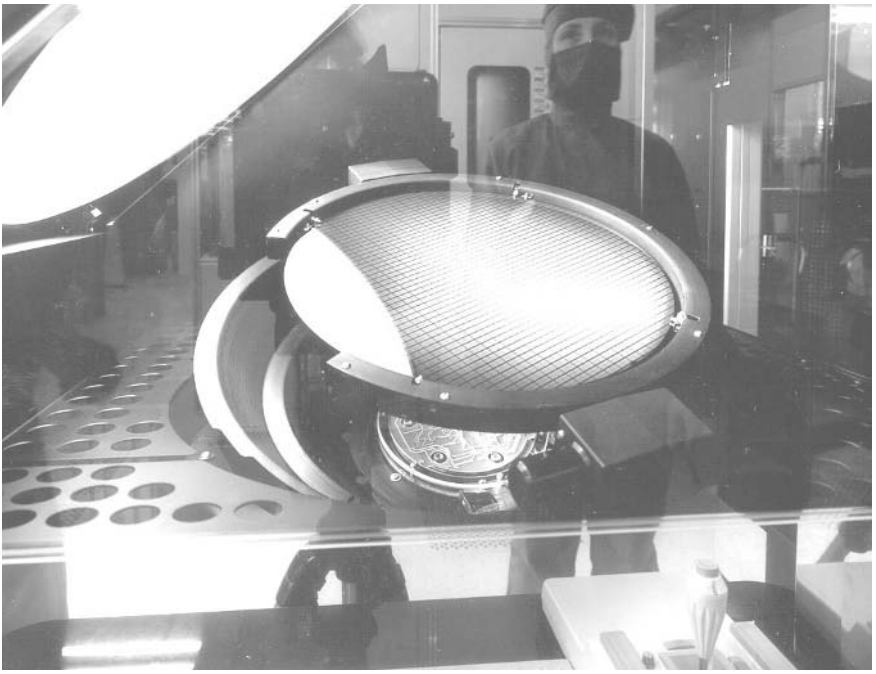


Fig. 2.4. A 300-mm wafer at so-called floodlight inspection to check for correct printing of the mask).

data collected about each lot is therefore scattered about various databases. Lately, single-wafer processing has become more important. Often the exact sequence of wafers through a single-wafer process or the position of wafers (respectively lots in batch tools) will be needed to pinpoint problems found with specific wafers. For so-called single-wafer tracking, this information needs to be fully recorded, which is only possible if all tools can read the wafer ID and lot information automatically and are connected to a database system. Additionally, a vast amount of (often temporary) data is produced and evaluated for inline process monitoring and closed-loop process control. It can be estimated that a typical semiconductor manufacturing line produces such data in excess of 1 TByte per day. It is therefore essential to evaluate, prune, and compact much of this data directly at the source. Routine reports are extracted for common purposes like maintenance, documentation, process control and optimization, and quality management. Process data that are actually stored, whether on the process tool itself or in a database, are usually kept only for a limited time or in a rolling log file to limit the storage requirements. This is far from an optimal solution as most of the data will be completely normal and therefore uninteresting, while crucial data needed to

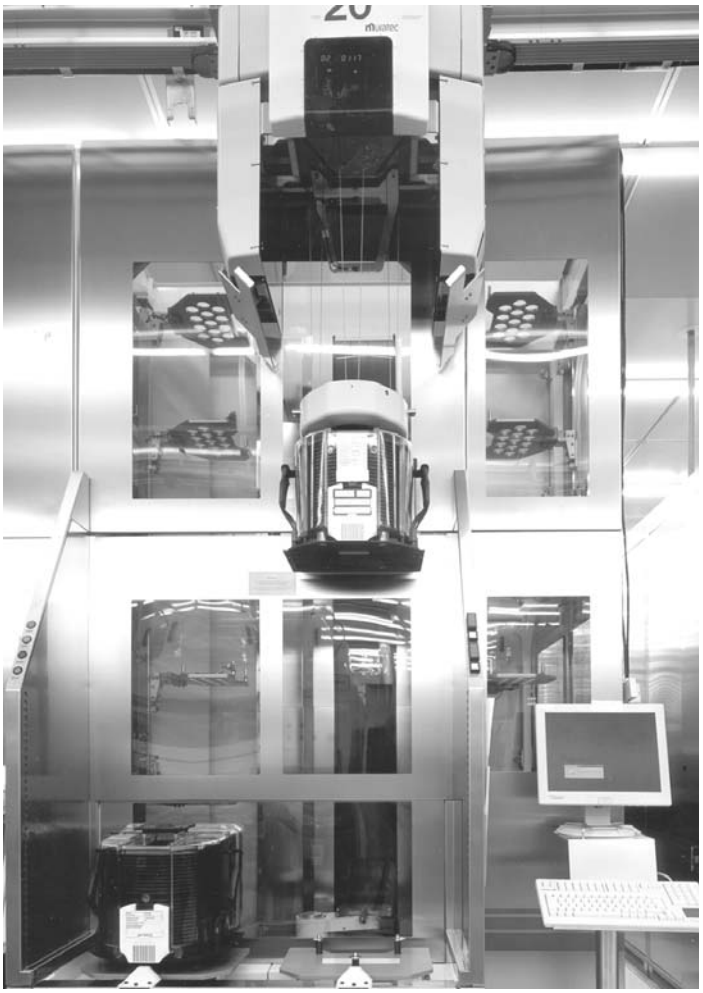


Fig. 2.5. Automatic transport system loading up a fully automatic wafer storage (Stocker).

analyze a process failure may already have been deleted before the anomaly is recognized and triggers a detailed investigation.

Collecting and evaluating all data for even a single lot are a formidable tasks. The resulting very large multivariate data set must therefore be analysed for deviations from allowed parameters or operation ranges, i.e., anomaly or novelty detection, and nonobvious multivariate dependencies of the involved parameters and the structure in the data must be disclosed for improved process control. Here, appropriate methods, e.g., from soft computing, for online observation and offline interactive analysis employing novelty clas-

sification, dimensionality reduction, and interactive data visualization techniques can be employed.

2.2.4 Electrical Test Data

After fabrication, electrical tests (ET) on the wafer level are carried out to assess that all single devices defined by the process are within their specified range. The devices tested are separate from the actual ICs on the wafer, often placed into the space between individual chips that is needed to singulate them later. These test structures are laid out carefully to isolate the layers needed to process them as much as possible from other layers. These tests are also called parametric tests as the results are actual measurement values for device parameters, like the threshold voltage or saturation currenrt of some specific transistor.

Later the actual ICs on the wafer are subjected to functional and parametric tests (FT and PT) on the wafer to decide which devices should be packaged after singulation. These tests are usually performed on a multitude of devices to save time. A sequence of tests is performed on each chip, and the first test that fails is recorded. The failed chips continue to be tested, but as the fail may have put it into an undefined state, the results of these tests cannot be relied on.

Both electrical and functional test data are stored in databases (s. Fig. 2.6) and is often preprocessed to facilitate analysis. Such preprocessing routinely includes the removal of spurious faults, calculation of derived values for parameteric data, and binning for functional data. Binning collects several individual tests that are associated with the same failure mechanism into a so-called fail bin.

Often the IC will again be tested after being fully packaged. When reliability is of utmost concern, a burn-in procedure may be performed to weed out early fails, necessitating further tests.

2.2.5 Data Analysis

Standard data analysis concentrates on keeping the process within specification limits, thus ensuring the quality of the final product. Typically a normal distribution of the measured parameter is assumed and parameters of the distribution like median and sigma are reported. In conjunction with the process specification limits, the so-called process capability *cp* and process centering *cpk* can be calculated. These methods and their application are widely accepted and mandated by various quality management methods and standards like ISO 9000.

However, their application to process specification, process trouble shooting, and process optimization often does not yield the desired results. Due to their univariate nature, complex interactions between parameters are not

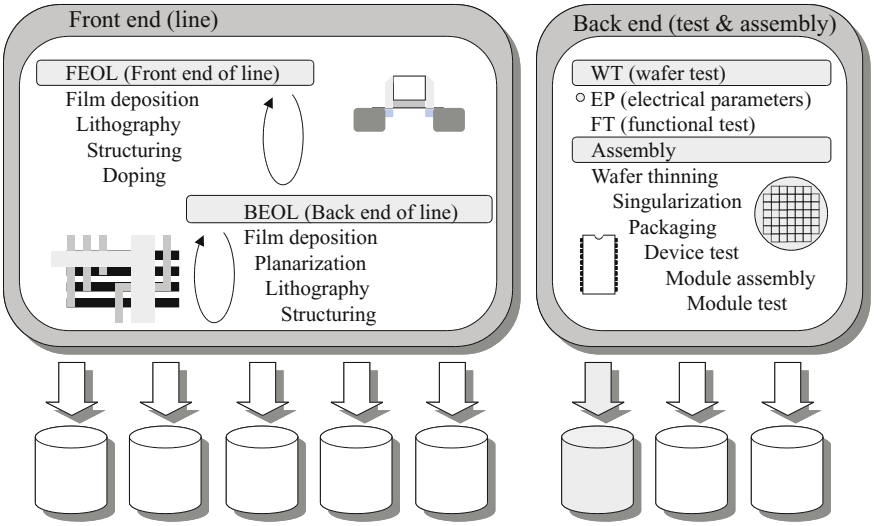


Fig. 2.6. Illustration of the overall process flow and the various origins of data.

taken into account. Also, while nonnormal distributions can in principle be accounted for properly, the procedure is cumbersome to implement and does not immediately address the failure mechanisms that change the shape of the distribution, for instance, to a multimodal distribution.

2.2.6 Process Experiment

Two lots of 25 wafers each were split identically into three groups at two process steps (s. Fig. 2.7) to vary the process parameters of these steps and in accordance the electrical parameters of certain devices. The intention of the split was to vary the threshold voltages of both n- and p-type logic transistors about the target voltage for each device. This is also called a performance split, indispensable for dynamic performance characterization, as it results in slow, nominal, and fast logic gates for the final product. As a side effect, some parameters related to the threshold voltage (most notably saturation current) and the so-called IO device coupled to the logic device will also follow the split.

This particular experiment was chosen because its effects are well known in advance and the analysis is reasonably tractable by conventional methods (Fig. 2.8). Thus there is an established baseline to compare the results of our newly developed data analysis methods against. We expect any successful method to reconstruct the split information in the two individual lots and to recognize that any residual differences between the two lots are not related to the splits, as the splitgroups are identical. Further, both the intended

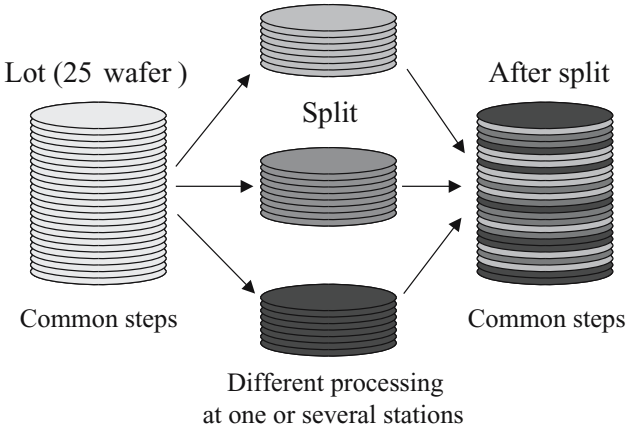


Fig. 2.7. Illustration of the split operation.

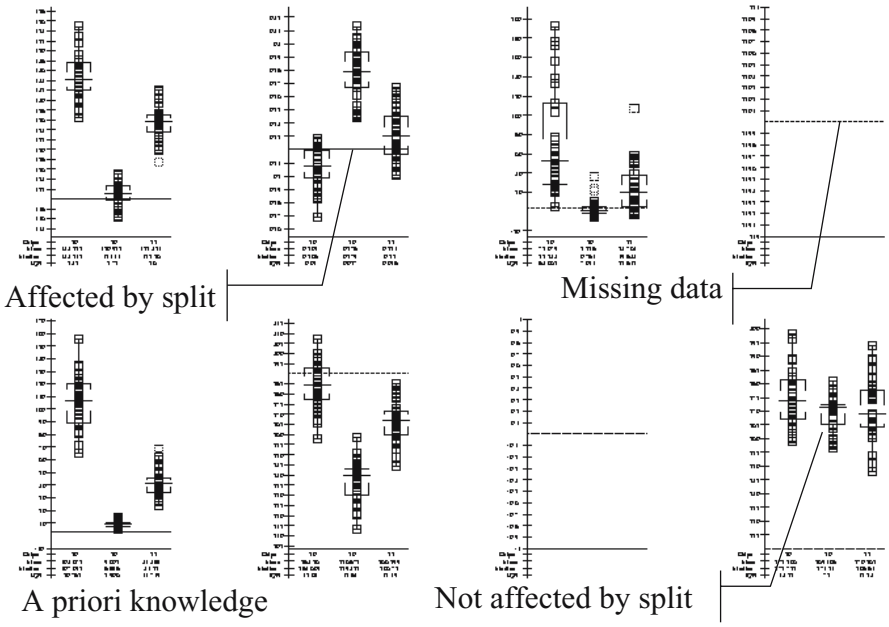


Fig. 2.8. Typical result from conventional statistical analysis. The three split groups have been separated by a priori knowledge and are shown in a single diagram to facilitate further evaluation of the experiment.

parameter changes and the side effects should be flagged as belonging to the split.

2.2.7 Experimental Data

From the proprietary software system and company database affiliated with the regarded manufacturing process, a subset of data generated for processing two wafer lots with five measurement positions for each wafer was extracted. A split of three, i.e., a partitioning of each wafer batch into three subgroups for individual processing of each partition, was carried out during production. Six wafers from the second lot were still staged in the fab for another experiment, so no data were available for these wafers. The electrical test data contain redundancies with regard to the particular split, as for each device specimen, different channel length and width are available. Also, as already explained, variation of the threshold voltage will influence further parameters belonging to that particular device. By means of conversion to an Excel spread-sheet and the application of a standard conversion tool, the database is converted to the QuickCog system requirements. The QuickCog system comprises all the methods discussed in this chapter, in particular the interactive data visualization methods and tools. A first database of 220 vectors with 205 dimensions will be regarded in the following experiments. It will be denoted by SPLIT in the following. The size of this database is given by the typical wafer batch size of 25 times the five measurement sites per wafer. However, the measurement values of six of the wafers from one set were not available, which reduces the data from the expected 250 to 220 samples. Larger databases could only be generated if larger wafer batches were made subject to identical split processing. With regard to the associated effort and cost, the aim of this work was to assess the applicability of the regarded methods also for rather sparse data of this application. No general limitation of the approach is implied by the choice of this practically relevant problem, as the regarded methods themselves scale well for large database sizes [2.23], [2.24].

Complementing the parameter data, class affiliations were generated in two files. A three-class file was generated, regarding split information only for the complete database. The data labeled by this class file will be denoted by SPLIT3 in the following. Additionally, a six-class file was generated according to lot and split affiliation of each wafer/measurement location. The labeled data will be denoted by SPLIT6 in the following. Additionally, according to the underlying lots the data have been separated into two databases denoted by SPLITTrain3 and SPLITTest3 with three classes each, corresponding to the underlying split of 3 of each lot. Finally, for the novelty classification purposes, a training set was extracted from the first lot containing data only from one split. This will be denoted by SPLITTrainOCC in the following.

2.3 Selected Soft-Computing Methods

In the following, we focus our investigations on two method groups. With the objective to achieve a (semi)automatic monitoring and control system, selected classification methods are regarded first. These shall serve the purpose of automatic assessment or classification of online generated process data with regard to its relevance and potential storage as well as the determination of the current process state within the process window. As the second group, methods for offline exploratory data analysis are regarded. We focus on relevant methods of dimensionality reduction and interactive visualization that allow us to extract nonobvious structure and underlying dependencies from the database. The results obtained using these methods also provide the baseline for the design of the effective (semi)automatic classification methods.

2.3.1 Novelty or Anomaly Detection

For the (semi)automatic classification task, powerful decision units are required that can deal with complex, nonlinear, separable, nonparametric, and potentially multimodal data. For instance k-nearest-neighbor classifiers (kNN), multi-layer perceptrons (MLP), radial-basis-function networks (RBF), or, more recently, support-vector machines (SVM) are attractive candidates for this task. In the context of the regarded application, dominantly decision trees, adaptive-resonance theory (ART) networks, and MLPs have been applied so far (see, e.g., [2.53] [2.36] [2.3]). However, especially RBF networks are intriguing for this application due to numerous salient features. In addition to being universal function approximators, RBF networks provide iterative topology learning, rapid training, fast convergence, and excellent predictable generalization capabilities [2.4], [2.43], [2.44]. In contrast to MLPs, the hidden layer of RBF networks comprises distance computation units equipped with a radially declining nonlinearity. The Euclidean distance and the Gaussian function are typical instances for RBF networks, which are closely related to the Parzen-Window technique [2.41]. However, storing all sample patterns is a significant burden with regard to storage and computation requirements. Thus, generalized RBF networks [2.4], i.e., networks with fewer hidden neurons N^* than training patterns N, are typically applied, which are given for the case of a one-dimensional function $s(\boldsymbol{x})$ by

$$s(\boldsymbol{x}) = \sum_{i=1}^{N^*} w_i \phi_i(\| \boldsymbol{x} - \boldsymbol{t}_i \|), \quad \boldsymbol{x} \epsilon \Re^M. \tag{2.1}$$

Here, t_i denotes the centroid vector of the basis function, ϕ_i denotes the radial basis function, w_i denotes the weight for the linear combination of the basis function outputs by the output neuron, $\boldsymbol{x}$ denotes an input vector, M denotes the dimension of the input vector, and N^* denotes the number of hidden neurons. Judicious and efficient choice of a sufficient but minimum

number N^* of hidden neurons is a major issue, especially for large scale problems. Several top-down and bottom-up strategies have been developed in the past [2.42] [2.15] [2.35] [2.30], employing and combining both supervised and unsupervised learning techniques. In a typical top-down strategy, a large number of centers will be determined by vector quantization techniques, e.g., Kohonen's self-organizing map. Fine-tuning of the network is achieved by a following supervised learning step, e.g., using gradient descent. Further network optimization and size reduction can be achieved by pruning techniques.

On the other hand, in bottom-up approaches the network is generated from scratch, thus completing a network-size tailored to the training data. The RBF network proposed by Platt [2.42] and the *restricted-Coulomb-energy* (RCE) network [2.46], [2.5] are significant examples of this category, as they allow dynamic automatic topology construction tailored to the problem requirements. This and an additional advantage of RBF-type networks make them excellent candidates for the investigations in this work. They also allow the concept of background classification (BC) to be implemented, which can be generalized from multiclass to one-class classification (OCC). BC is implemented by assigning the whole feature space to the selected background class. Other class regions are established by placing kernel functions and appropriately adjusting their widths during the learning process. Clearly, the network loses the rejection capability associated with the appearance of data far from the training samples. However, in cases like visual inspection or semiconductor manufacturing, in contrast to the plethora of potential errors, the desired condition can be described by sufficient examples. Thus assigning the background to such an error class can be advantageous. Initial ideas can be found in the *Nestor-learning-system* (NLS) [2.5], [2.6], which comprises a special RBF model denoted by RCE network [2.46]. The concept has been generalized to RBF networks in [2.20]. The special case of OCC, also addressed in the literature as novelty filtering [2.19] or anomaly detection [2.17], [2.31], [2.50], [2.33], is attractive because the classifier structure can be generated just by presenting data from a normal process situation. This is fortunate, as typically a lot of data from normal operation conditions are available; however, the universe of potential deviations is hard to grasp in terms of representative data samples actually covering all relevant regions in the high-dimensional parameter space for appropriate class border definition.

Thus, in the following, a model for OCC will be briefly derived from RBF-type networks for the regarded application domain.

The RCE Algorithm. The RCE network [2.46] is a special case of the RBF network given earlier. Instead of smooth nonlinearities as, e.g., the Gaussian function, a hard limiter or step function with a variable threshold parameter is applied. Each RCE basis function is equivalent to a hypersphere, represented by a center $\boldsymbol{t}_j$ and the threshold parameter, which has the meaning of a radius R_j. Each hypersphere is affiliated to one of the classes of the

application and gets activated if $S(||\boldsymbol{x} - \boldsymbol{t}_j|| <= R_j)$, i.e., if pattern $\boldsymbol{x}$ is situated within the hypersphere. The RCE output layer is also modified from a linear combination to an OR-like logic operation combining the hypersphere responses to determine the overall classification.

The algorithm practically requires only two parameter settings, $R_{\max}$ and $R_{\min}$, for operation. The following situations can arise in classification:

- A pattern is uniquely classified by one or several hyperspheres of the same class.
- No hypersphere is activated by the presented pattern. This defines a rejection mechanism, which can be controlled by setting $R_{\max}$ in training. A decision can be forced by, e.g., the nearest-neighbor rule. The rejection mechanism is replaced if the background is affiliated to one of the problem classes in BC.
- Several hyperspheres of different classes are activated by the presented pattern. The pattern is identified as ambiguous. A decision can be made according to the affiliation of the majority of the activated hyperspheres or by the nearest-neighbor rule.

The iterative RCE training algorithm starts with an empty network and presents all patterns of the training set until no more changes take place in the following basic training steps:

- If no hypersphere is activated by the presented pattern k, it is stored as $\boldsymbol{t}_{J+1}$ with $R_{J+1} = R_{\max}$, where J denotes the current number of reference vectors.
- A pattern is uniquely classified by one or several hyperspheres of the same class. All radii are left unchanged, the pattern is not stored.
- Several hyperspheres of the same and different classes are activated by the presented pattern. Radii of hyperspheres affiliated to different classes will be reduced until the pattern is no more included, or $R_j = R_{\min}$ is reached for the regarded hypersphere j. The pattern is not stored.
- Only hyperspheres of different classes are activated by the presented pattern. Radii of activated hyperspheres will be reduced until the pattern is no more included or $R_j = R_{\min}$ is reached. In the first case, pattern k will be stored with $R_{J+1} = ||\boldsymbol{t}_l - \boldsymbol{t}_{J+1}||$, i.e., the radius will extend just to the center of the closest or nearest-neighbor hypersphere l. In the second case, pattern k will not be stored.

With the choice of $R_{\min}$, the storage of vectors close to class borders can be suppressed, thus influencing network resubstitution and generalization properties. Evidently, patterns once stored in the RCE network will never be removed. Just the pattern radii will be reduced until $R_{\min}$ is reached. This means that the size and quality of the achieved network are determined by the order of presentation of training vectors. A probabilistic presorting of sample data for RCE (ProRCE) based on local probability estimation and sorting of the training presentation order proportional to the probability has

proven to be one beneficial extension of the method [2.20]. However, for the regarded application, the focus will be on the extension of RCE to BC and OCC.

Extension of RCE for OCC. As already addressed, it is of practical interest to derive a system that is trained just by available examples of one class and that detects samples from the other class, e.g., production errors or system malfunctions, as deviations from the normal state. An instance of such a system has been introduced in prior work for image processing. The NOVelty detecting ASsociative memory (NOVAS) [2.31] stores a number of multidimensional pixel images and generates for each pixel an internal representation of hyperspheres with uniform radii, which is quite similar to an RCE classifier with BC. The difference is that RCE with BC assigns one problem class as the background class and trains the radii of the remaining classes' hyperspheres according to the correct classification of training patterns from all classes. In case of OCC, no patterns will be available for the background class. So the hypersphere radii must be determined by an additional rule or method. RCE can heuristically be adapted to that aim by storing all selected examples of the normal class from the training set based on a prior computation of a radius $R_{\max}$ for all hyperspheres according to the maximum distance of two nearest neighbors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in the normal class [2.31]:

$$R_{\max} = \max_{j=1}^{N}(\min_{\substack{i=1 \\ i \neq j}}^{N} ||\boldsymbol{x}_i - \boldsymbol{x}_j||). \tag{2.2}$$

After $R_{\max}$ computation, the normal training data can be completely stored as classifier reference data of the novelty classifier (NOVCLASS). Data vectors $\boldsymbol{x}_l$ from the monitored process can be classified with regard to their novelty by the following steps:

1. Compute the nearest neighbor $\boldsymbol{t}_{NN}$ of $\boldsymbol{x}_l$ in the prototype set $\mathbf{T}$ with:

$$d_{t_{NN}} = \min_{j=1}^{N}(\sum_{i=1}^{M}(x_{li} - t_{ji})^2). \tag{2.3}$$

2. Classify the pattern $\boldsymbol{x}_l$ as:

$$\boldsymbol{x}_l \quad \text{is} \quad \begin{cases} \text{normal} & \text{for} \quad (\sum_{i=1}^{M}(x_{li} - t_{NN\ i})^2) < R_{\max} \\ \text{novel} & \text{for} \quad (\sum_{i=1}^{M}(x_{li} - t_{NN\ i})^2) \geq R_{\max}. \end{cases} \tag{2.4}$$

The resulting novelty detection can be employed to perceive process deviations and filter data out as representing an important event worth storing. Deviations or anomalies are detected as patterns on the background, outside of the normal domain, similar to the BC mode of RCE in multiclass problems. This is illustrated for the two-dimensional case in Fig. 2.9. Employing an iterative presentation of the training data, data reduction in terms of stored vectors, similar to the original RCE classifier, could be achieved, trading off alleviation of storage requirements and real-time classification with

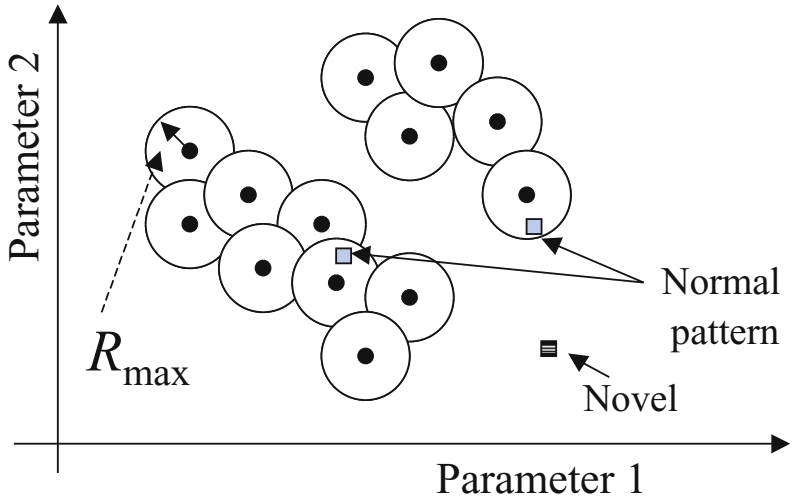


Fig. 2.9. Principle of OCC by NOVCLASS.

sufficient covering of the normal domain. In this iterative training case, a new hypersphere with center $\boldsymbol{t}_{J+1} = \boldsymbol{x}_l$ and radius $R_{\max}$ is added to the initially empty classifier iff a presented vector $\boldsymbol{x}_l$ from the training set is classified as novel by the already stored J reference vectors $\boldsymbol{t}_j$ according to the basic steps given earlier. The denseness of the NOVCLASS model potentially can be controlled by scaling the $R_{\max}$ parameter by a scale factor η to $\eta \times R_{\max}$ in the training process. A large-scale factor implies few stored vectors and potential coarse window modeling, whereas a small-scale factor $\eta < 1$ means fine window modeling at the cost of storing and processing a potentially large number of vectors. A functional nonparametric classifier is thus achieved with examples of just one class. Additionally, if at least a few examples for anomalies are available, these can be used to fine-tune the radii of the stored normal class hyperspheres by applying RCE-like adaptation for the conflicting hyperspheres. In this case, radii will no longer be uniform.

Currently, a prototype NOVCLASS version has been implemented and validated with modified Iris data, where all examples of class 3 were affiliated to class 2. Class 1 was chosen as the normal class. Resubstitution of the training set was perfect and in generalization just one vector slightly separated from the main cluster was misclassified.

Summarizing, the NOVCLASS algorithm allows both data reduction and arbitrary coverage of the parameter space. Thus, the concept of the process window is generalized to arbitrary shapes, including no convex boundaries. The current rather ad hoc uniform $R_{\max}$ computation approach could be improved by more sophisticated methods, e.g., locally adaptive radii computation, in future work. The present NOVCLASS implementation will be

applied to semiconductor application data for basic feasibility demonstration in Section 2.4.

2.3.2 Dimensionality Reduction and Interactive Visualization

Motivation. In addition to semiconductor manufacturing, a wide variety of other technical problems are characterized by typically large sets of high-dimensional data, obtained, e.g., from sensor registration, medical laboratory parameters, manufacturing process parameters, financial databases, measurements, or other generally observed features. With regard to the given application, significance, correlations, redundancy, and irrelevancy of the variables x_i are a priori unknown. The extraction of underlying knowledge or the reliable automatic classification requires reduction of the initial data set to the essential information and the corresponding variables. This especially holds, as the well-known curse of dimensionality (COD) [2.12] makes the compaction of the data a mandatory prerequisite for reliable decision making. Unsupervised and supervised methods can be employed for this reduction step for interactive and automatic processing of the data. The exploitation of the remarkable human perceptive and associative capabilities for the complex problem of identifying nonobvious correlations, structure, and hidden knowledge in the data can be a powerful complement of existing computational methods. Of course, an appropriate visual representation is required, which can be achieved by means of dimensionality reduction or multivariate projection methods combined with interactive visualization of the data [2.49]. Typical database representation, e.g., as an Excel spread-sheet is not easily amenable to human perception and understanding. This is illustrated in Fig. 2.10, together with the alternative human-adapted visual representation of the same database. Thus, dimensionality reduction is a ubiquitous problem and together with multivariate data visualization a topic of interest and interdisciplinary research for more than three decades. Applications of high economical interest, e.g., the one investigated in this work and other data mining and knowledge discovery applications, give renewed strong incentive to the field. Numerous methods were derived in the past for dimensionality reduction that considerably differ with regard to the methodology, computational complexity, transparence, and ease of use. In this work, effective methods promising the best productivity increase will be preferred. The following common definitions of two main groups of dimensionality reduction methods, briefly adapted from [2.16], shall clarify the pursued objectives. For a given sample set $\mathbf{X}$ with N M-dimensional feature vectors $\mathbf{x} = [x_1, x_2, \ldots, x_M]^T$ feature extraction is defined as a transformation

$$J(\mathbf{A}) = \mathbf{max}_{\mathcal{A}} \mathbf{J}(\mathcal{A}(\mathbf{v})) \tag{2.5}$$

and the special case of feature selection is defined as a transformation

$$J(\mathbf{A^S}) = \mathbf{max}_{\mathcal{A}^S} \mathbf{J}(\mathcal{A}^S). \tag{2.6}$$

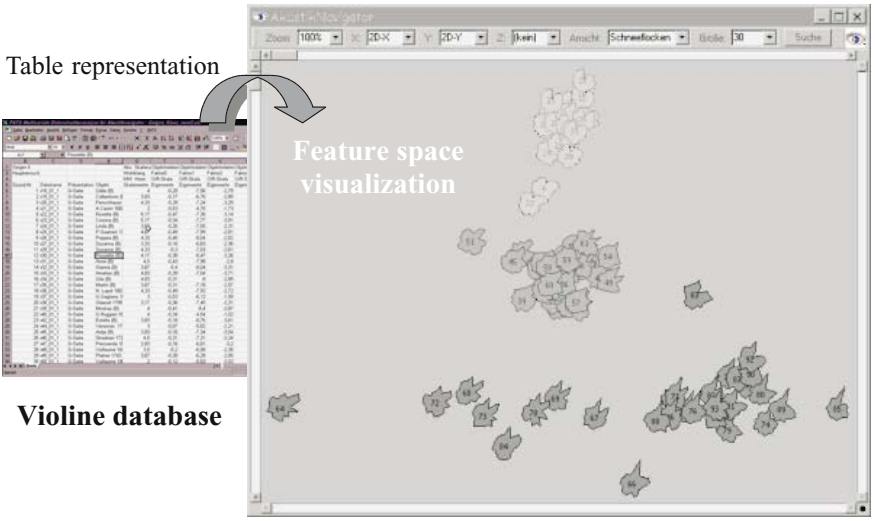


Fig. 2.10. Exploitation of human perceptive capabilities by appropriate presentation of multivariate data employing dimensionality reduction and interactive visualization.

While in selection, according to a chosen criterion J and the applied selection matrix A^S (see Eq. 2.13), the best features are retained and the remaining ones are discarded; in extraction all features are retained and subject to transformation A. In both cases a mapping $\Phi : R^M \rightarrow R^m$ optimizing a criterion J with $m \leq M$ and $\mathbf{y} = [y_1, y_2, \ldots, y_m]^T$ is determined. Here $\mathbf{y} = \mathcal{A}(\mathbf{v})$ can be a linear or nonlinear mapping and employ unsupervised as well as supervised information. The optimization criterion or cost function J can represent various objectives, e.g., signal preservation, distance preservation, topology preservation, or discrimination gain for the underlying L-class problem (see Fig. 2.12). For the latter case, selected instances of J will be given in the following. Figure 2.11 gives a taxonomy of state-of-the-art dimensionality reduction methods for multivariate data classification, analysis, and visualization in a unified presentation. This taxonomy has been elaborated on in the last few years and is continuously enhanced, including new methods. Most of the methods have been implemented in the QuickCog system [2.28] [2.29] and compared in previous survey publications [2.24] and tutorials [2.29] [2.22]. The taxonomy given in Fig. 2.11 covers methods as, e.g., the principal-component analysis (PCA) [2.12], scatter matrices (SCM) [2.12], Sammon's nonlinear mapping (NLM) [2.48], and accelerated heuristic variants, the nonlinear discrimination analysis method of Koontz and Fukunaga [2.32], or Kohonen's self-organizing map [2.19] (see also [2.24]). For visualization purposes, in this work distance-preserving nonlinear mappings, e.g., the one introduced by Sammon [2.48] have been applied. Interpoint distances

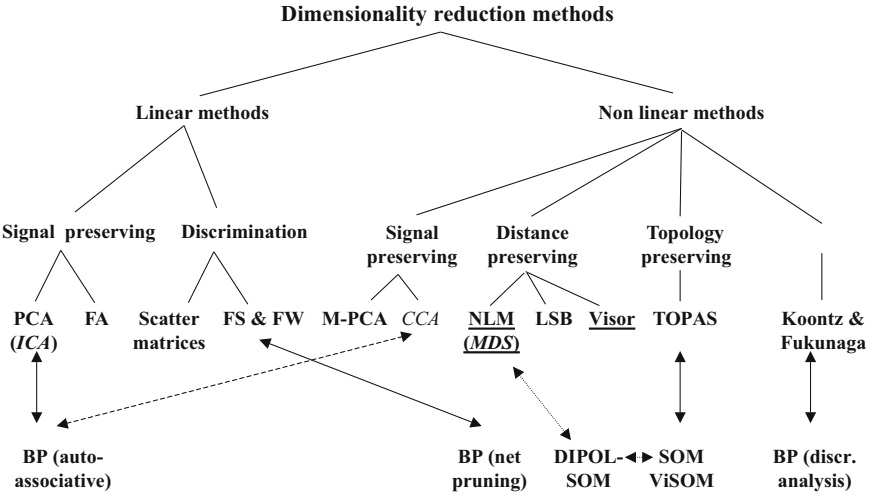


Fig. 2.11. Taxonomy of dimensionality reduction methods.

d_{Xij}, and, thus, implicitly the data structure, shall be preserved in the NLM according to the cost function $E(m)$:

$$E(m) = \frac{1}{c} \sum_{j=1}^{N} \sum_{i=1}^{j} \frac{(d_{Xij} - d_{Yij}(m))^2}{d_{Xij}}. \tag{2.7}$$

Here

$$d_{Yij}(m) = \sqrt{\sum_{q=1}^{d} (y_{iq}(m) - y_{jq}(m))^2} \tag{2.8}$$

denotes the distance of the respective data points in the visualization plane and

$$d_{Xij} = \sqrt{\sum_{q=1}^{M} (x_{iq} - x_{jq})^2} \tag{2.9}$$

in the original data space and

$$c = \sum_{j=1}^{N} \sum_{i=1}^{j} d_{Xij}. \tag{2.10}$$

Based on a gradient descent approach, the new coordinates of the N pivot vectors in the visualization plane $\mathbf{y}_i$ are determined by:

$$y_{iq}(m+1) = y_{iq}(m) - \mathrm{MF} * \Delta y_{iq}(m) \tag{2.11}$$

with

$$\Delta y_{iq}(m) = \frac{\partial E(m)}{\partial y_{iq}(m)} \Bigg/ \left| \frac{\partial^2 E(m)}{\partial y_{iq}(m)^2} \right| \quad and\ 0 < MF \leq 1. \tag{2.12}$$

In particular for large databases, due to the underlying computational complexity of the standard methods, e.g., the NLM with $O(N^2)$, mapping computation becomes infeasible. Therefore, particular interest was placed on heuristic and hierarchical methods of dimensionality reduction as mapping accelerators.

One of the first heuristic accelerating methods of the NLM was published by Lee, Slaggle, and Blum [2.34]. Rightly assuming that the gradient procedure does not always achieve an accurate projection (cf., e.g., [2.9]), they developed a fast distance-preserving mapping that focuses on the exact preservation of only a limited number of $2N - 3$ distances, neglecting all remaining ones. For this mapping, the minimum spanning tree (MST) of the data distance graph is computed. Points are mapped by common triangulation while traversing the MST, based on the previously mapped MST neighbors serving as pivot point. However, in spite of the appealing heuristic idea, MST computation and traversal itself still has $O(N^2)$ complexity. Thus, in own prior work, an even faster mapping algorithm was developed [2.26]. This alternative mapping, denoted as Visor mapping, also uses a triangulation mapping step, but with three fixed global pivot points that are heuristically chosen from the data set. The purpose of the pivot point determination is to find the three most extruded data points that meet the additional constraint of maximum mutual distance while enclosing the remaining data set. Based on centroid computation, these three data points are successively selected as pivot points from the data set. These points are placed first and the remaining N-3 data points are placed in the visualization plane employing triangulation.

This algorithm, denoted by Visor [2.26], has O(N) complexity and thus provides data projections with a very short response time and negligible sensitivity to the database size. As shown by prior investigations with a mapping quality measure, achievable mapping quality is similar to the NLM [2.26], [2.24]. Due to their salient properties with regard to speed, convenience, and transparence, distance-preserving mappings have been applied throughout this work to the regarded semiconductor manufacturing data. In addition, efficient hierarchical methods, offering a more delicate speed-accuracy tradeoff are available [2.23] and will be employed in the next stages of the work.

The unsupervised mapping methods discussed so far retain all features from the high-dimensional feature space and compute a more compact optimized feature space, e.g., for visualization and analysis purposes.

In contrast to this, feature selection actually helps to discard incoming variables that have no or little significance for the tackled problem. It must be remembered, that two very different aims can be pursued by the method of feature selection. For classification tasks, the selection of an as-small-as-possible group is desired, to allow generalization with a minimum classifica-

tion error. For data analysis, the discovery of all involved variables and the underlying knowledge are aspired. Feature selection can be understood as the computation of a constrained matrix A^S for a linear mapping with the following form

$$A^S = \begin{pmatrix} c_1 & 0 & \cdots & 0 \\ 0 & c_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & c_M \end{pmatrix}, \tag{2.13}$$

where only diagonal elements can have nonzero values and the $c_i \in \{0, 1\}$ are binary variables or switch variables determined by a preceding optimization process. Thus, a linear mapping $\mathbf{y} = A\mathbf{x}$ is constituted. However, due to the constrained matrix A and the fact that column vectors with $c_i = 0$ can be entirely omitted, computation can be simplified to $\mathbf{y} = [y_1, y_2, \ldots, y_m]^T$ with $y_i = x_j \quad \forall c_j \neq 0$, i.e., m corresponds to the number of $c_j \neq 0$ and the corresponding features x_j are just copied to the y_i. Feature selection performs a scaling of feature or coordinate axes by binary variables, i.e., switching off dimensions and thus defining a subspace that is salient with regard to the chosen criterion J. As no rotation of the basis vectors is carried out, explicit interpretability of the result is sustained. However, due to the binary nature of the selection process, the difference in importance or the impact of individual features is occluded. A straightforward extension of the binary matrix A^S given for feature selection is feasible, which allows continuous valued ranking of the features. The binary c_i are replaced by real variables $a_i \in [0, 1]$, which are determined by a preceding optimization process. The limitation or normalization to $[0, 1]$ is introduced for the sake of interpretability and comparison with corresponding feature selection results. This approach commonly denoted by feature weighting (FW) allows a continuous scaling of features or coordinate axes for $a_i \neq 0$. Those columns with $a_i = 0$ can be omitted, reducing the matrix from $M \times M$ to $M \times m$ with $m \leq M$. Thus, in addition to the aspired potentially higher achievable discrimination and better generalization properties, explicit salient information for data analysis purposes and rule weighting is extracted by this method. One particular method of finding appropriate a_i based on a certain cost function J and a gradient descent technique can be found in [2.21]. Numerous other options with regard to the chosen J and the optimization strategy, e.g., evolutionary computation, are feasible [2.45] and are currently being pursued in ongoing work. Various strategies and methods for feature selection will be discussed after presentation of relevant cost functions J.

Cost Functions. In the following, from a larger collection of potential cost or assessment functions summarized in Fig. 2.12, dedicated cost functions for feature space assessment introduced in prior work, e.g., [2.27] and [2.28] [2.22], will be briefly presented for the aim of a self-contained presentation. These serve for discrimination measuring in terms of class regions separability,

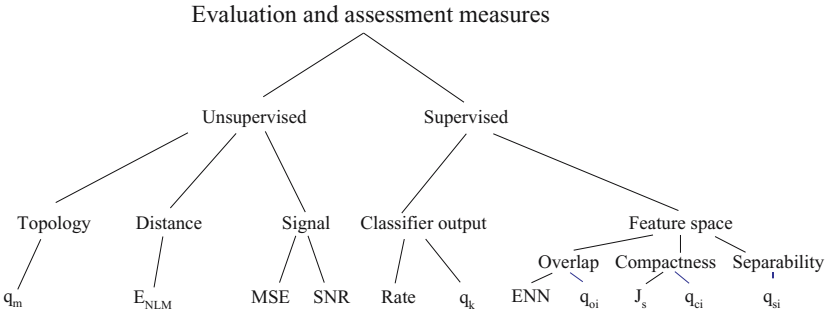


Fig. 2.12. Taxonomy of cost functions.

overlap, or compactness in the regarded feature space and ensuing systematic dimensionality reduction. Though the classification rate or a posteriori probabilities of any classifier could serve here (cf, e.g., [2.16] or [2.45]), for obvious practical reasons, robust measures nearly free of required parameters, model assumptions, and intricate training requirements are preferred in this work.

For instance, to measure separability, a nonparametric measure q_s exploiting nearest-neighbor techniques can be computed. For this class separability assessment, the RNN-classifier [2.13] is exploited, which iteratively selects a subset of relevant vectors as reference vectors from the training set, as the number of these selected reference vectors T_{RNN} is proportional to the feature space separability. This is illustrated in Fig. 2.13, where selected reference vectors T_{RNN} are emphasized in bold. In the case of linear separability of class regions, one vector per class region would be required. So the quality measure given by

$$q_s = \frac{N - (T_{RNN} - L)}{N} \tag{2.14}$$

has 1.0 as its optimum value indicating linear separability. An improved variant of q_s takes significantly different a priori probabilities in account:

$$q_{si} = \frac{1}{L} \sum_{i=1}^{L} \frac{N_i - (T_{RNN_i} - 1)}{N_i}. \tag{2.15}$$

Here N_i denotes the number of patterns affiliated to class ω_i and T_{RNN_i} the number of reference vectors selected for class ω_i. (It is assumed here that N_i corresponds to the actual a priori probability of class ω_i). The quality measures q_s and q_{si} have O(N) complexity and thus are very fast; however, the resolution is quite coarse, which can be detrimental for optimization schemes. Numerous feature space configurations can be mapped on the same assessment value.

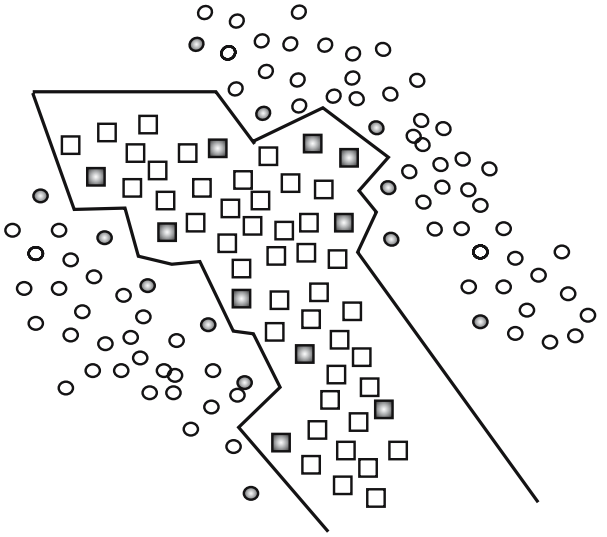


Fig. 2.13. Class separability assessment.

A very simple parametric measure for overlap computation was introduced in [2.49]. The class specific distributions are modeled by Gaussian functions and an overlap of two-class regions, denoted by ω_i und ω_j, can be computed from the respective mean values μ_i, μ_j and standard deviations σ_i, σ_j by

$$q_{x_{l_{ij}}} = \frac{|\mu_i - \mu_j|}{(N_i - 1)\sigma_i + (N_j - 1)\sigma_j}. \tag{2.16}$$

The merit of a feature for the separation of one class from all others is given by

$$q_{x_{l_i}} = \frac{1}{L-1} \sum_{j \neq i}^{L} q_{x_{l_{ij}}}. \tag{2.17}$$

Also, the merit of a single feature to distinguish all classes could be computed by

$$q_{x_l} = \frac{1}{L} \sum_{i=1}^{L} q_{x_{l_i}}. \tag{2.18}$$

However, practical experience has shown that the global summation can be misleading in some cases. A feature can, for instance, be excellent for certain class separations and meaningless for most others but have a summation value that outperforms other features that are good everywhere in feature space.

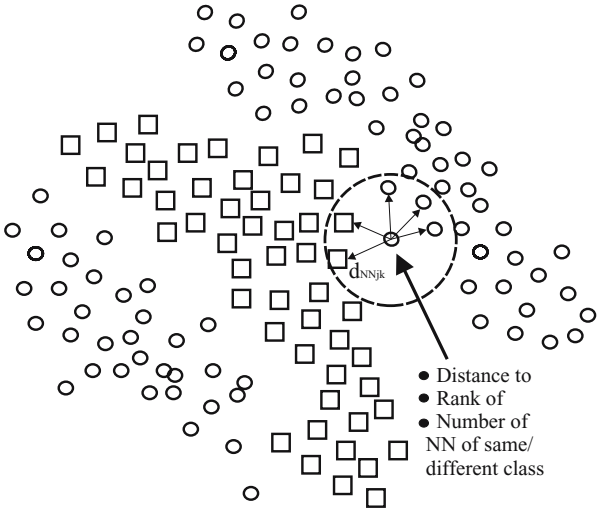


Fig. 2.14. Class overlap assessment.

Proposals for efficient application of these simple measures will be given in the following sections on feature selection strategies.

A nonparametric overlap measure q_o, which was inspired by the edited-nearest-neighbor (ENN) algorithm [2.8], in contrast to q_s, provides a very fine-grained value range and thus is better suited for optimization schemes. However, the price tag is an increased complexity of $\mathrm{O}(N^2)$ with regard to q_s. The basic idea of q_o is illustrated in Fig. 2.14. The overlap measure q_o is computed by:

$$q_o = \frac{1}{N}\sum_{j=1}^{N}\frac{\sum_{i=1}^{k} q_{NN_{ji}} + \sum_{i=1}^{k} n_i}{2\sum_{i=1}^{k} n_i} \tag{2.19}$$

with

$$n_i = 1 - \frac{d_{NN_{ji}}}{d_{NN_{jk}}} \tag{2.20}$$

and

$$q_{NN_{ji}} = \begin{cases} n_i & : \quad \omega_j = \omega_i \\ -n_i & : \quad \omega_j \neq \omega_i. \end{cases} \tag{2.21}$$

Here, n_i denotes the weighting factor for the position of the ith nearest neighbor NN_{ji}, $d_{NN_{ji}}$ denotes the distance between $\mathbf{x}_j$ and NN_{ji}, $d_{NN_{jk}}$ denotes

the distance between $\mathbf{x}_j$ and most distant nearest neighbor NN_{jk}, $q_{NN_{ji}}$ denotes the measure contribution of $\mathbf{x}_j$ with regard to NN_{ji}, and ω_j and ω_i denote the class affiliation of $\mathbf{x}_j$ and NN_{ji}, respectively. The influence of a nearest neighbor in the quality measure decays with its rank position to $n_i = 0$ for NN_{jk}. The final measure q_o is fine-grained and sensitive to small changes in the feature space. Further q_o is also normalized in [0,1], where 1.0 indicates no overlap in the feature space. Typically, 5 to 10 nearest neighbors are well suited for computation of this quality measure. Simplification of the measure is feasible, trading off fine-grained resolution in overlap computation and, thus, sensitivity to small changes in the feature space against computational savings. An improved variant of q_o takes significantly different a priori probabilities into account

$$q_{oi} = \frac{1}{L}\sum_{c=1}^{L}\frac{1}{N_c}\sum_{j=1}^{N_c}\frac{\sum_{i=1}^{k} q_{NN_{ji}} + \sum_{i=1}^{k} n_i}{2\sum_{i=1}^{k} n_i}. \tag{2.22}$$

Finally, compactness q_c can be measured by explicitly computing the ratio of current intra- and interclass distances. Implicitly this criterion is also used in the computation of scatter matrices [2.12]. The compactness q_c previously introduced in [2.22] suffers from the flaw that the measure will be optimum, if the majority of intraclass distances will be made small, i.e., class regions with the majority of patterns will dominate the assessment and consequently any optimization process based on the measure q_c. An improved measure q_{ci} for different a priori probabilities and corresponding N_l in the L-class problem can be obtained by class-specific normalization during compactness computation

$$q_{ci} = \frac{\frac{1}{L}\sum_{l=1}^{L}\frac{2}{N_l(N_l-1)}\sum_{i=1}^{N}\sum_{j=i+1}^{N}\delta(\omega_i,\omega_j)\delta(\omega_i,l)d_{X_{ij}}}{\frac{1}{N^B}\sum_{i=1}^{N}\sum_{j=i+1}^{N}(1-\delta(\omega_i,\omega_j))d_{X_{ij}}} \tag{2.23}$$

with

$$d_{X_{ij}} = \sqrt{\sum_{q=1}^{M}(x_{iq}-x_{jq})^2} \tag{2.24}$$

and $\delta(\omega_i,\omega_j)$ is the Kronecker delta, which is $\delta(\omega_i,\omega_j) = 1$ for $\omega_i = \omega_j$, i.e., both patterns have the same class affiliation, and $\delta(\omega_i,\omega_j) = 0$ elsewhere. Also, $\delta(\omega_i, l)$ prescribes that only distances with $\omega_i = \omega_j = l$ are accumulated for the lth-class sum of intraclass distances. Further, the normalization factor N^B is given by

$$N^B = \sum_{i=1}^{N}\sum_{j=i+1}^{N}(1-\delta(\omega_i,\omega_j)). \tag{2.25}$$

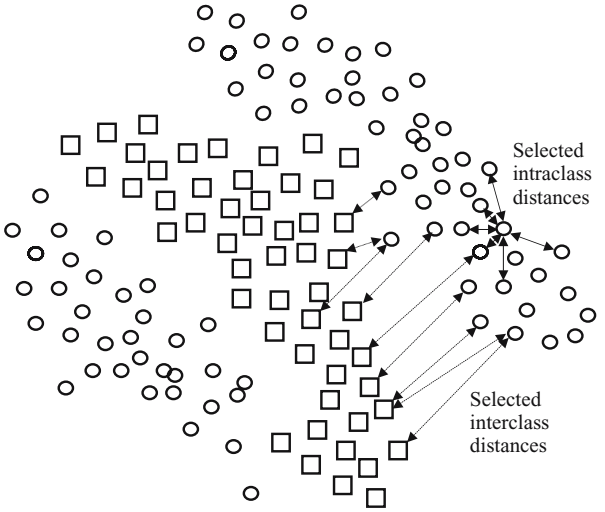


Fig. 2.15. Class compactness assessment.

The principal idea of intraclass and interclass distance computation for q_{ci} is illustrated in Fig. 2.15. The improved compactness q_{ci} has a complexity of $O(N^2)$, is a nonparametric measure, and requires no parameters to be set by the user. It shows a high sensitivity to changes in feature space, as these are immediately mirrored by changes in distance, and, thus, in changes in q_{ci}. In comparison to the existing overlap measure q_{oi} with equal sensitivity and computational complexity, q_{oi} is inferior, as it requires the parameter k to be set. But q_{oi} is superior with regard to normalization properties, returning a value in [0,1], whereas q_{ci} values depend on the distances in the data set and only allow the observation of relative changes. For FS, an additional normalization step for each selection or configuration is required for q_{ci}.

These measures will serve in the following as feature space assessment or ranking measures J. Information on the individual features' merit as well as the current feature combinations' merit can be obtained by employing the presented measures. Also, the results of different dimensionality reduction methods, e.g., FS or FW, can be quantitatively compared and assessed [2.24].

Feature Selection Methods. The process of finding the appropriate coefficients c_i in (Eq. 2.13) is an intricate optimization problem. Due to the combinatorial complexity inherent to the problem of FS, the computational effort of finding the best selection, i.e., feature combination, grows exponentially. Thus, the global optimum solution for the selection process cannot be found with polynomial complexity or effort, i.e., we have an NP-complete problem (cf., e.g., [2.1]). Therefore, a complete or exhaustive search of all

feature combinations in general is out of the question. Several alternative search strategies for FS, employing the cost functions from Section 2.3.2, will be summarized with regard to achievable performance and required computational effort.

First-Order Selection Techniques. One simple but often effective way of finding a suboptimum solution with minimum effort is to compute an individual figure of merit for each feature. This first-order approach neglects possible higher-order correlations between feature pairs or feature tuples. For assessment or figure of merit computation, for instance, one of the cost function given in the previous subsection has to be applied. However, the cost function in this simplified case will be computed separately for each feature. Three permutations are basically feasible:

- The figure of merit is computed for a selected feature and a selected combination of classes, i.e., the feature contribution to pairwise class discrimination is assessed. For instance, the measure $q_{x_{l_{ij}}}$ could be computed here. For each class pair, features are ranked according to their individual merit. Selection from these rank tables can be achieved, for instance, by choosing all features in first-rank position. Table 2.1 gives an example of this first-order selection scheme for the well-known Iris data. Obviously, for first-rank position **R**, features 3 and 4 will be selected. The method can be computed very quickly, but the rank table grows for given feature number M and class number L by $M * (L(L-1)/2)$.
- The figure of merit is computed for a selected feature and for the discrimination of one class versus all others. The corresponding rank table grows for given feature number M and class number L by $M * L$.
- Computing the figure of merit with regard to discriminating all classes for each feature returns a single column with M elements.

As shown in Table 2.1, the parametric overlap measure $q_{x_{l_{ij}}}$ and its variants can serve for the three approaches of fast first-order feature selection. If the parametric assumption is met, then this simple scheme can be very effective. However, in many practical cases, even for the one-dimensional distributions of the individual features, a nonparametric nature can be observed. An effective remedy for this situation is the application of, e.g., the overlap

Table 2.1. Rank table from first-order assessment for Iris data.

Feature	R	C 1-2	R	C 1-3	R	C 2-3
x_1	4	1,020	3	1,482	3	0,442
x_2	3	1,065	4	0,890	4	0,255
x_3	2	4,139	1	**5,451**	2	1,218
x_4	1	**4,387**	2	5,180	1	**1,660**

Table 2.2. Rank order for first-order feature selection computed for visual inspection feature data based on parametric (left column) and nonparametric (right column) assessment measure.

Feature	R	C 1-2	R	C 1-2
x_1	5	0,2917	5	0,6977
x_2	4	0,6489	3	0,8211
x_3	1	1,1558	4	0,7058
x_4	3	0,8547	2	0,8808
x_5	2	1,0047	1	0,9270

measure q_o (or q_{oi}) separately for each individual feature. This returns a corresponding nonparametric measure to the parametric one given earlier. For Iris data, the selection will be identical. In Table 2.2, however, a feature set computed from images of a practical visual inspection problem is subject to both the parametric and the nonparametric first-order feature selection scheme.

For the regarded nonparametric example data set only the nonparametric measure provides the a priori known correct solution. Summarizing, first-order selection schemes are a special case of heuristic approaches to find solutions to the otherwise NP-complete feature selection problems. Suboptimum solutions can be found at very low computational costs. Employment of the nonparametric measure provides more robustness due to the relaxed distribution assumption at moderate cost increase, which is dependent on the sample set size with $O(N^2)$. Further, the simple first-order selection could be employed to weed out variables, which already possess distinct meaning for themselves, and apply more complex search strategies on the residual variables.

Higher-Order Selection Techniques. Higher-order correlations or dependencies of features require the computation of the feature merit with regard to a tuple of other features. In the limit, the effect of a certain feature with regard to all other features has to be considered. As mentioned before, this is a problem of combinatorial optimization and the best possible solution, i.e., the global optimum can be found by exhaustive search. Due to the exponential increase of possible combinations, which grow by 2^M for the binary selection problem and the number M of features, and the underlying NP-completeness of the problem only for small to moderate M is an exhaustive search feasible.

Let us assume that computation of the assessment measure q_o, which depends on the sample set size N with $O(N^2)$, takes one second on a standard computer. Then an exhaustive search for $M = 12$ will consume $2^12 = 4096$ seconds, which amounts approximately to 1 hour and 8 minutes of computation time. For $M = 16$, more than 18 hours of computation time will be required. It is obvious that for larger databases, either for classification or for

data analysis, the employment of exhaustive search, and thus the guaranteed finding of the global optimum, will be infeasible.

In addition to first-order selection schemes, for more features, heuristic search strategies employing tree search schemes, e.g., Sequential Forward/Backward Selection (SFS/SBS) were devised [2.16]. These are also implemented in the method collection and corresponding toolbox within the QuickCog system [2.28]. These heuristic approaches systematically reduce the number of searched and assessed feature combinations. As many combinations are left out of consideration, the global optimum can be missed, and convergence to just a local optimum solution for the selection problem is guaranteed. In SFS, for instance, initially no features are selected. Now each of the N features is tentatively selected and its effect on the figure of merit, e.g., class regions overlap, is computed. The feature with the best assessment is permanently selected and frozen. The same procedure is iteratively repeated for the remaining $(N-1)$ features until only one feature can be altered. Now either the feature combination with the best assessment value can be selected, regardless of the number of selected features, or for a fixed maximum number of features the row with the best compromise of assessment value and required minimum number of features will be selected. In SBS, the same process starts from the initial condition that all features are selected and get rejected in the process. Figure 2.16 elucidates the SBS process and Table 2.3 shows an example of a selection process protocol for Iris train data using SBS [2.16] and the q_s quality measure [2.28]. As $M*(M-1)/2+M$ combinations have to be assessed in both cases, the computational complexity is given by $O(M^2)$. Thus, for $M = 16$, in this case, a local optimum solution will be found within approximately 4 seconds compared to more than 18 hours for an exhaustive search. Though the finding of a global optimum is not guaranteed, these robust methods provide good solutions quickly, and in practical work the global optimum was often found.[9] Comparing these heuristic methods with the simple first-order selection schemes, it can be stated with some

[9] These were cases where an exhaustive search for result comparison was still feasible.

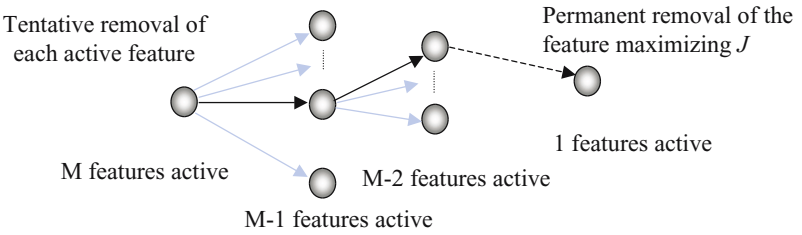


Fig. 2.16. Illustration of SBS feature selection.

Table 2.3. Feature selection protocol for Iris data.

```
Selection strategy:  SBS
Assessment measure:  Separability qs
1 2 3 4   0.90667
- 2 3 4   0.94667
- - 3 4   0.96000
- - - 4   0.00000
Optimum quality     :   0.96
Significant Features:   3 4
```

caution, that the higher-order methods are usually superior. But, of course, it is possible that the simple first-order scheme runs on a configuration that is neglected by the higher-order methods due to the search strategy and returns a better solution. Instead of strict top-down or bottom-up processing, as met in SBS or SFS, an alternation between feature rejection and selection during the search process can be found in other approaches, e.g., branch-and-bound approaches or floating search.

Further heuristic search strategies, employing stochastic methods, e.g., simulated annealing (SA) [2.1] or Boltzmann machines (BM) [2.1], as well as bio-inspired techniques for optimization, e.g., genetic algorithms (GA) in particular and evolutionary strategies (ES) in general, can be applied for FS [2.45], [2.11]. Also, multiobjective optimization can be merged with the GA/ES approach [2.11]. This subject is pursued in ongoing work.

The permanent elimination of redundant and irrelevant features from the sample set by FS provides an effective means of dimensionality reduction. However, the crispness of the selection process can lead to stronger sensitivity with regard to variances in the feature representation in generalization due to the loss of information contained in the discarded features. The issue of the stability of the FS solution and the underlying maximum of the cost function is raised here. It is especially painful for data analysis and knowledge acquisition, if for minor changes in the data entirely different features are selected. The methods discussed so far are specialized to classification problems and require revision and enhancement with regard to stability and data analysis.

Visualization Techniques and Dedicated Tools. In contrast to the state of the art, e.g., static scatter plots, in the methodology pursued in this research work, the achieved projections are the baseline for interactive human analysis. Interactive CAD-like visualization techniques, e.g., interactive navigation, diverse component plots, grid plots, and attribute plots, support human perception and analysis [2.24]. Figure. 2.17 gives a taxonomy of relevant visualization techniques for large high-dimensional data. For instance, at each projection point, the value of a selected variable can be plotted in a Hinton diagram style, i.e., the variable value is coded by the side length

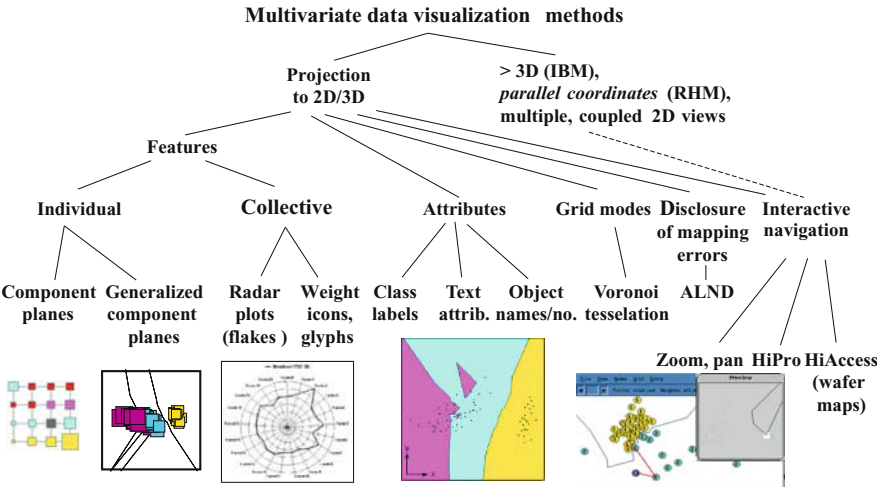


Fig. 2.17. Taxonomy of visualization techniques for high-dimensional data.

or the area of a rectangle. Alternatively, several variables can be plotted by iconified radar plots at each projection point (see Fig. 2.10).

Figure 2.18 (a) shows the underlying multivariate data visualization architecture. Especially the features for accessing database contents from the top-level map should be pointed out here as unique characteristics of the approach. Two implementations have been conceived so far, the general-purpose tool WeightWatcher (WW) in QuickCog (Fig. 2.18 (b)) and the dedicated Acoustic Navigator [2.25] with enhanced interactive features (Fig. 2.18 (c)). Further interactive enhancements are on the way, e.g., interactive selection, labeling, and extraction of arbitrary data from the map. The outlined methods and tools have been compared, assessed [2.24], and employed in numerous scientific and industrial applications. Examples of applicability are given in

- rapid prototyping in the design of recognition systems [2.10];
- analysis of medical databases [2.18];
- analysis of psychoacoustic sound databases with the extension to synthesis in sound engineering [2.25]; and
- analysis and design of integrated circuits with regard to design centering and yield optimization.

For the case of rapid and transparent recognition system design a brief example will be given. A vision system was designed for a medical robot in an object recognition task [2.10]. Dimensionality reduction and interactive visualization approach helped to assess the current system's capability in terms of feature space discrimination and occurrence of pop-outs or outliers. This is illustrated in Fig. 2.19. Additionally, the backtracking capability from the resulting interactive map is illustrated by invoking the original image of a

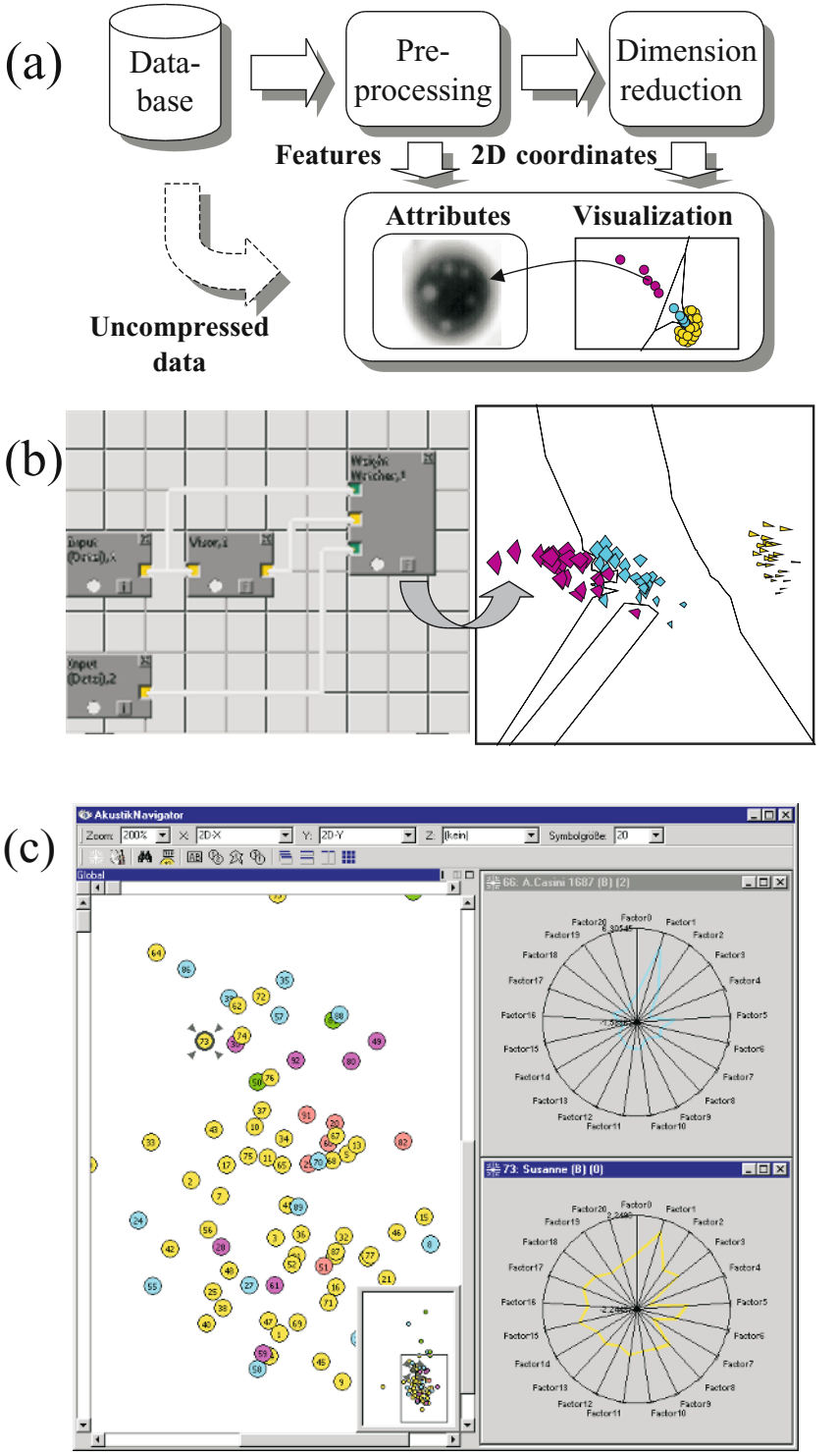


Fig. 2.18. Feature space reduction and interactive visualization: (a) Architecture and dedicated tools; (b)WeightWatcher; and (c) acoustic navigator.

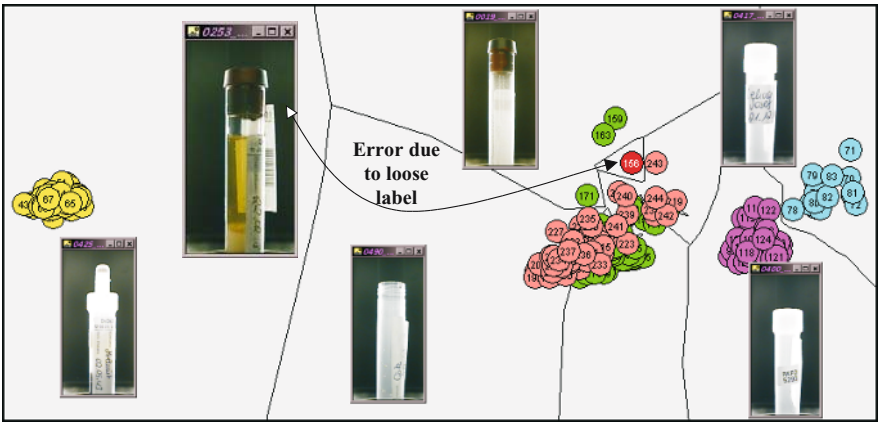


Fig. 2.19. Feature space for vision system of medical laboratory robot.

selected object for each class from the underlying database. Thus, occurring problems, e.g., misclassifications, and underlying causes, can be easily made overt. This alleviates troubleshooting in system design and increases design speed, reliability, and overall productivity. The work is extended to microelectronic manufacturing process data analysis and the features elaborated in prior research and application projects are adapted to this domain. For instance, data entries can be tracked back from the projection in the process database as illustrated in Fig. 2.19 for image data. Thus, the database can be browsed and analyzed according to the inherent clustering and structure in the data. The extension of the existing approach to semiconductor manufacturing will be presented in the following section and in Section 2.5, giving an outline of the envisioned domain-specific system.

2.4 Experiments and Results

The first step of the work in this feasibility study targets the validation and demonstration of the actual practical assistance of the dimensionality reduction and visualization approach to discover structure in and extract knowledge from the industrial high-dimensional database. Thus, it is expected from the visualization that the known split information can be effortlessly retrieved from the map. In this case, unknown clustering in the data, due to detrimental and unintended effects, could also be made overt to the process analyst at a glance.

The most simple and fast Visor projection method was applied to the data first [2.24]. Figure 2.20 shows that distinct yet overlapping clusters can be identified in the data. It is well known from physical and technological background knowledge, that the generated split affects only a fraction of

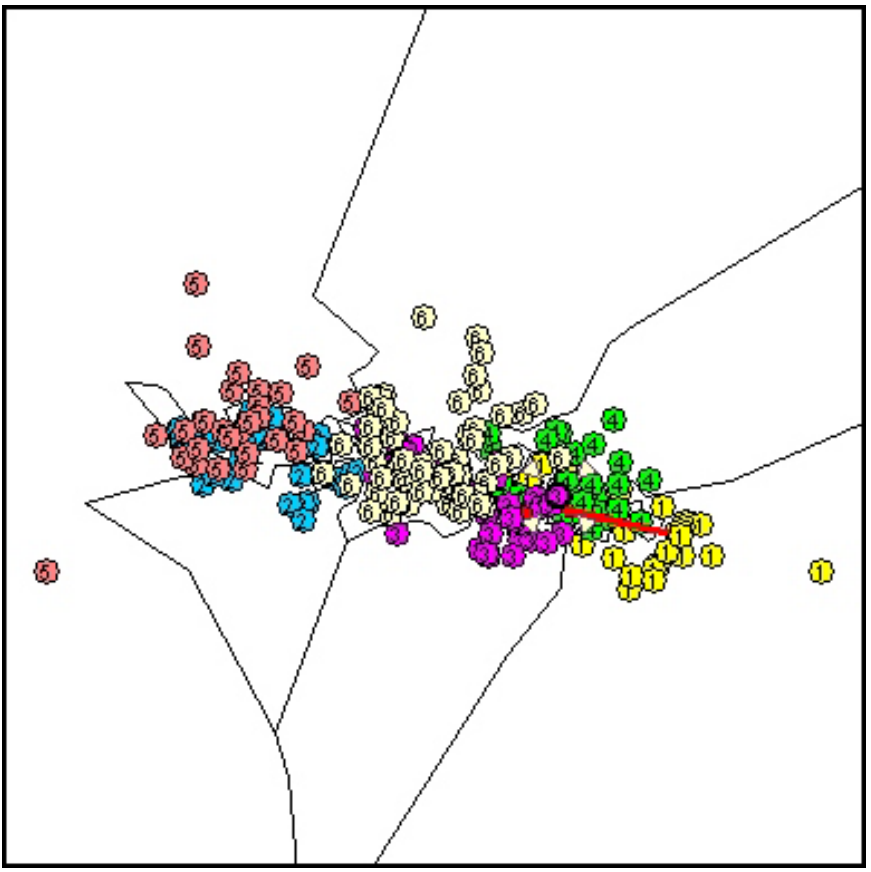


Fig. 2.20. Visualization of SPLIT6.

more than 200 parameters included in the database. Therefore, the observed cluster overlap in this unsupervised mapping approach is related to the quasi-noise of the large number of variables unrelated to the split. However, as in previous application projects, the feasibility of the dimensionality reduction and visualization approach could be shown for the regarded semiconductor manufacturing process.

Additionally, in Fig. 2.21 four selected variables are displayed by component plots. It can be perceived from this representation that the variables C118 and C119 are characteristic for the existing split, whereas C071 distinguishes the lots rather than the split, and finally C063, which is characteristic for neither the lots nor the split.

In addition to the overall visualization of the data, based on unsupervised dimensionality-reducing mapping and all variables, it is of importance to determine which parameters or groups of parameters are conforming with or

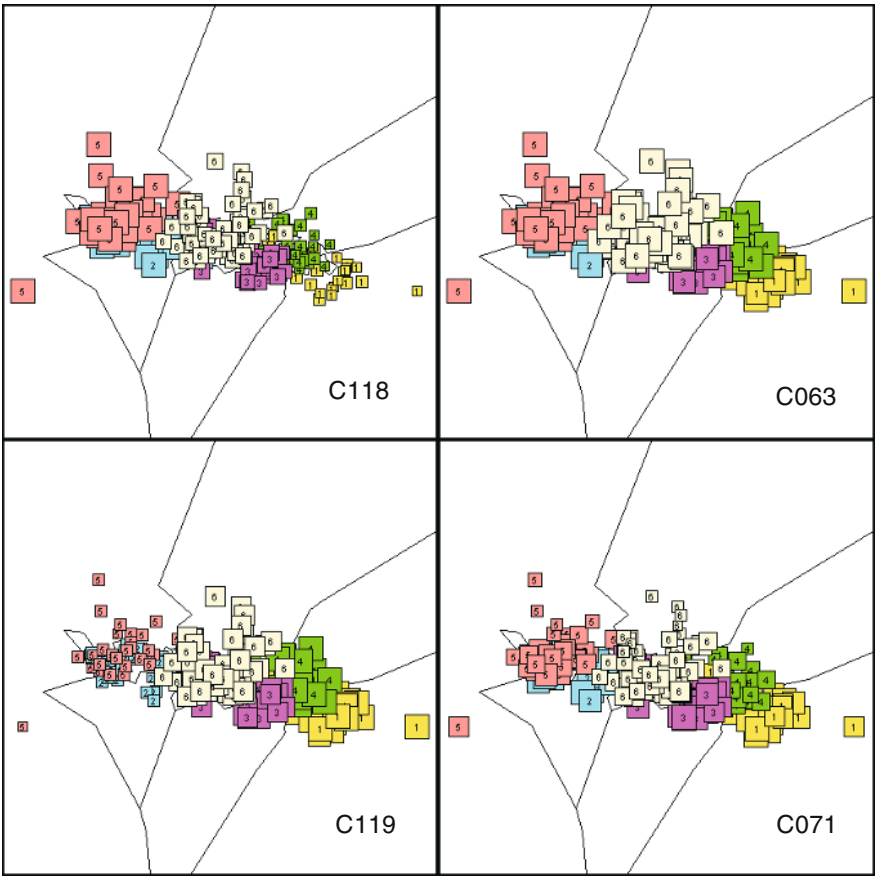


Fig. 2.21. Visualization of SPLIT6 by four selected component plots.

opposed to the existing split. Parameters also might be redundant with regard to this issue. From the available supervised methods, automatic selection of features has been employed to find an answer to this question for the regarded application data. The SBS selection method delivered the best results for the higher-order methods in the conducted experiments. In Table 2.4 the results for lot and split discrimination (SPLIT6) and only split discrimination (SPLIT3) are documented for the three regarded cost functions and the best obtained results.

For instance, application of SBS with q_{si} reduced the SPLIT6 database to just nine parameters. Figure 2.22 shows the resulting projection with nearly linear separability of the data. From the resulting projection in Fig. 2.22, as well as the later Fig. 2.23, the existing asymmetry of the split can clearly be observed, which is a very significant achievement of the regarded visualization method. The expectation, of course, is that the selected parameters are dom-

Table 2.4. FS results for SPLIT6 and SPLIT3.

Selection method	Cost function	Dim.	Chosen features
1rstOP	1. Rank	4	1, 32, 118, 141
SBS	$q_{si} = 0.99487$	9	32, 65, 79, 114, 119, 142, 191, 198, 199
SBS	$q_{oi} = 1.0$	8	32, 65, 86, 129, 131, 142, 191, 201
SBS	q_{ci}	15	78, 79, 114, 115, 118, 119, 120, 121, 126, 129, 140, 141, 142, 143, 144
1rstOP	1. Rank	2	118, 141
1rstOP	1.– 2. Rank	4	118, 120, 140, 141
1rstOP	1.– 3. Rank	6	118, 119, 120, 140, 141, 144
SBS	$q_{si} = 1.0$	1	126
SBS	$q_{oi} = 1.0$	2	118, 205
SBS	q_{ci}	15	78, 79, 114, 115, 118, 119, 120, 121, 126, 129, 140, 141, 142, 143, 144

inantly responsible for the observed split. However, it must be minded that weaker correlations of potential interest for the data analyst are removed by this method, which is tailored to the needs of classification. Only those variables of value for optimum separability or optimum overlap will be chosen. The measure q_{oi} saturated early in the selection process, i.e., the maximum cost function value 1.0 was reached very early, which means the measure lost capability to properly distinguish between the contribution of the remaining variables. Correlating the achieved result with the underlying physical meaning of the variables showed that only a fraction of the relevant variables were identified (see Table 2.5). For comparison purposes, the described first-order method (1rstOP) also has been applied, employing the first highest-ranking variables for pairwise class separation. Some of the relevant variables were found with a significant speed difference compared to the higher-order methods, i.e., seconds vs. several hours on a state-of-the-art PC. However, the method identifies an irrelevant variable, too, and regretfully leaves out of consideration numerous relevant ones.

For SPLIT3, for q_{si} only one and for q_{oi} only two variables were selected. The methods both saturated early in the selection process. In both cases the class regions are not compact and show considerable scatter. Though a lean classification system could be devised from this result for the information gathering and knowledge discovery this results is far from desirable. The application of the 1rstOP delivered similar results for first-rank variables. Only a few of the relevant variables were identified. Increasing the included rank positions, more relevant variables were included (see Table 2.4). However, it is difficult for the user to judge, which parameter value for the rank position should be set to include all relevant variables and avoid irrelevant ones. Also, redundant variables could still be present in the selection. Due to its speed,

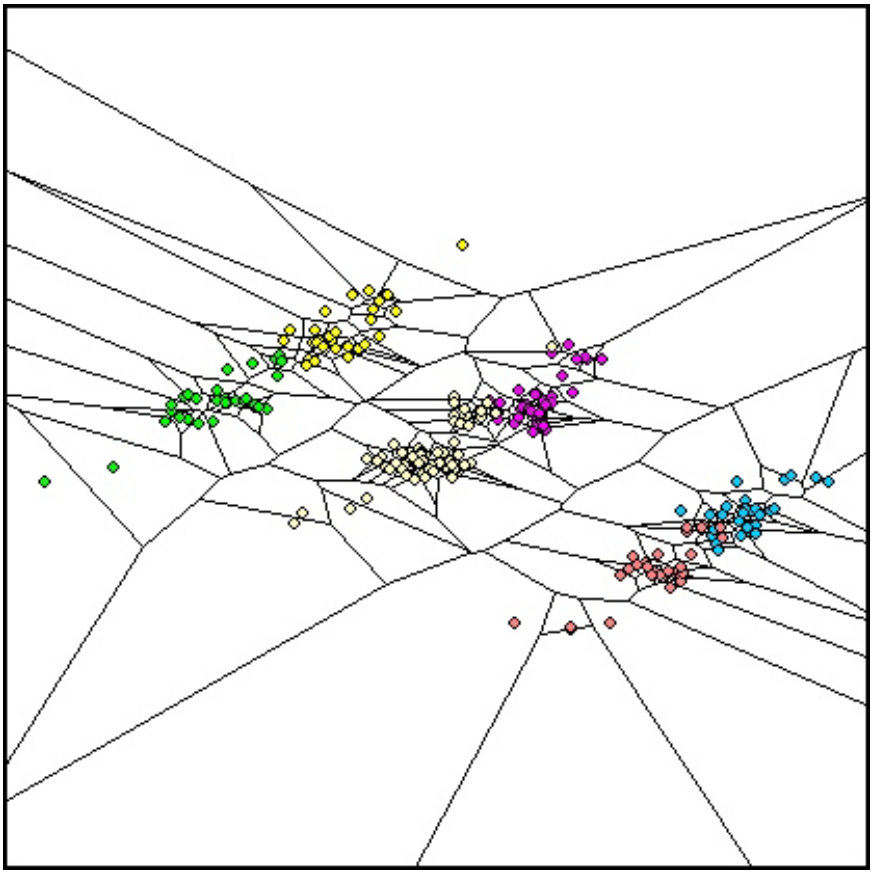


Fig. 2.22. Visualization of selected SPLIT6.

the method could be applied to create a starting solution for a higher-order method in a hierarchical approach. Such a hierarchical approach is considered very promising for future work.

The most meaningful result with regard to identified underlying physical and technological evidence was achieved by the most recent FS variant, employing SBS and q_{ci} for SPLIT6 as well as SPLIT3. Fifteen variables have been selected (see Table 2.4), and a feature space with compact and well-separated class regions is obtained by this selection. Figure 2.23 shows the resulting projection of the 15-dimensional data of SPLIT3, which is definitely superior to the result obtained for q_{si} application. Regarding the underlying physical meaning of the variables, the validity and significance of this selection is underpinned. Table 2.5 explains the meaning of the selected variables for the regarded submicron CMOS process.

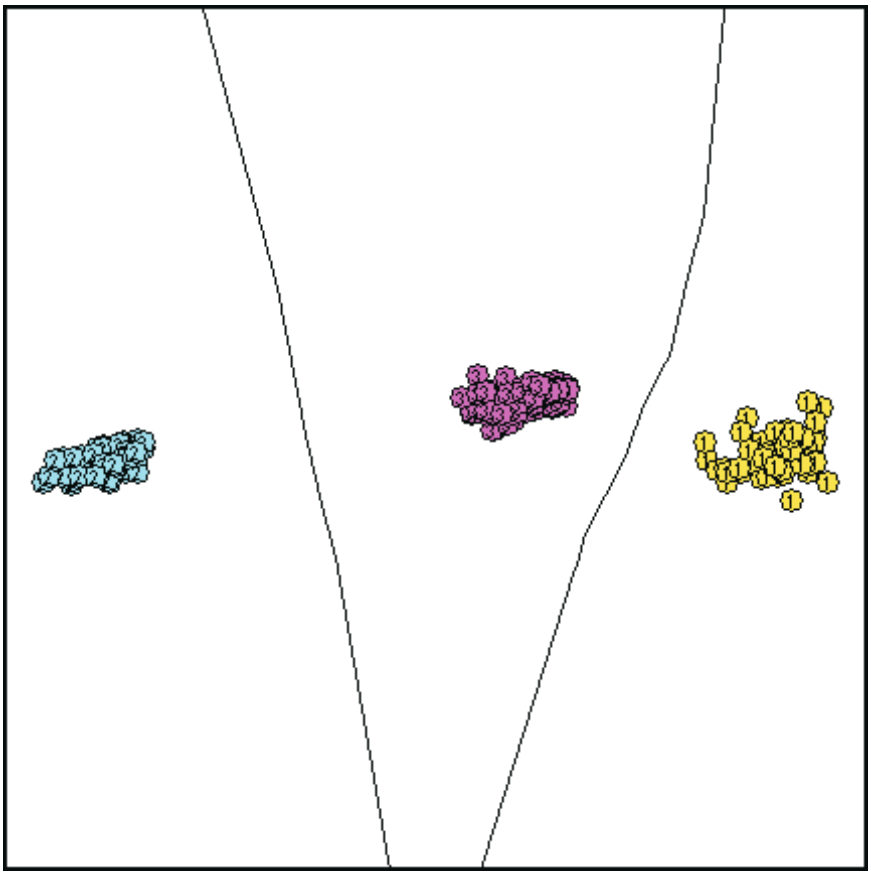


Fig. 2.23. Visualization of selected SPLIT3 according to compactness q_{ci}.

After the regarded steps of interactive visualization, analysis, and automatic determination of relevant variables, the monitoring of the process state by classification methods is investigated. According to the underlying lots, SPLIT6 was separated after feature selection (SBS, q_{si}, 9 features) into a training set, SPLITTrain3, and a test set, SPLITTest3. The six different classes in SPLIT6 were due to the distinguishing of the lots. Splitting SPLIT6 into a training and a test set reduces the classification task to an $L = 3$ class problem. In the first step of this part of the work, the training set was used to train a reduced nearest neighbor classifier (RNN) [2.13]. As can be seen from Fig. 2.24, generalization was perfect and data from the second lot can perfectly be classified according to the three split classes and the features chosen for optimum separability. However, in this approach numerous samples of the novel or abnormal cases were available. In the second step of this part of the work, OCC was applied to the same data. It must be kept in mind

Table 2.5. Physical and technological meaning of selected variables.

Parameter number	Explanation
IO device	
78, 79	Threshold voltages
Logic NMOS device	
114, 118, 120, 129	Threshold voltages
115, 119, 121	Saturation currents
131	Punchthrough current
Logic PMOS device	
140, 143	Threshold voltages
141, 144	Saturation currents
142	Channel leakage
Parameters unrelated to split	
1	Breakdown voltage
32	Saturation current MV device
65	Sheet resistance well
71, 73	Threshold voltage HV devices
191	Gate oxide thickness
198, 199, 201, 205	Sheet resistance poly
Derived parameter	
126	Universal curve FOM

that NOVCLASS only uses the samples affiliated to class 1 of the training set during learning. Thus, samples affiliated to classes 2 and 3 were not involved in the training of NOVCLASS and were unknown to the OCC classifier. In the following, classes 2 and 3 will be merged to class 2, denoting abnormal or novel measurements and respective process states. The aim was to assess the feasibility of NOVCLASS for (semi)automatic significance and novelty data

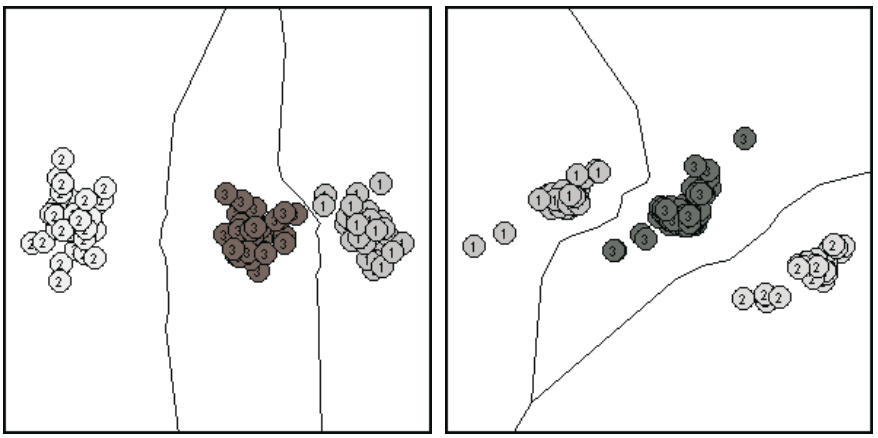


Fig. 2.24. Visualization of selected SPLIT6 classification.

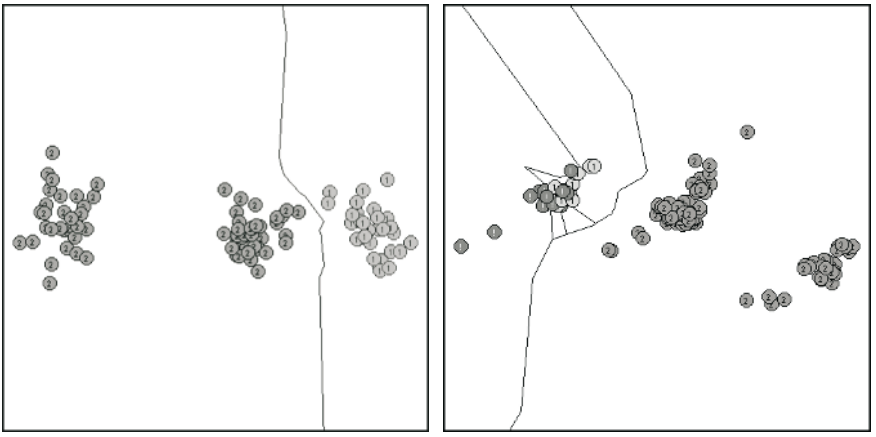


Fig. 2.25. Visualization of selected SPLIT6 novelty classification.

filtering within an information-processing hierarchy for process analysis, control, and optimization. The achieved results are illustrated in Fig. 2.25. The complete training set itself was correctly classified with regard to the bifurcation normal (class 1) or novel (class 2). For the test set, the vectors of classes 2 and 3 were also correctly identified as novel. However, numerous vectors of the normal test data were also classified as novel, as they occur a significant distance from the normal training data. Thus, a recognition rate of only 87.2% was achieved for the test set. It must be minded that the superior result of the RNN classifier required training by 95 vectors. The majority of these samples were counterexamples from the abnormal or novel range. In contrast, OCC was trained with only 30 vectors. The presented training data are rather sparse, so improvements of the OCC performance can be expected by providing larger data sets of normal process data as well as by a more sophisticated $R_{\max}$ computation and resulting normal range coverage in parameter space. However, though numerous practical improvements are possible, the feasibility of the described method to filter out significant novel data and perform as a data-reduction module also has been demonstrated.

The objectives of this feasibility study for the chosen problem and data have all been achieved. The feasibility of the selected soft-computing methods could be confirmed and relevant approaches for method improvement could be identified.

2.5 Proposed System Architecture

In the presented feasibility study, several selected methods were investigated with actual problem data with regard to their applicability for semiconductor manufacturing. As encouraging results have been obtained, a more so-

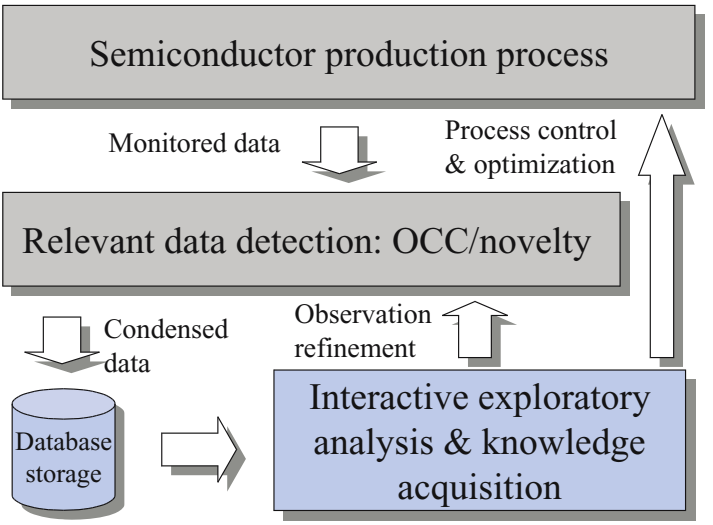


Fig. 2.26. Proposed system architecture for semiconductor manufacturing process analysis.

phisticated approach of employing and combining the regarded methods will be pursued next. A rough sketch of the envisioned information-processing architecture is given in Fig. 2.26. Similar to other applications, e.g., event classification in high-energy physics [2.39], a real-time classification stage is included in the proposed architecture. This module shall assess locally and in realtime whether interesting and relevant, i.e., novel, data occurred that should be stored for ensuing interactive analysis by human experts. OCC and the NOVCLASS model are first-choice candidates for this module. After storing in the database, dimensionality-reduction methods and interactive visualization will be undertaken for the analysis of the novel or abnormal data. Resulting understanding and knowledge extraction provide the baseline for potential actions as, e.g., classifier stage refinement or process control and optimization activities. Especially the interactive data visualization module can be significantly improved to the benefit of the regarded application. This has already been demonstrated for a different application domain in psychoacoustics, where an enhanced tool, denoted Acoustic Navigator (AN), was devised [2.25]. AN has been equipped with improved display features, such as multiple- and single-radar plots and practical search functions, which effortlessly direct the analyst to data entries of interest in the map visualization. These and numerous other convenience functions will allow transparent, fast, consistent, and thus, productive work on large, high-dimensional, and abstract databases. Figure 2.27 shows a first adaptation of the AN to the regarded application. Radar plots and the search function are illustrated. The focus of the follow-up research shall be put on this crucial system compo-

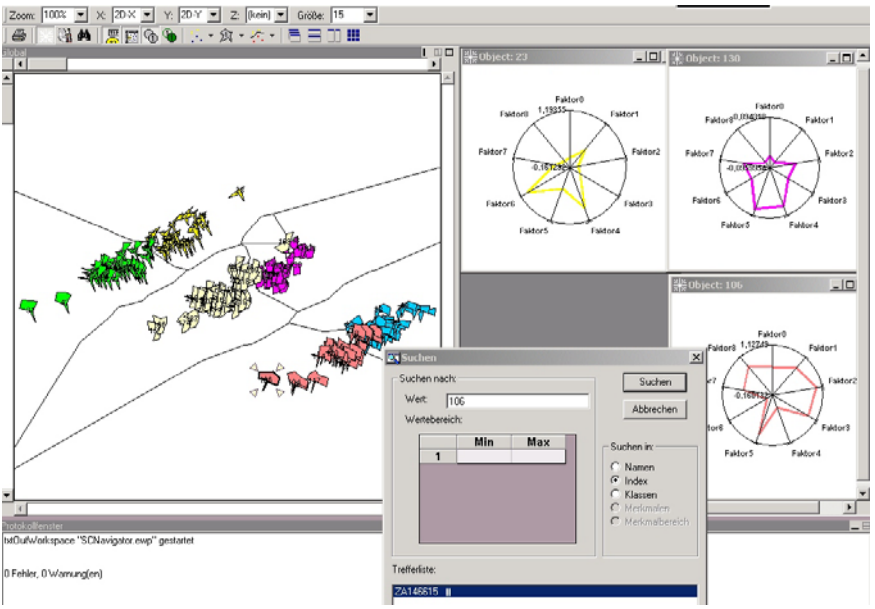


Fig. 2.27. Illustration of enhanced visualization features by the adapted AN.

nent and the related dimensionality-reduction methods, which also can be of assistance to cluster, select, and rank features or measurement parameters.

2.6 Conclusions

The presented work contributes to the industrial application of advanced soft-computing methods in the field of semiconductor manufacturing process data analysis. In particular, fast and efficient methods for multivariate data-dimensionality reduction, including automatic methods for parameter or parameter group saliency detection, and interactive visualization have been investigated in this first feasibility study.

Already the least complex and therefore most computationally inexpensive visualization methods allow significant insight into the structure of the data. Complemented by an interactive feature-selection tool, these visualization methods represent a powerful addition to the standard statistical analysis that is usually performed. Online visualization of the process trajectory in the multivariate space is also feasible by available fast methods for adding new data vectors in an existing mapping [2.23].

Furthermore, the investigation of automatic feature-selection methods has yielded very promising results. For instance, from the resulting projection, the asymmetry of the split can clearly be observed, which is a very significant achievement. Additionally, even in those cases where variables selected

were not pertinent to the split, the selection is soundly based. The bases are differences between the two lots, between single wafers in each lot, and even variations with regards to the position on the wafer. Again this is properly accounted for in the projections, further validating our approach.

In addition to these offline analysis and knowledge-extraction methods, dedicated classification techniques for online observation and potential control of the underlying process have been investigated. The feasibility of OCC and the proposed NOVCLASS method for selective data storage could be confirmed.

In this early stage of the work, the proposed methods were confronted with actual high-dimensional process data from a practical but, in terms of available samples N, small-scale problem. Most of the presented methods are more sensitive to the increase in the number of dimensions M than in the sample count N. Thus, it can be rightfully assumed that the methods will scale well with larger databases.

Future work will emphasize the improvement of the visualization tool and the integration of the algorithms and tools into the existing industrial environment for meaningful large-scale method application, assessment, and improvement based on more comprehensive data and data containing heretofore unknown information on the process.

Acknowledgments

The contributions of Michael Eberhardt and Robert Wenzel to the QuickCog System and Acoustic Navigator are gratefully acknowledged. Michael Eberhardt made part of this work feasible by contributing a data-converting tool and adapting Acoustic Navigator to the task presented. Thanks go to Bernd Vollmer and Christian Esser for friendly support and encouragement and to Klaus Franke for providing the photographs in Section 2.2.

References

2.1 Aarts, E., and Korst, J., *Simulated Annealing and Boltzmann Machines*, Addison Wesley, 1988.

2.2 Semiconductor Industry Association, *International Technology Roadmap for Semiconductors*, Semiconductor Industry Association, San Jose, CA, http://notes.sematech.org/ntrs/Rdmpmem.nsf, 1999.

2.3 Braha, D., and Shmilovici, A., Data mining for improving a cleaning process in the semiconductor industry, *IEEE Transactions on Semiconductor Manufacturing*, 15(1):91–101, Feb. 2002.

2.4 Broomhead, D. S., and Lowe, D., Multivariable functional interpolation and adaptive networks, in *Complex Systems 2*, pp. 321–55, 1988.

2.5 Collins, E., Glosh, S., and Scofield, C., Neural network decision learning system applied to risk analysis: Mortgage underwriting and delinquency risk assessment, in *DARPA Neural Network Study Final Report – Appendix E, Technical Report 840*, pp. 65–79. Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, MA, March 1989.
2.6 Cooper, L. N., Elbaum, C., Reilly, D. L., and Scofield, C. L., Parallel, multiunit, adaptive, nonlinear pattern class separator and identifier, in *United States Patents, Patent Number: 4,760,604*, July 1988.
2.7 Quadrillion Corporation, Q-Yield, http://www.quadrillion.com, 2002.
2.8 Devijver, P. A., and Kittler, J., On the edited nearest neighbor rule, in *Proc. 5th International Conference on Pattern Recognition*, vol. 1, pp. 72–80, Dec. 1980.
2.9 Dzwinel, W., How to make Sammon's mapping useful for multidimensional data structure analysis, in *Pattern Recognition*, 27(7), Elsevier Science Ltd, pp. 949–59, 1994.
2.10 Eberhardt, M., Hecht, R., and König, A., Einsatz des Konzepts *Machine-in-the-Loop-Learning* zum individuellen, robusten Anlernen von Laborrobotersystemen, in *KI-Zeitschrift*, No. 2, pages 44–7, 2002.
2.11 Eberhardt, M., Kossebau, F. K. H., and König, A., Automatic feature selection by genetic algorithms, in *Proc. Int Conf. on Artificial Neural Networks and Genetic Algorithms, ICANNGA01*, pages 256–9, Prague, April 2001.
2.12 Fukunaga, K., *Introduction to Statistical Pattern Recognition.* Academic Press, Harcourt Brace Jovanovich, Publishers, Boston, San Diego, New York, London, Sydney, Tokyo, Toronto, 1990.
2.13 Gates, G. W., The reduced nearest neighbour rule, in *IEEE Transactions on Information Theory*, vol. IT-18, pp. 431–3, 1972.
2.14 Goser, K., Marks, K. M., Rückert, U., and Tryba, V., Selbstorganisierende Karten zur Prozessüberwachung und -voraussage, in *Proc. 3. Int. GI-Kongress über wissensbasierte Systeme, München (16.-17. Okt.)*, pp. 225–37. Informatik Fachberichte Nr. 227, Berlin: Springer Verlag, 1989.
2.15 Katayama, R., Watanabe, M., Kuwata, K., Kajitani, Y., and Nishida, Y., Performance of self-generating radial basis function for function approximation, in *Proc. International Joint Conference on Neural Networks IJCNN'93*, Nagoya, Japan, Vol.I, pp. 471–4, IEEE, 1993.
2.16 Kittler, J., *Feature Selection and Extraction*, Academic Press, Inc., Tzai. Y. Young, King Sun-Fu, Publishers, Orlando, San Diego, New York, Austin, London, Montreal, Sydney, Tokyo, Toronto, 1986.
2.17 Kober, R., Howard, C., and Bock, P., Anomaly detection in video images, in *Proc. 5th International Conference on Neural Networks and Their Applications NEURO-NIMES'92*, 1992.
2.18 Köhler, C., König, A., Temelkova-Kurktschiev, T., and Hanefeld, M., Application of interactive multivariate data visualisation to the analysis of patients findings in metabolic research, in *Proc. 3rd Int. Conf. on Knowledge-Based Intelligent Information Engineering Systems KES'99*, pp. 397–402, Adelaide, Australia, Aug. 1999.
2.19 Kohonen, T., *Self-Organization and Associative Memory*, Springer-Verlag, Berlin, Heidelberg, London, Paris, Tokyo, Hong Kong, 1989.
2.20 König, A., Neuronale Strukturen zur sichtgestützten Oberflächeninspektion von Objekten in industrieller Umgebung, Darmstädter Dissertation D 17 (available from http://www.iee.et.tu-dresden.de/~koeniga), Sept. 1995.
2.21 König, A., A novel supervised dimensionality reduction technique by feature weighting for improved neural network classifier learning and generalization, in

Proc. 6th Int. Conf. on Soft Computing and Information/Intelligent Systems IIZUKA'2000, pp. 746–53, Iizuka, Fukuoka, Japan, Oct. 2000.

2.22 König, A., Dimensionality reduction techniques for multivariate data classification, interactive visualization, and Analysis – Systematic feature selection vs. extraction, in *Proc. 4th Int. Conf. on Knowledge-Based Intelligent Engineering Systems & Allied Technologies KES'2000*, pp. 44–56, University of Brighton, UK, Aug. 2000.

2.23 König, A., Interactive visualization and analysis of hierarchical neural projections for data mining. in *IEEE Trans. on Neural Networks, Special Issue for Data Mining and Knowledge Discovery*, pp. 615–24, May 2000.

2.24 König, A., Dimensionality reduction techniques for interactive visualisation, exploratory data analysis, and classification, in Pal, N. R. (ed.), *Pattern Recognition in Soft Computing Paradigm*, vol. 2, chap. 1, pp. 1–37, World Scientific, FLSI Soft Computing Series, Singapore, Jan. 2001.

2.25 König, A., Blutner, F. E., Eberhardt, M., and Wenzel, R., Design and application of an acoustic database navigator for the interactive analysis of psychoacoustic sound archives and sound engineering, in Hsu, C. (ed.), *Advanced Signal Processing Technology by Soft Computing*, vol. 1, chap. 3, pp. 36–65, World Scientific, FLSI Soft Computing Series, Singapore, Nov. 2000.

2.26 König, A., Bulmahn, O., and Glesner, M., Systematic methods for multivariate data visualization and numerical assessment of class separability and overlap in automated visual industrial quality control, in *Proc. 5th British Machine Vision Conf. BMVC'94*, pp. 195–204, Sept. 1994.

2.27 König, A., Eberhardt, M., and Wenzel, R., A transparent and flexible development environment for rapid design of cognitive systems, in *Proc. EUROMICRO'98 conference, Workshop Computational Intelligence*, Publisher IEEE CS, pp. 655–62, Västeraas, Sweden, Aug. 25–27 1998.

2.28 König, A., Eberhardt, M., and Wenzel, R., QuickCog self-learning recognition system – Exploiting machine learning techniques for transparent and fast industrial recognition system design, in *Image Processing Europe*, pp. 10–9, PennWell, Sept./Oct. 1999.

2.29 König, A., Eberhardt, M., and Wenzel, R., QuickCog – HomePage, in http://www.iee.et.tu-dresden.de/~ koeniga/QuickCog.html, 2000.

2.30 König, A., Raschhofer, R., and Glesner, M., A novel method for the design of radial-basis-function networks and its implication for knowledge extraction, in *IEEE International Conference on Neural Networks*, vol. III, Orlando, pp. 1804–9, Piscataway, NJ, June/July 1994.

2.31 König, A., Windirsch, P. and Glesner, M., Massively parallel VLSI-implementation of a dedicated neural network for anomaly detection in automated visual quality control, in *Proc. 4th Int. Conf. on Microelectronics for Neural Networks and Fuzzy Systems*, pp. 354–63, Sept. 1994.

2.32 Koontz, W. L. G., and Fukunaga, K., A nonlinear feature extraction algorithm using distance transformation, in *IEEE Transactions on Computers C-21*, no. 1, pp. 56–63, 1972.

2.33 Kozma, R., Kitamura, M., Sakuma, M., and Yokoyama, Y., Anomaly detection by neural network models and statistical time series analysis, in *Proc. International Conference on Neural Networks ICNN'94, Orlando, Vol.V*, pp. 3207–10, IEEE, 1994.

2.34 Lee, R. C. T., Slaggle, J. R., and Blum, H., A triangulation method for the sequential mapping of points from N-space to two-space, in *IEEE Transactions on Computers C-26*, pp. 288–92, 1977.

2.35 Lemarie, B., Size reduction of a radial basis function network, in *Proc. International Joint Conference on Neural Networks IJCNN'93, Nagoya, Japan, Vol.I*, pp. 331–4, IEEE, 1993.
2.36 Ludwig, L., Epperlein, U., Kuge, H.-H., Federl, P., Koppenhoefer, B., and Rosenstiel, W., Classification of fingerprints of process control monitoring-data with self-organizing maps, in *Proc. of EANN97, Stockholm, June 16*, pp. 107–12, 1997.
2.37 Ludwig, L., Pelz, E., Kessler, M., Sinderhauf, W., Koppenhoefer, B., and Rosenstiel, W., Prediction of functional yield of chips in semiconductor industry applications, in *Proc. of EANN98, Gibraltar, June 12-14*, pp. 157–161, 1998.
2.38 Marks, K. M., and Goser, K., Analysis of VLSI process data based on self-organizing feature maps, in *Proc. of Neuro-Nimes, Nimes (15.-17. Nov.)*, pp. 337–48, 1988.
2.39 Masa, P., Hoen, K., and Wallinga, H., A high-speed analog neural processor, in *IEEE Micro*, pp. 40–50, IEEE Computer Society, June 1994.
2.40 Moore, G .E., Cramming more components onto integrated circuits, *Electronics Magazine*, 38:114–7, 1965.
2.41 Parzen, E., On estimation of a probability density function and mode, in *Ann. Math. Stat., No.33*, p. 1065, 1962.
2.42 Platt, J., A resource-allocating network for function interpolation, in *Neural Computation, Vol.3*, pp. 213–25, 1991.
2.43 Poggio, T., and Girosi, F., Networks for approximation and learning, in *Proc. IEEE, Vol.78, No.9*, pp. 1481–97, 1990.
2.44 Powell, M. J. D., *Radial Basis Functions for Multivariable Interpolation*, Clarendon Press, Oxford, 1987.
2.45 Raymer, M. L., Punch, W. F., Goodman, E. D., Kuhn, L. A., and Jain, A. K., Dimensionality reduction using genetic algorithms, *IEEE Transactions on Evolutionary Computation*, 4(2):164–71, July 2000.
2.46 Reilly, D. L., Cooper, L. N., and Elbaum, C., A neural model for category learning, in *Biological Cybernetics, 45*, pp. 35–41, 1982.
2.47 Rückert, U., Softwareumgebung DANI zur schnellen explorativen Analyse sehr grosser Datenbestände, http://wwwhni.uni-paderborn.de/sct/cognitronics/#, 2002.
2.48 Sammon, J. W., A nonlinear mapping for data structure analysis, in *IEEE Transactions on Computers C-18, No.5*, pp. 401–9, 1969.
2.49 Sammon, J. W., Interactive pattern analysis and classification, in *IEEE Transactions on Computers C-19, No.7*, pp. 594–616, 1970.
2.50 Smith, S. .D. G., and Escobedo, R. A., Engineering and manufacturing applications of ART-1 neural networks, in *Proc. International Conference on Neural Networks ICNN'94, Orlando, Vol. VI*, pp. 3780–5, IEEE, 1994.
2.51 IDS Software Systems, dataPOWERSsc – Software for Semiconductor Analysis and Data Management, http://www.idsusa.com/site/products/datapower.html, 2002.
2.52 Knights Technology, KnightsYield Management Products (Data Explorer, Yield Manager), http://www.eletroglas.com/products/knights_datasheets/, 2002.
2.53 Turney, P., Data engineering for the analysis of semiconductor manufacturing data, in *Proc. of IJCAI Workshop on Data Engineering for Inductive Learning*, pp. 1–10, 1995.
2.54 Ultsch, A., and Siemon, H. P., Exploratory data analysis: Using Kohonen networks on transputers, in *Interner Bericht Nr. 329 Universität Dortmund, Dezember 1989*, 1989.

3. Clustering and Visualization of Retail Market Baskets

Joydeep Ghosh and Alexander Strehl

The University of Texas at Austin, Austin TX 78705, USA;
email: ghosh@ece.utexas.edu, alexander@strehl.com

Transaction analysis, including clustering of market baskets, is a key application of data mining to the retail industry. This domain has some specific requirements, such as the need for obtaining easily interpretable and actionable results. It also exhibits some very challenging characteristics, mostly stemming from the fact that the data have thousands of features and are highly non-Gaussian and sparse. This chapter proposes a relationship-based approach to clustering such data that tries to sidestep the "curse-of-dimensionality" issue by working in a suitable similarity space instead of the original high-dimensional feature space. This intermediary similarity space can be suitably tailored to satisfy business criteria such as requiring customer clusters to represent comparable amounts of revenue. We apply efficient and scalable graph-partitioning-based clustering techniques in this space. The output from the clustering algorithm is used to reorder the data points so that the resulting permuted similarity matrix can be readily visualized in two dimensions, with clusters showing up as bands. The visualization is very helpful for assessing and improving clustering. For example, actionable recommendations for splitting or merging clusters can be easily derived, and it also guides the user toward a suitable number of clusters. Results are presented on a real retail industry data set of several thousand customers and products.

3.1 Introduction

Knowledge discovery in databases often requires clustering the data into a number of distinct segments or groups in an effective and efficient manner. Good clusters show high similarity within a group and low similarity between any two different groups. Grouping customers based on buying behavior provides useful marketing decision support knowledge, especially in e-business applications where electronically observed behavioral data are readily available. Customer clusters can be used to identify up-selling and cross-selling opportunities with existing customers [3.1]. One can also cluster products that tend to sell together. Clustering of transactional data has widespread applications in the retail industry. This chapter focuses on this important application domain for data mining, first highlighting its unique requirements and challenges and then proposing customized methods for clustering and visualization of large-scale transactional data.

Domain-Specific Requirements. There are certain special requirements in real-life processes involving customer segmentation that are not encountered in more general clustering scenarios. First, it is usually desired that the clusters be *balanced*, i.e., of comparable size according to some measure of importance (number of customers/products, revenue represented, etc.), so that comparable amounts of resources (number of sales teams or marketing dollars, shelf/floor space, etc.) can be allocated to each segment [3.2]. Note that the natural clusters in the data may be highly imbalanced, so this requirement is not coming from data characteristics but from the need to make clustering results more *actionable*, a key data mining criteria. Also, because balancing is a global attribute, it is difficult to achieve by locally iterative, greedy methods. Interestingly, even in situations where balancing is not a requirement, techniques such as frequency-sensitive competitive learning that try to encourage balancing often provide superior results. This is because they have a regularization effect that helps avoid underrepresented clusters and guards against poor initialization [3.3].

Second, because the final results will be interpreted by a nontechnical person, such as the store manager, it is very important that each cluster be easily characterized in layman's terms, and the overall results visualized in intuitive ways. This eliminates several clustering and postprocessing choices. Third, it is not necessary that each data point be assigned to a cluster. For example, a low-revenue customer with atypical behavior can be safely ignored. This effect is quite noticeable when studying Web site–based purchasing behavior, where more than 30% of the visitors are often removed from the final clustering. However, very high-revenue customers may not be ignorable even if they are not well clustered. Finally, seasonality effects need to be taken care of to get valid and useful results. For example, clustering annual data instead of separately looking at summer and winter patterns can mask important seasonal associations [3.4].

Domain-Specific Challenges. Clustering real-life transactional data also poses some unique challenges that severely test traditional techniques for clustering and cluster visualization. To see why, consider a large market basket database involving thousands of customers and product lines. Each record corresponds to a store visit by a customer, so each customer could have multiple entries over time. The transactional database can be conceptually viewed as a sparse representation of a product (feature) by customer (object) matrix. The (i, j)th entry is nonzero only if customer j bought product i in that transaction. In that case, the entry represents pertinent information such as quantity bought or extended price (quantity $\times$ price) paid.

Because most customers buy only a small subset of these products during any given visit, the corresponding feature vector (column) describing such a transaction is high-dimensional (large number of products) but sparse (most features are zero). Also, transactional data typically have significant outliers, such as a few big corporate customers that appear in an otherwise small retail

customer data. Filtering these outliers may not be easy or desirable because they could be very important (e.g., major revenue contributors). In addition, features are often neither nominal nor continuous, but they may have discrete positive ordinal attribute values, with a strongly non-Gaussian distribution.

One way to reduce the feature space is to consider only the most dominant products (attribute selection), but in practice this may still leave hundreds of products to be considered. And because product popularity tends to follow a Zipf distribution [3.5], the tail is "heavy," meaning that *revenue* contribution from the less-popular products is significant for certain customers. Moreover, in retail, the higher *profit margins* are often associated with less-popular products. One can do a "roll-up" to reduce the number of products, but with a corresponding loss in resolution or granularity. Feature extraction or transformation is typically not carried out, as derived features lose the semantics of the original ones as well as the sparsity property.

The alternative to attribute reduction is to try "simplification via modeling." One approach would be to consider only binary features (bought or not). This reduces each transaction to an unordered set of the purchased products. Thus one can use techniques such as the a priori algorithm to determine associations or rules. In fact, this is currently the most popular approach to market basket analysis (see chap. 8 [3.6]). Unfortunately, this results in loss of vital information: one cannot differentiate between buying one gallon of milk and 100 gallons of milk, nor one can weight importance between buying an apple versus buying a car, though clearly these are very different situations from a business perspective. In general, association-based rules derived from such sets will be inferior when revenue or profits are the primary performance indicators, because the simplified data representation loses information about quantity, price, or margins. The other broad class of modeling simplifications for market basket analysis is based on taking a macrolevel view of the data having characteristics capturable in a small number of parameters. In retail, a 5–dimensional model for customers composed from indicators for recency, frequency, monetary value, volume, and tenure (RFMVT) is popular. However, this useful model is at a much lower resolution than looking at individual products and fails to capture actual purchasing behavior in more complex ways such as taste/brand preferences or price sensitivity,

Due to these characteristics, it is not surprising that traditional metric vector space–based clustering techniques work poorly on real-life market basket data. For example, a typical result of hierarchical agglomerative clustering (both single-link and complete-link approaches) on market basket data are to obtain one huge cluster near the origin, because most customers buy very few items,[2] and a few scattered clusters otherwise. Applying k-means could forceably split this huge cluster into segments depending on the initialization, but not in a meaningful manner.

[2] This is the dilution effect described in [3.7].

In summary, the challenges mainly arise from two aspects:[3] (i) large numbers of data samples, n; and (ii) each sample having *a large number of attributes or features* (dimensions, d), which cannot be readily simplified without much information loss.

The first aspect can be dealt with by subsampling the data, exploiting summary statistics, aggregating or "rolling up" to consider data at a coarser resolution, or using approximating heuristics that reduce computation time at the cost of some loss in quality. See chap. 8 [3.8] for several examples of such approaches. This chapter provides a way of addressing the second aspect by describing an alternative way of clustering and visualization when, even after feature reduction, one is left with hundreds of dimensions per object (and further reduction will significantly degrade the results), and moreover, simplifying data modeling assumptions are also not valid. Because clustering basically involves grouping objects based on their interrelationships or similarities, one can alternatively work in *similarity space* instead of the original feature space. The key insight in this work is that if one can find a similarity measure (derived from the object features) that is appropriate for the problem domain, then a single number can capture the essential "closeness" of a given pair of objects, and any further analysis can be based only on these numbers. This can be of great benefit when the data are very high-dimensional and simplifications such as further reducing dimensionality through projections or assuming conditional independence of features are not appropriate. Indeed, several researchers have recently proposed similarity-based clustering techniques for data mining applications [3.7], [3.9], [3.10], and this is emerging as an active area of research.

The similarity space also lends itself to a simple technique to visualize the clustering results. A major contribution of this chapter is to demonstrate that this technique has increased power when the clustering method used contains ordering information (e.g., *top-down*). Popular clustering methods in feature space are either nonhierarchical (as in k-means) or bottom-up (agglomerative clustering). However, if one transforms the clustering problem into a related problem of partitioning a similarity graph, several powerful partitioning methods with ordering properties can be applied. Moreover, the overall framework is quite generally applicable if one can determine the appropriate similarity measure for a given situation.

We begin by considering domain-specific transformations into similarity space in Section 3.2. Section 3.3 describes a specific clustering technique for transaction data (OPOSSUM), based on a multilevel graph-partitioning algorithm [3.11]. In Section 3.4, we describe a simple but effective visualization technique applicable to similarity spaces (CLUSION). Clustering and visualization results are presented in Section 3.5. In Section 3.6, we consider system

[3] A third issue of how to deal with seasonality and other temporal variations in the data is also critical in some applications. This aspect is not within the scope of this chapter, but see [3.4] for a solution for retail data.

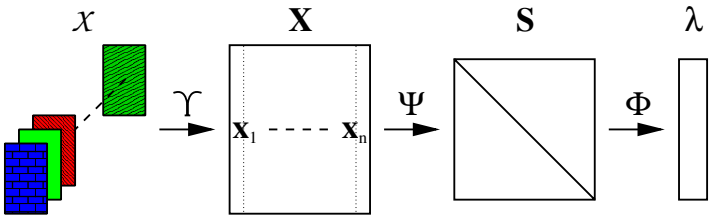


Fig. 3.1. The relationship-based clustering framework.

issues and briefly discuss several strategies to scale OPOSSUM for large data sets. Section 3.7 summarizes related work in clustering, graph partitioning and visualization.

3.2 Domain-Specific Features and Similarity Space

Notation. Let n be the number of objects/samples/points (e.g., customers, documents, Web sessions) in the data and d the number of features (e.g., products, words, Web pages) for each sample $\mathbf{x}_j$ with $j \in \{1, \ldots, n\}$. Let k be the desired number of clusters. The input data can be represented by a $d \times n$ data matrix $\mathbf{X}$ with the jth column vector representing the sample $\mathbf{x}_j$. $\mathbf{x}_j^\dagger$ denotes the transpose of $\mathbf{x}_j$. Hard clustering assigns a label $\lambda_j \in \{1, \ldots, k\}$ to each d-dimensional sample $\mathbf{x}_j$ such that similar samples get the same label. In general the labels are treated as nominals with no inherent order, though in some cases, such as 1-dimensional SOMs, any top-down recursive bisection approach our proposed method, the labeling contains extra ordering information. Let $\mathcal{C}_\ell$ denote the set of all objects in the ℓth cluster ($\ell \in \{1, \ldots, k\}$), with $\mathbf{x}_j \in \mathcal{C}_\ell \Leftrightarrow \lambda_j = \ell$ and $n_\ell = |\mathcal{C}_\ell|$.

Figure 3.1 gives an overview of our relationship-based clustering process from a set of raw object descriptions $\mathcal{X}$ (residing in input space $\mathcal{I}$) via the vector space description $\mathbf{X}$ (in feature space $\mathcal{F}$) and relationship description $\mathbf{S}$ (in similarity space $\mathcal{S}$) to the cluster labels λ (in output space $\mathcal{O}$): $(\mathcal{X} \in \mathcal{I}^n) \xrightarrow{\Upsilon} (\mathbf{X} \in \mathcal{F}^n \subset \mathbb{R}^{d \times n}) \xrightarrow{\Psi} (\mathbf{S} \in \mathcal{S}^{n \times n} = [0,1]^{n \times n} \subset \mathbb{R}^{n \times n}) \xrightarrow{\Phi} (\lambda \in \mathcal{O}^n = \{1, \ldots, k\}^n)$. For example, in Web page clustering, $\mathcal{X}$ is a collection of n Web pages x_j with $j \in \{1, \ldots, n\}$. Extracting features using Υ yields $\mathbf{X}$, the term frequencies of stemmed words, normalized such that for all documents $\mathbf{x} : \|\mathbf{x}\|_2 = 1$. Similarities are computed, using, e.g., cosine-based similarity Ψ, yielding the $n \times n$ similarity matrix $\mathbf{S}$. Finally, the cluster label vector λ is computed using a clustering function Φ, such as graph partitioning. In short, the basic process can be denoted as $\mathcal{X} \xrightarrow{\Upsilon} \mathbf{X} \xrightarrow{\Psi} \mathbf{S} \xrightarrow{\Phi} \lambda$.

Similarity Measures. In this chapter, we work in similarity space rather than the original vector space in which the feature vectors reside. This approach is particularly useful when there are many fewer objects than the

dimensionality of the feature space. In such cases, the objects can be well represented in a much lower-dimensional similarity space for further analysis, and several recent works on classification have fruitfully exploited this idea [3.10].

A similarity measure captures the relationship between two d–dimensional objects in a single number (using on the order of nonzeros or d, at worst, computations). Once this is done, the original high-dimensional space is not dealt with at all; we only work in the transformed similarity space, and subsequent processing is independent of d.

A similarity measure $\in [0, 1]$ captures how related two data points $\mathbf{x}_a$ and $\mathbf{x}_b$ are. It should be symmetric ($s(\mathbf{x}_a, \mathbf{x}_b) = s(\mathbf{x}_b, \mathbf{x}_a)$), with self-similarity $s(\mathbf{x}_a, \mathbf{x}_a) = 1$. However, in general, similarity functions (respectively their distance function equivalents $\delta = \sqrt{-\log(s)}$) do *not* obey the triangle inequality.

The suitability of a similarity measure depends on the nature (generative model) of the data. A precise notion of a similarity between two points in terms of the Fisher score is provided by information geometry arguments [3.12], but this of course needs an appropriate parametric generative model of the data. Let us now consider some popular similarity measures.

An obvious way to compute similarity is through a suitable monotonic and inverse function of a Minkowski (L_p) distance, δ. Candidates include $s = 1/(1 + \delta)$ and $s = e^{-\delta^2}$. Because maximizing average similarity (likelihood) is equivalent to minimizing the average squared distance under a Gaussian probability model, the latter formulation is preferable [3.13].

Similarity can also be defined by the cosine of the angle between two vectors:

$$s^{(\mathrm{C})}(\mathbf{x}_a, \mathbf{x}_b) = \frac{\mathbf{x}_a^\dagger \mathbf{x}_b}{\|\mathbf{x}_a\|_2 \cdot \|\mathbf{x}_b\|_2}. \tag{3.1}$$

Cosine similarity is widely used in text clustering because two documents with the same proportions of term occurrences but different lengths are often considered identical. Its normalization guards against domination by longer vectors.

In retail data, such normalization loses important information about the lifetime customer value, and we have recently shown that the *extended Jaccard similarity* measure is more appropriate [3.13]. For binary features, the Jaccard coefficient [3.14] measures the ratio of the intersection of the product sets to the union of the product sets corresponding to transactions $\mathbf{x}_a$ and $\mathbf{x}_b$, each having binary (0/1) elements:

$$s^{(\mathrm{J})}(\mathbf{x}_a, \mathbf{x}_b) = \frac{\mathbf{x}_a^\dagger \mathbf{x}_b}{\|\mathbf{x}_a\|_2^2 + \|\mathbf{x}_b\|_2^2 - \mathbf{x}_a^\dagger \mathbf{x}_b}. \tag{3.2}$$

Note that dimensions that have "0" entries for both vectors have no effect on the measure. This is very helpful because transactional data are highly sparse, so many zeros are encountered.

The extended Jaccard coefficient is also given by Eq. (3.2), but it allows elements of $\mathbf{x}_a$ and $\mathbf{x}_b$ to be arbitrary positive real numbers. This coefficient captures a vector-length-sensitive measure of similarity. However, it is still invariant to scale (dilating $\mathbf{x}_a$ and $\mathbf{x}_b$ by the same factor does not change $s(\mathbf{x}_a, \mathbf{x}_b)$). A detailed discussion of the properties of various similarity measures can be found in [3.13], where it is shown that the extended Jaccard coefficient is particularly well-suited for market basket data.

Because for general data distributions, one cannot avoid the "curse of dimensionality," there is no similarity metric that is optimal for all applications. Rather, one needs to determine an appropriate measure for the given application that captures the essential aspects of the class of high-dimensional data distributions being considered.

3.3 OPOSSUM

In this section, we present OPOSSUM (Optimal Partitioning of Sparse Similarities Using Metis), a similarity-based clustering technique particularly tailored to market basket data. OPOSSUM differs from other graph-based clustering techniques by application-driven balancing of clusters, nonmetric similarity measures, and visualization-driven heuristics for finding an appropriate k.

3.3.1 Balancing

Typically, one segments transactional data into five to twenty groups, each of which should be of comparable importance. Balancing avoids trivial clusterings (e.g., $k-1$ singletons and one big cluster). More importantly, the desired balancing properties have many application-driven advantages. For example, when each cluster contains the same number of customers, discovered phenomena (e.g., frequent products, co-purchases) have equal significance or support and are thus easier to evaluate. When each customer cluster equals the same revenue share, marketing can spend an equal amount of attention and budget for each of the groups. OPOSSUM strives to deliver "balanced" clusters using one of the following two criteria:

- *Sample Balanced:* Each cluster should contain roughly the same number of samples, n/k. This allows, for example, retail marketers to obtain a customer segmentation with comparably sized customer groups.
- *Value Balanced:* Each cluster should contain roughly the same number of feature values. Thus, a cluster represents a kth fraction of the total feature value $v = \sum_{j=1}^{n} \sum_{i=1}^{d} x_{i,j}$. In customer clustering, we use extended price per product as features and thus each cluster represents a roughly equal contribution to total revenue.

We formulate the desired balancing properties by assigning each object (customer, document, Web session) a weight and then softly constrain the sum of weights in each cluster. For sample-balanced clustering, we assign each sample $\mathbf{x}_j$ the same weight $w_j = 1/n$. To obtain value-balancing properties, a sample $\mathbf{x}_j$'s weight is set to $w_j = \frac{1}{v}\sum_{i=1}^{d} x_{i,j}$. Note that the sum of weights for all samples is 1.

3.3.2 Vertex-Weighted Graph Partitioning

We map the problem of clustering to partitioning a vertex-weighted graph $\mathcal{G}$ into k unconnected components of approximately equal size (as defined by the balancing constraint) by removing a minimal number of edges. The objects to be clustered are viewed as a set of vertices $\mathcal{V} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. Two vertices $\mathbf{x}_a$ and $\mathbf{x}_b$ are connected with an undirected edge $(a, b) \in \mathcal{E}$ of positive weight given by the similarity $s(\mathbf{x}_a, \mathbf{x}_b)$. This defines the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. An edge separator $\Delta\mathcal{E}$ is a set of edges whose removal splits the graph $\mathcal{G}$ into k pairwise unconnected components (subgraphs) $\{\mathcal{G}_1, \ldots, \mathcal{G}_k\}$. All subgraphs $\mathcal{G}_\ell = (\mathcal{V}_\ell, \mathcal{E}_\ell)$ have pairwise disjoint sets of vertices and edges. The edge separator for a particular partitioning includes all the edges that are not part of any subgraph or $\Delta\mathcal{E} = (\mathcal{E} \setminus (\mathcal{E}_1 \cup \mathcal{E}_2 \cup \ldots \cup \mathcal{E}_k))$. The clustering task is thus to find an edge separator with a minimum sum of edge weights, which partitions the graph into k disjoint pieces. The following equation formalizes this *minimum-cut objective*:

$$\min_{\Delta\mathcal{E}} \sum_{(a,b)\in\Delta\mathcal{E}} s(\mathbf{x}_a, \mathbf{x}_b). \tag{3.3}$$

Without loss of generality, we can assume that the vertex weights w_j are normalized to sum to one: $\sum_{j=1}^{n} w_j = 1$. While striving for the minimum-cut objective, *the balancing constraint*

$$k \max_{\ell\in\{1,\ldots,k\}} \sum_{\lambda_j=\ell} w_j \leq t \tag{3.4}$$

has to be fulfilled. The left-hand side of the inequality is called the imbalance (the ratio of the biggest cluster in terms of cumulative normalized edge weight to the desired equal cluster size, n/k) and has a lower bound of one. The balancing threshold t enforces perfectly balanced clusters for $t = 1$. In practice t is often chosen to be slightly greater than 1 (e.g., we use $t = 1.05$ for all our experiments, which allows at most 5% of imbalance).

Thus, in graph partitioning one has to essentially solve a constrained optimization problem. Finding such an optimal partitioning is an NP-hard problem [3.15]. However, there are fast, heuristic algorithms for this widely studied problem. We experimented with the Kernighan-Lin (KL) algorithm, recursive spectral bisection, and multilevel k-way partitioning (Metis).

The basic idea in KL [3.16] to dealing with graph partitioning is to construct an initial partition of the vertices either randomly or according to some problem-specific strategy. Then the algorithm sweeps through the vertices, deciding whether the size of the cut would increase or decrease if we moved this vertex **x** to another partition. The decision to move **x** can be made in time proportional to its degree by simply counting whether more of **x**'s neighbors are on the same partition as **x**. Of course, the desirable side for **x** will change if many of its neighbors switch, so multiple passes are likely to be needed before the process converges to a local optimum.

In recursive bisection, a k-way split is obtained by recursively partitioning the graph into two subgraphs. Spectral bisection [3.17], [3.18] uses the eigenvector associated with the second smallest eigenvalue of the graph's Laplacian (Fiedler vector) [3.19] for splitting.

Metis by Karypis et al. [3.11] handles multiconstraint multiobjective graph partitioning in three phases: (i) coarsening, (ii) initial partitioning, and (iii) refining. First a sequence of successively smaller and therefore coarser graphs is constructed through heavy-edge matching. Second, the initial partitioning is constructed using one out of four heuristic algorithms (three based on graph growing and one based on spectral bisection). In the third phase the coarsened partitioned graph undergoes boundary Kernighan-Lin refinement. In this last phase vertices are only swapped among neighboring partitions (boundaries). This ensures that neighboring clusters are more related than nonneighboring clusters. This ordering property is beneficial for visualization, as explained in Section 3.6.1. In contrast, because recursive bisection computes a bisection of a subgraph at a time, its view is limited. Thus, it cannot fully optimize the partition ordering and global constraints. This renders it less effective for our purposes. Also, we found the multilevel partitioning to deliver the best partitioning and to be the fastest and most scalable of the three choices we investigated. Hence, Metis is used as the graph partitioner in OPOSSUM.

3.3.3 Determining the Number of Clusters

Finding the "right" number of clusters k for a data set is a difficult and often ill-posed problem, because even for the same data set, there can be several answers depending on the scale or granularity one is interested in. In market basket clustering, the number of clusters is often prespecified due to business constraints (such as marketing personalization, human understandability) in the range from 5 to 20. Alternatively, heuristics for finding the right k can be used [3.2].

3.4 CLUSION: Cluster Visualization

In this section, we present our visualization tool, highlight some of its properties, and compare it with some popular visualization methods. Applications of this tool are illustrated in Section 3.5.

3.4.1 Coarse Seriation

When data are limited to two or three dimensions, the most powerful tool for judging cluster quality is usually the human eye. CLUSION, our CLUSter visualizatION toolkit, allows us to convert high-dimensional data into a perceptually more suitable format and to employ the human vision system to explore the *relationships* in the data, *guide* the clustering process, and *verify* the quality of the results. In our experience with two years of Dell customer data, we found CLUSION effective for getting clusters balanced with respect to number of customers or net dollar amount, and even more so for conveying the results to marketing management.

CLUSION looks at the output of a clustering routine, reorders the data points such that points with the same cluster label are contiguous, and then visualizes the resulting permuted similarity matrix, $\mathbf{S}'$. More formally, the original $n \times n$ similarity matrix $\mathbf{S}$ is permuted with an $n \times n$ permutation matrix $\mathbf{P}$ defined as follows:

$$p_{i,j} = \begin{cases} 1 \text{ if } j = \sum_{a=1}^{i} l_{a,\lambda_i} + \sum_{\ell=1}^{\lambda_i - 1} n_\ell \\ 0 \text{ otherwise,} \end{cases} \tag{3.5}$$

where l are entries in the binary $n \times k$ cluster membership indicator matrix $\mathbf{L}$:

$$l_{i,j} = \begin{cases} 1 \text{ if } \lambda_i = j \\ 0 \text{ otherwise.} \end{cases} \tag{3.6}$$

In other words, $p_{i,j}$ is 1 if j is the number of points among the first i points that belong to the cluster λ_i plus the number of points in the clusters 1 to $\lambda_i - 1$. This is easily understood with an example. Let there be four points with the following labeling:

$$\lambda = \begin{pmatrix} 3 \\ 2 \\ 1 \\ 2 \end{pmatrix}. \tag{3.7}$$

Then the binary cluster membership matrix is

$$\mathbf{L} = \begin{pmatrix} 0\,0\,1 \\ 0\,1\,0 \\ 1\,0\,0 \\ 0\,1\,0 \end{pmatrix}. \tag{3.8}$$

The permutation matrix then turns out to be

$$\mathbf{P} = \begin{pmatrix} 0\,0\,1\,0 \\ 0\,1\,0\,0 \\ 0\,0\,0\,1 \\ 1\,0\,0\,0 \end{pmatrix}. \tag{3.9}$$

The permuted similarity matrix $\mathbf{S}'$ and the corresponding label vector λ' and data matrix $\mathbf{X}'$ are:

$$\mathbf{S}' = \mathbf{P}\mathbf{S}\mathbf{P}^{\dagger}\ ,\ \lambda' = \mathbf{P}\lambda\ ,\ \mathbf{X}' = \mathbf{P}\mathbf{X}\,. \tag{3.10}$$

So in our example

$$\lambda' = \begin{pmatrix} 1 \\ 2 \\ 2 \\ 3 \end{pmatrix}$$

is now serialized (through row swapping) and the corresponding similarity matrix $\mathbf{S}$ contains the entries shown in CLUSION (through row and column swapping).

For a "good" clustering algorithm and $k \to n$, this is related to sparse matrix reordering, for this results in the generation of a "banded matrix" where high entries should all fall near the diagonal line from the upper left to the lower right of the matrix. Because Eq. 3.10 is essentially a partial ordering operation we also refer to it as coarse *seriation*, a phrase used in disciplines such as anthropology and archaeology to describe the reordering of the primary data matrix so that similar structures (e.g., genetic sequences) are brought closer [3.20], [3.21].

3.4.2 Visualization

The seriation of the similarity matrix, $\mathbf{S}'$, is very useful for visualization. Because the similarity matrix is two dimensional, it can be readily visualized as a gray-level image where a white (black) pixel corresponds to minimal (maximal) similarity of 0 (1). The darkness (gray-level value) of the pixel at row a and column b increases with the similarity between the samples $\mathbf{x}_a$ and $\mathbf{x}_b$. When looking at the image it is useful to consider the similarity s as a random variable taking values from 0 to 1. The similarity *within* cluster ℓ is thus represented by the average intensity within a square region with side length n_ℓ around the main diagonal of the matrix. The off-diagonal rectangular areas visualize the relationships *between* clusters. The brightness distribution in the rectangular areas yields insight toward the quality of the clustering and possible improvements. To make these regions apparent, thin

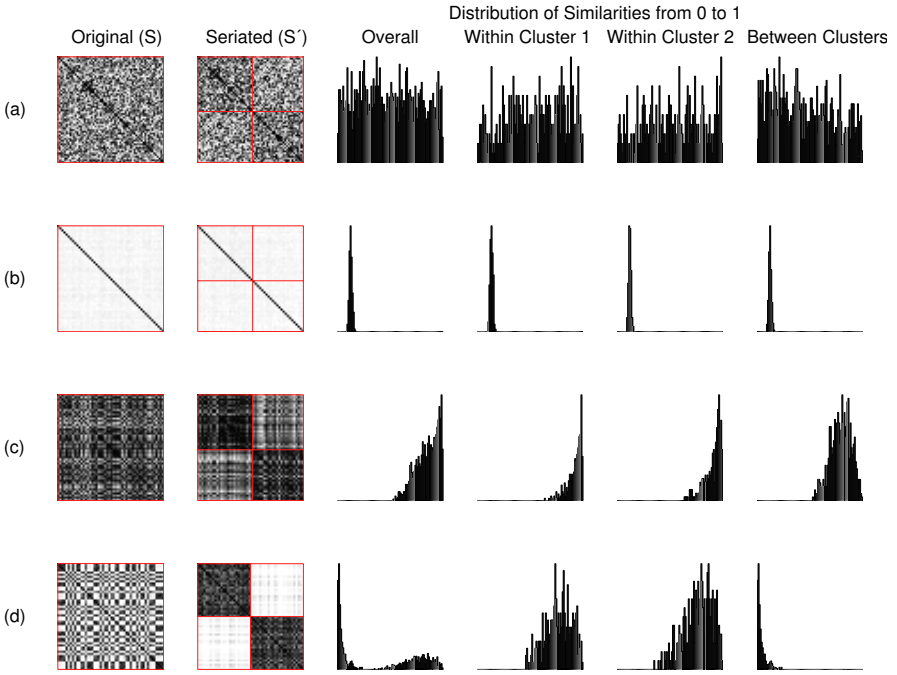


Fig. 3.2. Illustrative CLUSION patterns in original order and seriated using optimal bipartitioning are shown in the left two columns. The right four columns show corresponding similarity distributions. In each example there are 50 objects: (a) no natural clusters (randomly related objects), (b) set of singletons (pairwise near orthogonal objects), (c) one natural cluster (unimodal Gaussian), and (d) two natural clusters (mixture of two Gaussians).

red horizontal and vertical lines are used to show the divisions into the rectangular regions.[4] Visualizing similarity space in this way can help to quickly get a feel for the clusters in the data. Even for a large number of points, a sense for the intrinsic number of clusters k in a data set can be gained.

Figure 3.2 shows CLUSION output in four extreme scenarios to provide a feel for how data properties translate to the visual display. Without loss of generality, we consider the partitioning of a set of objects into two clusters. For each scenario, on the left-hand side the original similarity matrix $\mathbf{S}$ and the seriated version $\mathbf{S}'$ (CLUSION) for an optimal bipartitioning are shown. On the-right hand side four histograms for the distribution of similarity values s, which range from 0 to 1, are shown. From left to right, we have plotted: distribution of s over the entire data, within the first cluster, within the second cluster, and between the first and second clusters. If the data are naturally clustered and the clustering algorithm is good, then the middle two columns of plots will be much more skewed to the right compared to the

[4] This can be more clearly seen in the color pictures in the soft- copy.

first and fourth columns. In our visualization this corresponds to brighter off-diagonal regions and darker block-diagonal regions in $\mathbf{S}'$ compared to the original $\mathbf{S}$ matrix. The proposed visualization technique is quite powerful and versatile. In Figure 3.2(a) the chosen similarity behaves randomly. Consequently, no strong visual difference between on- and off-diagonal regions can be perceived with CLUSION in $\mathbf{S}'$. It indicates clustering is ineffective, which is expected because there is no structure in the similarity matrix. Figure 3.2(b) is based on data consisting of pairwise almost equidistant singletons. Clustering into two groups still renders the on-diagonal regions very bright, suggesting more splits. In fact, this will remain unchanged until each data point is a cluster by itself, thus revealing the singleton character of the data. For monolithic data (Fig. 3.2(c)), many strong similarities are indicated by an almost uniformly dark similarity matrix $\mathbf{S}$. Splitting the data results in dark off-diagonal regions in $\mathbf{S}'$. A dark off-diagonal region suggests that the clusters in the corresponding rows and columns should be merged (or not be split in the first place). CLUSION indicates that these data are actually one large cluster. In Fig. 3.2(d), the gray-level distribution of $\mathbf{S}$ exposes bright and dark pixels, thereby recommending it should be split. In this case, $k = 2$ apparently is a very good choice (and the clustering algorithm worked well) because in $\mathbf{S}'$ on-diagonal regions are uniformly dark and off-diagonal regions are uniformly bright.

This induces an intuitive mining process that guides the user to the "right" number of clusters. Too small a k leaves the on-diagonal regions inhomogeneous. On the contrary, growing k beyond the natural number of clusters will introduce dark off-diagonal regions. Finally, CLUSION can be used to visually compare the appropriateness of different similarity measures. Let us assume, for example, that each row in Fig. 3.2 illustrates a particular way of defining similarity for the same data set. Then CLUSION makes visually apparent that the similarity measure in (d) lends itself much better to clustering than the measures illustrated in rows (a), (b), and (c).

An interactive tool that facilitates exploration of the merge/split process can be experienced at `http://lans.ece.utexas.edu/~strehl/`.

3.4.3 Comparison

CLUSION gives a *relationship-centered* view, as contrasted with common projective techniques, such as the selection of dominant features or optimal linear projections (PCA), which are *object-centered*. In CLUSION, the actual features are transparent, instead, all pairwise relationships, the relevant aspect for the purpose of clustering, are displayed.

Figure 3.3 compares CLUSION with other popular visualizations. In Fig. 3.3(a) parallel axis, PCA projection, CViz (projection through plane defined by centroids of clusters 1, 2, and 3), as well as CLUSION succeed in visualizing the IRIS data. Membership in cluster 1/2/3 is indicated by colors red/blue/green (parallel axis), colors red/blue/green and shapes $\circ/\times/+$ (PCA and CViz),

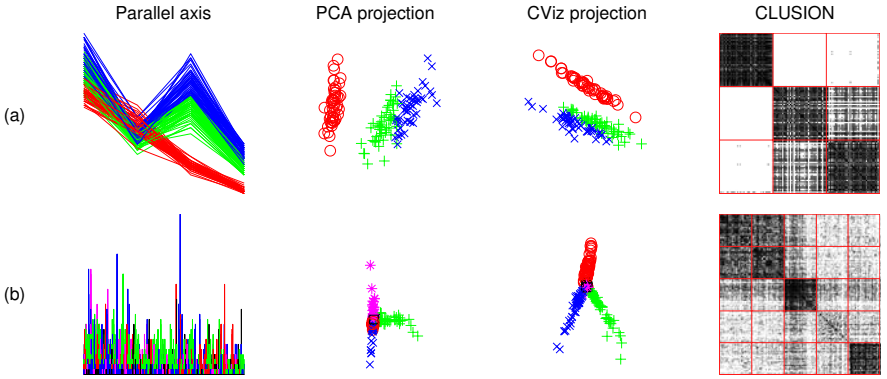


Fig. 3.3. Comparison of visualization techniques. All tools work well on the 4-dimensional IRIS data (a). But on the 2903-dimensional Yahoo! news document data (b), only CLUSION reveals that clusters 1 and 2 are actually highly related, cluster 3 is strong and interdisciplinary, cluster 4 is weak, and cluster 5 is strong.

and position on-diagonal from the upper-left to the lower-right corner (CLUSION), respectively. All four tools succeed in visualizing three clusters and making apparent that clusters 2 and 3 are closer than any other and cluster 1 is very compact. Figure 3.3(b) shows the same comparison for 293 documents from which 2903 word frequencies were extracted to be used as features. In fact this data set consists of five clusters selected from 40 clusters extracted from a Yahoo! news document collection that will be described in more detail in Section 3.5.2. The colors black/magenta and the shapes $\square/*$ have been added to indicate cluster 4/5, respectively. The parallel axis plot becomes useless clutter due to the high number of dimensions and the large number of objects. PCA and CViz succeed in separating three clusters each (2, 3, 5, and 1, 2, 3, respectively) and show all others superimposed on the axis origin. They give no suggestions toward which clusters are compact or which clusters are related. Only CLUSION suggests that clusters 1 and 2 are actually highly related, cluster 3 is interdisciplinary, cluster 4 is weak, and cluster 5 is a strong cluster. Indeed, when looking at the cluster descriptions (which might not be so easily available and understandable in all domains), the intuitive interpretations revealed by CLUSION are proven to be very true:

Cluster	Dominant category	Purity (%)	Entropy	Most frequent word stems
1	`health` (H)	100	0.00	hiv, depress, immun
2	`health` (H)	100	0.00	weight, infant, babi
3	`online` (o)	58	0.43	apple, intel, electron
4	`film` (f)	38	0.72	hbo, ali, alan
5	`television` (t)	83	0.26	household, sitcom, timeslot

Note that the majority category, purity, and entropy are available only where a supervised categorization is given. Of course the categorization cannot be used to tune the clustering. Clusters 1 and 2 contain only documents from the `Health` category so they are highly related. The fourth cluster, which is indicated to be weak by CLUSION, in fact has the lowest purity in the group, with 38% of documents from the most dominant category (`film`). CLUSION also suggests that cluster 3 is not only strong, as indicated by the dark diagonal region, but that it also has distinctly above average relationships to the other four clusters. On inspecting the word stems typifying this cluster (Apple, Intel, and electron(ics)), it is apparent that this is because of the interdisciplinary appearance of technology-savvy words in recent news releases. Because such cluster descriptions might not be so easily available or well understood in all domains, the intuitive display of CLUSION is very useful.

CLUSION has several other powerful properties. For example, it can be integrated with product hierarchies (metadata) to provide simultaneous customer and product clustering, as well as multilevel views and summaries. It also has a graphical user interface so one can interactively browse, split, or merge a data set, which is of great help to speed up the iterations of analysis during a data-mining project.

3.5 Experiments

3.5.1 Retail Market Basket Clusters

Let us illustrate clustering in a real retail transaction database of 21,672 customers of a drugstore.[5] For the illustrative purpose of this chapter, we randomly selected 2500 customers. The total number of transactions (cash-register scans) for these customers is 33,814 over three months. We rolled up the product hierarchy once to obtain 1236 different products purchased. Fifteen percept of the total revenue is contributed by the single item `Financial-Depts` (on-site financial services such as check cashing and bill payment), which was removed because it was too common. Of these, 473 products accounted for less than $25 each in total and were dropped. The remaining $n = 2466$ customers (34 customers had empty baskets after removing the irrelevant products) with their $d = 762$ features were clustered using OPOSSUM. The extended price was used as the feature entries to represent purchased quantity weighted according to price.

In this customer clustering case study we set $k = 20$. In this application domain, the number of clusters is often predetermined by marketing considerations such as advertising industry standards, marketing budgets, marketers' ability to handle multiple groups, and the cost of personalization. In general, a reasonable value of k can be obtained using heuristics (Section 3.3.3).

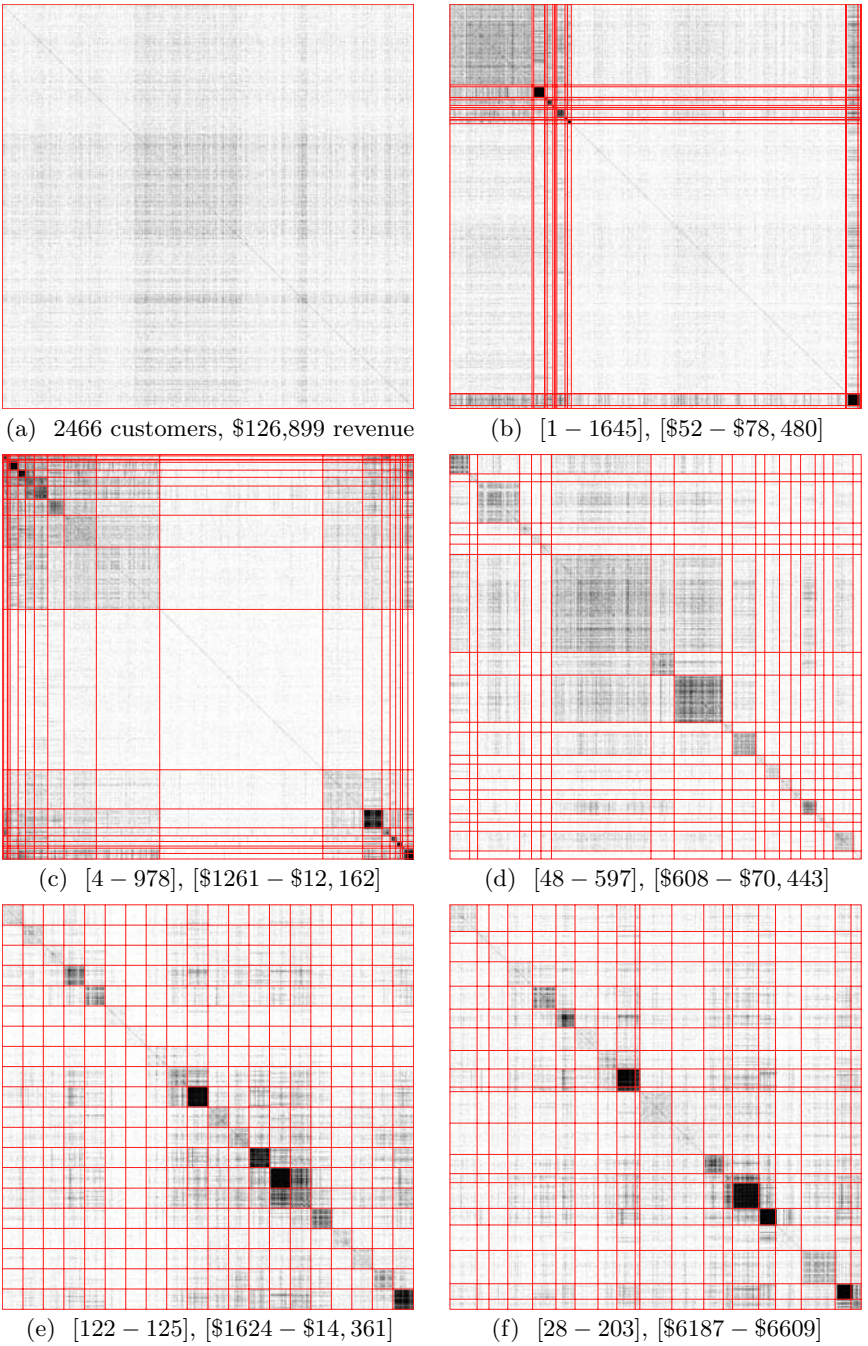
[5] provided by Knowledge Discovery 1.

OPOSSUM's results for this example were obtained with a 1.7 GHz Pentium 4 PC with 512 MB RAM in approximately 35 seconds (∼30s file I/O, 2.5s similarity computation, 0.5s conversion to integer weighted graph, 0.5s graph partitioning). Figure 3.4 shows the extended Jaccard similarity matrix (83% sparse) using CLUSION in six scenarious: (a) original (randomly) ordered matrix, (b) seriated using Euclidean k-means, (c) using SOM, (d) using standard Jaccard k-means, (e) using extended Jaccard sample balanced OPOSSUM, and (f) using value balanced OPOSSUM clustering. Customer and revenue ranges are given beneath each image. In (a), (b), (c), and (d) clusters are neither compact nor balanced. In (e) and (f) clusters are much more compact, even though there is the additional constraint that they be balanced, based on an equal number of customers and equal revenue metrics, respectively. Beneath each CLUSION visualization, the ranges of numbers of customers and revenue total in money among the 20 clusters are given to indicate balance. We also experimented with minimum distance agglomerative clustering but this resulted in 19 singletons and one cluster with 2447 customers, so we did not bother including this approach. Clearly, k-means in the original feature space, the standard clustering algorithm, does not perform well (Fig. 3.4(b)). The SOM after 100,000 epochs performs slightly better (Fig. 3.4(c)) but is outperformed by the standard Jaccard k-means (Fig. 3.4(d)), which is adopted to similarity space by using $\sqrt{-\log(s^{(J)})}$ as distances [3.13]. As the relationship-based CLUSION shows, OPOSSUM (Fig. 3.4(e),(f)) gives more compact (better separation of on- and off-diagonal regions) and well-balanced clusters compared to all other techniques. For example, looking at standard Jaccard k-means, the clusters contain between 48 and 597 customers contributing between $608 and $70,443 to revenue.[6] Thus the clusters may not be of comparable importance from a marketing standpoint. Moreover clusters are hardly compact: Darkness is only slightly stronger in the on-diagonal regions in Fig. 3.4(d). All visualizations have been histogram equalized for printing purposes. However, they are still much better observed by browsing interactively on a computer screen.

A very compact and useful way of profiling a cluster is to look at its most *descriptive* and most *discriminative* features. For market basket data, this can be done by looking at a cluster's highest-revenue products and the most unusual revenue drivers (e.g., products with the highest revenue lift). Revenue lift is the ratio of the average spending on a product in a particular cluster to the average spending in the entire data set.

In Table 3.1 the top three descriptive and discriminative products for the customers in the 20 value-balanced clusters are shown (see also Fig. 3.4(f)). Customers in cluster $\mathcal{C}_2$, for example, mostly spent their money on smoking-cessation gum ($10.15 on average). Interestingly, while this is a 35-fold average spending on smoking cessation gum, these customers also spend 35

[6] The solution for k-means depends on the initial choices for the means. A representative solution is shown here.



(a) 2466 customers, $126,899 revenue

(b) $[1-1645]$, $[\$52-\$78,480]$

(c) $[4-978]$, $[\$1261-\$12,162]$

(d) $[48-597]$, $[\$608-\$70,443]$

(e) $[122-125]$, $[\$1624-\$14,361]$

(f) $[28-203]$, $[\$6187-\$6609]$

Fig. 3.4. Visualizing partitioning drugstore customers into 20 clusters. Relationship visualizations using CLUSION: (a) original (randomly) ordered similarity matrix, (b) seriated or partially reordered using Euclidean k-means, (c) using SOM, (d) using standard Jaccard k-means, (e) using extended Jaccard sample balanced OPOSSUM, and (f) using value balanced OPOSSUM clustering. Customer and revenue ranges are given beneath each image. In (a), (b), (c), and (d) clusters are neither compact nor balanced. In (e) and (f) clusters are much more compact, even though there is the additional constraint that they be balanced, based on equal number of customers and equal revenue metrics, respectively.

$\mathcal{C}_\ell$	Top product	$	Lift	Sec. product	$	Lift	Third product	$	Lift
1	Bath gift packs	3.44	7.69	Hair growth m	0.90	9.73	Boutique island	0.81	2.61
2	Smoking cessati	10.15	34.73	tp canning item	2.04	18.74	Blood pressure	1.69	34.73
3	Vitamins other	3.56	12.57	tp coffee maker	1.46	10.90	Underpads hea	1.31	16.52
4	Games items 180	3.10	7.32	Facial moisturi	1.80	6.04	tp wine jug ite	1.25	8.01
5	Batt alkaline i	4.37	7.27	Appliances item	3.65	11.99	Appliances appl	2.00	9.12
6	Christmas light	8.11	12.22	Appliances hair	1.61	7.23	tp toaster/oven	0.67	4.03
7	Christmas food	3.42	7.35	Christmas cards	1.99	6.19	Cold bronchial	1.91	12.02
8	Girl toys/dolls	4.13	12.51	Boy toys items	3.42	8.20	Everyday girls	1.85	6.46
9	Christmas giftw	12.51	12.99	Christmas home	1.24	3.92	Christmas food	0.97	2.07
10	Christmas giftw	19.94	20.71	Christmas light	5.63	8.49	Pers cd player	4.28	70.46
11	tp laundry soap	1.20	5.17	Facial cleanser	1.11	4.15	Hand&body thera	0.76	5.55
12	Film cameras it	1.64	5.20	Planners/calend	0.94	5.02	Antacid h2 bloc	0.69	3.85
13	Tools/accessori	4.46	11.17	Binders items 2	3.59	10.16	Drawing supplie	1.96	7.71
14	American greeti	4.42	5.34	Paperback items	2.69	11.04	Fragrances op	2.66	12.27
15	American greeti	5.56	6.72	Christmas cards	0.45	2.12	Basket candy it	0.44	1.45
16	tp seasonal boo	10.78	15.49	American greeti	0.98	1.18	Valentine box c	0.71	4.08
17	Vitamins e item	1.76	6.79	Group stationer	1.01	11.55	tp seasonal boo	0.99	1.42
18	Halloween bag c	2.11	6.06	Basket candy it	1.23	4.07	Cold cold items	1.17	4.24
19	Hair clr perman	12.00	16.76	American greeti	1.11	1.34	Revlon cls face	0.83	3.07
20	Revlon cls face	7.05	26.06	Hair clr perman	4.14	5.77	Headache ibupro	2.37	12.65

$\mathcal{C}_\ell$	Top product	$	Lift	Sec. product	$	Lift	Third product	$	Lift
1	Action items 30	0.26	15.13	tp video comedy	0.19	15.13	Family items 30	0.14	11.41
2	Smoking cessati	10.15	34.73	Blood pressure	1.69	34.73	Snacks/pnts nut	0.44	34.73
3	Underpads hea	1.31	16.52	Miscellaneous k	0.53	15.59	tp irons items	0.47	14.28
4	Acrylics/gels/w	0.19	11.22	tp exercise ite	0.15	11.20	Dental applianc	0.81	9.50
5	Appliances item	3.65	11.99	Housewares peg	0.13	9.92	tp tarps items	0.22	9.58
6	Multiples packs	0.17	13.87	Christmas light	8.11	12.22	tv's items 6	0.44	8.32
7	Sleep aids item	0.31	14.61	Kava kava items	0.51	14.21	tp beer super p	0.14	12.44
8	Batt rechargeab	0.34	21.82	tp razors items	0.28	21.82	tp metal cookwa	0.39	12.77
9	tp furniture it	0.45	22.42	tp art&craft al	0.19	13.77	tp family plan,	0.15	13.76
10	Pers cd player	4.28	70.46	tp plumbing ite	1.71	56.24	Umbrellas adult	0.89	48.92
11	Cat litter scoo	0.10	8.70	Child acetamino	0.12	7.25	Pro treatment i	0.07	6.78
12	Heaters items 8	0.16	12.91	Laverdiere ca	0.14	10.49	Ginseng items 4	0.20	6.10
13	Mop/broom lint	0.17	13.73	Halloween cards	0.30	12.39	Tools/accessori	4.46	11.17
14	Dental repair k	0.80	38.17	tp lawn seed it	0.44	35.88	tp telephones/a	2.20	31.73
15	Gift boxes item	0.10	8.18	Hearing aid bat	0.08	7.25	American greeti	5.56	6.72
16	Economy diapers	0.21	17.50	tp seasonal boo	10.78	15.49	Girls socks ite	0.16	12.20
17	tp wine 1.5l va	0.17	15.91	Group stationer	1.01	11.55	Stereos items 2	0.13	10.61
18	tp med oint liq	0.10	8.22	tp dinnerware i	0.32	7.70	tp bath towels	0.12	7.28
19	Hair clr perman	12.00	16.76	Covergirl imple	0.14	11.83	tp power tools	0.25	10.89
20	Revlon cls face	7.05	26.06	Telephones cord	0.56	25.92	Ardell lashes i	0.59	21.87

Table 3.1. List of *descriptive* (top) and *discriminative* products (bottom) dominant in each of the 20 value balanced clusters obtained from the drugstore data (see also Figure 3.4(f)). For each item the average amount of money spent in this cluster and the corresponding lift are given.

times more on blood pressure–related items, peanuts, and snacks. Do these customers lead an unhealthy lifestyle and are they eager to change? Cluster $\mathcal{C}_{15}$, which can be seen to be a highly compact cluster of Christmas shoppers characterized by greeting-card and candy purchases. Note that OPOSSUM had an extra constraint that clusters be of comparable value. This may force a larger natural cluster to split, as may be the case causing the similar clusters $\mathcal{C}_9$ and $\mathcal{C}_{10}$. Both are Christmas-gift shoppers (Table 3.1(top)), cluster $\mathcal{C}_9$ are the moderate spenders, and cluster $\mathcal{C}_{10}$ are the big spenders, as cluster $\mathcal{C}_{10}$ is much smaller with equal revenue contribution (Fig. 3.4(f)). Our hunch is reinforced by looking at Fig. 3.4(f).

3.5.2 Other Applications

In this chapter, we focused on the application of OPOSSUM and CLUSION to market basket data. The framework is more general and can be applied to a variety of other domains, such as text document or Web session clustering. For each domain, the similarity metric s and the object weights have to be adopted. For text clustering, similarity of two pages can be defined as the cosine of their term frequency vectors. Page weight can be chosen to be proportional to the document length. In Web session clustering, the similarity can be based on the cosine of the visited pages vector and the weight can be based on the time spent on a particular Web page. For details and results, see [3.13] for document clustering and [3.22] for clustering Web sessions.

3.6 System Issues

3.6.1 Synergy Between OPOSSUM and CLUSION

The visualization and clustering techniques presented in this work need to be considered together, not in isolation. This is because CLUSION is particularly suited to viewing the output of OPOSSUM. First, the similarity matrix is already computed during the clustering step, so no extra computation is needed, except for permuting this matrix, which can be done in O(n) time because the size and seriation order of each partition are known. Second, because Metis involves boundary Kernighan-Lin refinement, clusters that are similar appear closer in the seriation order. Thus it is no coincidence that clusters $\mathcal{C}_{14}$ and $\mathcal{C}_{15}$ appear contiguous in Fig. 3.4(e). Finally, one can experiment with different similarity measures for OPOSSUM and quickly get visual feedback regarding their effectiveness using CLUSION.

3.6.2 Scalability

The computational bottleneck in the overall process lies in calculating the similarity matrix, which involves $O(n^2 d)$ operations, because similarity needs to be computed between each pair of data points and involves all the dimensions.[7] However, once this matrix is computed, any subsequent clustering routine does not depend on d at all! Metis is very fast, almost linear in the number of vertices for reasonably sparse graphs, as has been shown over numerous experiments [3.11]. Finally, the reordering of the similarity matrix for visualization is $O(n)$. Thus the overall method is linear in d.

The quadratic complexity with respect to the number of objects, n, is problematic for large data sets. Note that any clustering algorithm that compares an object with all others (e.g., agglomerative, all relationship-based

[7] By exploiting sparsity, computation of a single similarity value can be reduced from $O(d)$ to O(number of nonzeros in d).

methods) has a complexity at least $O(n^2)$, as does OPOSSUM. There are four main ways of reducing this computation. We mention them briefly and then explore the first option in a bit more detail.

1. Sampling: Sample the data, cluster the sample points, and then use a quick heuristic to allocate the nonsampled points to the initial clusters. This approach will yield a faster algorithm at the possible cost of some loss in quality and is employed, for example, in the Buckshot algorithm for the Scatter/Gather approach to iterative clustering for interactive browsing [3.23]. If the sample is $O(\sqrt{n})$, and the "nearest cluster center" is used to allocate the remaining points, one obtains an $O(kn)$ algorithm. Also related are randomized approaches that can partition a set of points into two clusters of comparable size in sublinear time, producing a $(1+\epsilon)$ solution with high probability [3.24]. We show later that because OPOSSUM is based on balanced clusters, sampling is a good choice because one can ensure with high probability that each cluster is represented in the sample without needing a large sample size [3.25].
2. Sequential Building: Construct a "core" clustering using a small number of elements, and then sequentially scan the data to allocate the remaining inputs, creating new clusters (and optionally adjusting existing centers) as needed. Such an approach is seen in CLARANS and BIRCH [3.26]. This style compromises balancing to some extent, and the threshold determining when a new cluster is formed has to be experimented with to bring the number of clusters obtained to the desired range. A version of this approach for graph partitioning using a corrupted clique model was proposed in [3.27] and applied to clustering gene expressions. This can be readily used for OPOSSUM as well. Sequential building is especially popular for out-of-core methods, the idea being to scan the database once to form a summarized model (for instance, the size, sum, and sum-squared values of each cluster [3.28]) in main memory. Subsequent refinement based on summarized information is then restricted to main memory operations without resorting to further disk scans.
3. Compare with representatives rather than with all points, as in k-means. The results, however, become sensitive to the initial selection of representatives.
4. Apply prior domain knowledge to "presegment" the data, e.g., using indices or other "partitionings" of the input space. As mentioned earlier, this becomes increasingly problematic as the dimensionality of the input space increases to the hundreds or beyond, where suitable segments may be difficult to estimate, predetermine, or populate.

All these approaches are somewhat orthogonal to the main clustering routine in that they can be applied in conjunction with most core clustering routines (including OPOSSUM) to save computation at the cost of some loss in quality.

3.6.3 FASTOPOSSUM

Because OPOSSUM aims to achieve balanced clusters, random sampling is effective for obtaining adequate examples of each cluster. If the clusters are perfectly balanced, the distribution of the number of samples from a specific cluster in a subsample of size $\underline{n}$ taken from the entire population is binomial with mean $\underline{n}/k$ and variance $\underline{n}(k-1)/k^2$. For a finite population that is balanced to begin with, the variance will be even less, because we are doing sampling without replacement. Thus if one cluster gets more than the expected number of points at some time in the sampling process, the fraction of the unsampled points that belong to this cluster becomes less than the expected number.

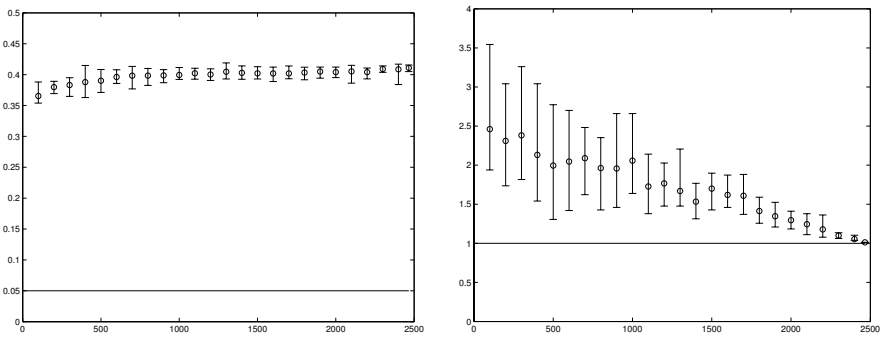
If we require at least r representatives from a cluster, then the number of samples is given by $\underline{n}/k \geq z_\alpha\sqrt{\underline{n}(k-1)}+r$, where $z_\alpha = 1.96$ or 2.81 for 97.5% and 99.5% confidence levels respectively. This is $O(rk)$. For example, if we have 10 clusters and need to ensure at least 20 representatives from a given cluster with probability 0.995, about 400 samples are adequate. Note that this number is independent of n if n is adequately large (at least 400 in this case), so even for more than one million customers, only 400 representatives are required. Additional analysis is provided in [3.25].

This suggests a simple and effective way to scale OPOSSUM to a very large number of objects n, using the following four-step process called FASTOPOSSUM:

1. Pick a boot-sample of size $\underline{n}$ so that the corresponding r value is adequate to define each cluster.
2. Apply OPOSSUM to the boot-sample to get k initial clusters.
3. Find the centroid for each of the k clusters.
4. Assign each of the remaining $n-\underline{n}$ points to the cluster with the nearest centroid.

Using $\underline{n} = \sqrt{n}$ reduces the complexity of FASTOPOSSUM to $O(kn)$. Note that the algorithm may not result in balanced clusters. We can enforce balancing by allocating the remaining points to the k clusters in groups, each time solving a stable-marriage problem [3.29], but this will increase the computation time.

Figure 3.5 illustrates the behavior of FASTOPOSSUM for the drugstore customer data set from Section 3.5.1. The remaining edge-weight fraction indicates how much of the cumulative edge weight remains after the edge separator has been removed: $[\sum_{\ell=1}^{k}\sum_{\lambda_a=\ell}\sum_{\lambda_b=\ell,b>a} s(\mathbf{x}_a,\mathbf{x}_b)]/[\sum_{a=1}^{n}\sum_{b=a+1}^{n} s(\mathbf{x}_a,\mathbf{x}_b)]$. The better the partitioning, the smaller the edge-separator and thus the larger the remaining edge weight fraction. Surprisingly the speedup does not result in a significantly decreased quality in terms of remaining edge weight (Fig. 3.5(a)). However, the balancing property is progressively relaxed as the boot sample becomes smaller in comparison to the full data set (Fig. 3.5(b)). Using $\underline{n} = 100$ initial points reduces the original computation time to less



(a) (b)

Fig. 3.5. Effect of subsampling on OPOSSUM. Cluster quality as measured by remaining edge weight fraction (a) and imbalance (b) of *total* graph with 2466 vertices (customers from Section 3.5.1) for various boot-sample sizes $\underline{n}$ in FASTOPOSSUM. For each setting of s the results' range and mean of 10 trials are depicted. Using all 2466 customers as the boot-sample (i.e., no subsampling) results in balancing within the 1.05 imbalance requirement and approximately 40% of edge weight remaining (compared to 5% baseline for random clustering). As the boot-sample becomes smaller the remaining edge weight stays approximately the same (a), but the imbalance increases (b).

than 1% at comparable remaining edge weight but at an imbalance of 3.5 in the worst of 10 random trials. These results indicate that scaling to large n is easily possible, if one is willing to relax the balance constraints.

3.6.4 Parallel Implementation

Another notion of scalability is with respect to the number of processors (speedup, iso-efficiency etc.). Our analysis [3.2] shows almost linear speedup for our method, as the similarity computation as well as graph partitioning can both be fairly trivially parallelized with little overhead. Parallel implementation of the all-pair similarity computation on SIMD or distributed memory processors is trivial. It can be done in a systolic or block systolic manner with essentially no overhead. Frameworks such as MPI also provide native primitives for such computations. Parallelization of Metis is also very efficient, and [3.30] reports partitioning of graphs with more than 7 million vertices in 7 seconds into 128 clusters on a 128 processor Cray T3E. For further details, see [3.2].

3.7 Related Work

Clustering has been widely studied in several disciplines, especially since the late 1960s [3.14], [3.31]. Classic approaches include partitional methods

such as k-means and k-medioids, bottom-up hierarchical approaches such as single-link or complete-link agglomerative clustering, soft partitioning approaches such as fuzzy clustering, EM-based techniques, and methods motivated by statistical mechanics [3.32]. Although several methods of clustering data defined by pairwise (dis)similarities are available [3.33], most classical techniques, as well as recent techniques proposed in the data mining community (CLARANS, DBScan, BIRCH, CLIQUE, CURE, WaveCluster etc. [3.34]), are based on distances between the samples in the original feature space. The emphasis of the data mining–oriented proposals is primarily on an efficient and scalable (with respect to number of records) implementation of approximate k-means, k-medioids, or local density estimation. Thus they are all faced with the "curse of dimensionality" [3.35] and the associated sparsity issues, when dealing with very high-dimensional data. Essentially the amount of data to sustain a given spatial density increases exponentially with the dimensionality of the input space, or alternatively, the sparsity increases exponentially given a constant amount of data, with points tending to become equidistant from one another. In general, this will adversely impact any method based on spatial density, unless the data follow certain simple distributions as described in the introduction. Certain other limitations of popular clustering methods are nicely illustrated in [3.9]. In [3.36], the authors recognize that one way of tackling high-dimensional data is to change the distance function in an application-specific way. They suggest some possible modified functions and principles but do not provide any experimental results.

In databases, where clustering is often tied to the need for efficient **indexing**, a variety of space-partioning methods (e.g., R-trees and variants) and data partitioning (such as KDB-trees), exist. These methods are typically tractable for up to 10- to 15-dimensional data, and by a judicious hybrid of these two approaches, data with tens of attributes may be partitioned [3.37]. Significant overlaps among the hyperrectangles the occurrences of several empty areas become increasingly problematic if the dimensionality is further increased (see [3.37] for more details).

Graph-theoretic clustering has been known for a while [3.14] though not commonly applied. But lately, such an approach has proved attractive for gene expression analysis [3.27]. Graphical methods also have emerged in the data mining literature to tackle high-dimensional data analysis. ROCK (Robust Clustering using linKs) [3.7] is an agglomerative hierarchical clustering technique for categorical attributes. It uses the binary Jaccard coefficient and a thresholding criterion to establish links between samples. Common neighbors are used to define interconnectivity of clusters, which is used to merge clusters. CHAMELEON [3.9] starts with partitioning the data into a large number of clusters by partitioning the v-nearest neighbor graph. In the subsequent stage clusters are merged based on relative interconnectivity and relative closeness measures. These localized measures lead to a dynamic

adaption capability with spectacular results for two-dimensional data. But its effectiveness and interpretability for higher-dimensional data are not reported. In [3.38], a hypergraph clustering approach was taken for clustering highly related items defined in high-dimensional space and generate the corresponding association rules. This method was applied to binarized data, with each frequent item set being represented by a weighted hyperedge. Like our method, it is suitable for high-dimensional data and is linear in d. Subsequently, this and another graph partitioning algorithm called principal direction divisive partitioning were applied for Web document categorization [3.39]. These two algorithms are the closest in spirit to our approach. Finally, spectral partitioning methods [3.17], [3.40] can be applied to similarity graphs. A probabilistic foundation for spectral methods for clustering and segmentation has been recently proposed [3.41].

Related work on scalability issues of clustering are discussed in Section 3.6.2. **Visualization** of high-dimensional data clusters can be largely divided into three popular approaches:

1. Dimensionality reduction by selection of two or three dimensions, or, more generally, projecting the data down to two or three dimensions. Often these dimensions correspond to principal components or a scalable approximation thereof. Another noteworthy method is CViz [3.42], which projects onto the plane that passes through three selected cluster centroids to yield a "discrimination optimal" two-dimensional projection. These projections are useful for a medium number of dimensions, i.e., if d is not too large (< 100).[8] Nonlinear projections have also been studied [3.43]. Re-creating a two- or three-dimensional space from a similarity graph can also be done through multidimensional scaling [3.44].
2. Parallel-axis plots show each object as a line along d parallel axes. However, this technique is rendered ineffective if the number of dimensions d or the number of objects gets too high.
3. Kohonen's self organizing map (SOM) [3.45] provides an innovative and powerful way of clustering while enforcing constraints on a logical topology imposed on the cluster centers. If this topology is two-dimensional, one can readily "visualize" the clustering of data. Essentially a two-dimensional manifold is mapped onto the (typically higher-dimensional) feature space, trying to approximate data density while maintaining topological constraints. Because the mapping is not bijective, the quality can degrade very rapidly with increasing dimensionality of the feature space, unless the data are largely confined to a much lower-order manifold within this space [3.43]. Multidimensional scaling (MDS) and associated methods also face similar issues.

[8] For text mining, linearly projecting down to about 20 to 50 dimensions does not affect results much (e.g., latent semantic indexing). However, it is still too high to visualize. A projection to lower dimensions leads to substantial degradation and 3-dimensional projections are of very limited utility.

Our visualization technique involves a smart reordering of the similarity matrix. **Ordering** of data points for visualization has previously been used in conjunction with clustering in different contexts. In cluster analysis of genome data [3.21] reordering the primary data matrix and representing it graphically have been explored. This visualization takes place in the primary data space rather than in the relationship space. Sparse primary data-matrix reorderings have also been considered for browsing hypertext [3.46].

A useful survey of visualization methods for data mining in general can be found in [3.47]. The popular books by E. Tufte on visualizing information are also recommended.

3.8 Concluding Remarks

A poll in June 2001 by KDNuggets (`http://www.kdnuggets.com/`) indicated that clustering was by far the most popular type of analysis in the last 12 months at 22% (followed by direct marketing at 14% and cross-sell models at 12%). The clustering process is characterized by extensive explorative periods where better domain understanding is gained. Often, in this iterative process the crucially important definitions of features or similarity are refined. The visualization toolkit CLUSION allows even nonspecialists to get an intuitive visual impression of the grouping nature of objects that may be originally defined in a high-dimensional space. Taking CLUSION from a postprocessing step into the loop can significantly accelerate the process of discovering domain knowledge, as it provides a powerful visual aid for assessing and improving clustering. For example, actionable recommendations for splitting or merging point-and-click user interface, and different similarity metrics can be compared visually. It also guides the user toward the "right number" of clusters. A demo and selected code of this tool can be found at `http://www.strehl.com/`.

The clustering algorithm presented is largely geared toward the needs of segmenting transactional data, with provision of getting balanced clusters and for selecting the quantity (revenue, margins) of interest to influence the grouping. Thus, rather than evaluating business objectives (such as revenue contribution) *after* clustering is done, they are directly *integrated* into the clustering algorithm. Moreover, it is a natural fit with the visualization algorithm. We also examined several ways of scaling the clustering routine to a large number of data points and elaborated on one approach that is able to use sampling effectively because of the balanced nature of the desired clusters.

Acknowledgments

We want to express our gratitude to Mark Davis of Knowledge Discovery 1 (since then acquired by Net Perceptions) for providing the drugstore re-

tail data set. This research was supported in part by Intel, Accenture, and IBM/Tivoli.

References

3.1 Lawrenc, R. D., et al, Personalization of supermarket product recommendations, *Data Mining and Knowledge Discovery*, 4(1/2):11–32, 2001.
3.2 Strehl, A., and Ghosh, J., Relationship-based clustering and visualization for high-dimensional data mining, *INFORMS Journal on Computing*, 15(2):208–30, 2003.
3.3 Banerjee, A., and Ghosh, J., Frequency sensitive competitive learning for clustering on high-dimensional hyperspheres, in *Proc. IJCNN'02, Honolulu*, pp. 1590–5, May 2002.
3.4 Gupta, G. K., and Ghosh, J., Detecting seasonal trends and cluster motion visualization for very high dimensional transactional data, in *Proc. 1st SIAM Conf. on Data Mining (SDM2001)*, pp. 115–29, 2001.
3.5 Zipf, G. K., Relative frequency as a determinant of phonetic change, Reprinted from the *Harvard Studies in Classical Philiology*, XL, 1929.
3.6 Berry, M. J. A., and Linoff, G., *Data Mining Techniques for Marketing, Sales and Customer Support*, Wiley, 1997.
3.7 Guha, S., Rastogi, R., and Shim, K., ROCK: A robust clustering algorithm for categorical attributes, *Information Systems*, 25(5):345–66, 2000.
3.8 Han, J., and Kamber, M., *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 2001.
3.9 Karypis, G., Han, E.-H., and Kumar, V., Chameleon: Hierarchical clustering using dynamic modeling, *IEEE Computer*, 32(8):68–75, Aug. 1999.
3.10 Pekalska, E., Paclik, P., and Duin, R. P. W., A generalized kernel approach to dissimilarity-based classification, *Journal of Machine Learning Research, Special Issue on Kernel Methods*, 2(2):175–211, 2002.
3.11 Karypis, G., and Kumar, V., A fast and high quality multilevel scheme for partitioning irregular graphs, *SIAM Journal on Scientific Computing*, 20(1):359–92, 1998.
3.12 Amari, S.-I., The EM algorithm and information geometry in neural network learning, *Neural Computation*, 7:13–8, 1995.
3.13 Strehl, A., Ghosh, J., and Mooney, R., Impact of similarity measures on web-page clustering, in *Proc. 17th National Conference on AI: Workshop on AI for Web Search (AAAI 2000)*, pp. 58–64, AAAI, July 2000.
3.14 Jain, A. K., and Dubes, R. C., *Algorithms for Clustering Data*, Prentice Hall, New Jersey, 1988.
3.15 Garey, M. R., and Johnson, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, CA, 1979.
3.16 Kernighan, B., and Lin, S., An efficient heuristic procedure for partitioning graphs, *Bell Systems Technical Journal*, 49:291–307, 1970.
3.17 Pothen, A., Simon, H., and Liou, K., Partitioning sparse matrices with eigenvectors of graphs, *SIAM Journal of Matrix Analysis and Applications*, 11:430–52, 1990.
3.18 Hendrickson, B., and Leland, R., An improved spectral graph partitioning algorithm for mapping parallel computations, *SIAM Journal on Scientific Computing*, 16(2):452–69, 1995.

3.19 Fiedler, M., A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory, *Czechoslovak Mathematical Journal*, 25:619–33, 1975.
3.20 Murtagh, F., *Multidimensional Clustering Algorithms*, Physica-Verlag, Heidelberg, Vienna, 1985.
3.21 Eisen, M. B., Spellman, P. T., Brown, P. O., and Botstein, D., Cluster analysis and display of genome-wide expression patterns, *Proc. Natl. Acad. Sci. USA*, 95:14863–8, Dec. 1998.
3.22 Banerjee, A., and Ghosh, J., Clickstream clustering using weighted longest common subsequences, in *Workshop on Web Mining: 1st SIAM Conference on Data Mining*, pp. 33–40, April 2001.
3.23 Cutting, D. R., Karger, D. R., Pedersen, J. O., and Tukey, J. W., Scatter/gather: A cluster-based approach to browsing large document collection, in *Proc. 15th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pp. 318–29, 1992.
3.24 Indyk, P., A sublinear-time approximation scheme for clustering in metric spaces, in *Proc. 40th Symposium on Foundations of Computer Science*, pp. 154–9, 1999.
3.25 Banerjee, A., and Ghosh, J., On scaling up balanced clustering algorithms, in *Proc. 2nd SIAM Intl Conf on Data Mining*, pp. 333–49, April 2002.
3.26 Zhang, T., Ramakrishnan, R., and Livny, M., BIRCH: A new data clustering algorithm and its applications, *Data Mining and Knowledge Discovery*, 1(2):141–82, 1997.
3.27 Ben-Dor, A., Shamir, R., and Yakhini, Z., Clustering gene expression patterns, *Journal of Computational Biology*, 6(3/4):281–97, 1999.
3.28 Bradley, P. S., Fayyad, U. M., and Reina, C., Scaling clustering algorithms to large databases, in *Knowledge Discovery and Data Mining*, pp. 9–15, 1998.
3.29 Gusfield, D. R., and Irving, R. W., *The Stable Marriage Problem: Structure and Algorithms*, MIT Press, Cambridge, MA, 1989.
3.30 Schloegel, K., Karypis, G., and Kumar, V., Parallel multilevel algorithms for multi-constraint graph partitioning, Technical Report 99-031, Dept. of Computer Sc. and Eng, Univ. of Minnesota, 1999.
3.31 Hartigan, J. A., *Clustering Algorithms*, Wiley, New York, 1975.
3.32 Chakaravathy, S. V., and Ghosh, J., Scale based clustering using a radial basis function network, *IEEE Transactions on Neural Networks*, 2(5):1250–61, Sept. 1996.
3.33 Kaufmann, L., and Rousseeuw, P., *Finding Groups in Data: An Introdution to Cluster Analysis*, John Wiley and Sons, 1990.
3.34 Rastogi, R. and Shim, K., Scalable algorithms for mining large databases, in Han, J. (ed.), *KDD-99 Tutorial Notes*, ACM, 1999.
3.35 Friedman, J. H., An overview of predictive learning and function approximation, in Cherkassky, V., Friedman, J. H., and Wechsler, H. (eds.), *From Statistics to Neural Networks, Proc. NATO/ASI Workshop*, pp. 1–61, Springer-Verlag, 1994.
3.36 Aggarwal, C., Re-designing distance functions and distance based applications for high dimensional data, *SIGMOD Record*, 30(1), March 2001.
3.37 Chakrabarti, K., and Mehrotra, S., The hybrid tree: An index structure for high dimensional feature spaces, in *Proc. ICDE*, pp. 440–7, 1999.
3.38 Han, E.-H., Karypis, G., Kumar, V., and Mobasher, B., Hypergraph based clustering in high-dimensional data sets: A summary of results, *Data Engineering Bulletin*, 21(1):15–22, 1998.

3.39 Boley, D., Gini, M., Gross, R., Han, E., Hastings, K., Karypis, G., Kumar, V., Mobasher, B., and Moore, J., Partitioning-based clustering for web document categorization, *Decision Support Systems*, 27:329–41, 1999.
3.40 Miller, G. L., Teng, S.-H., Thurston, W., and Vavasis, S. A., Separators for sphere packings and nearest neighbor graphs, *Journal of the ACM*, 44:1–29, 1997.
3.41 Meila, M., and Shi, J., Learning segmentation by random walks, in Leen, T. K., Dietterich, T. G., and Tresp, V. (eds.), *Advances in Neural Information Processing Systems 13*, pp. 873–9, MIT Press, 2001.
3.42 Dhillon, I., Modha, D., and Spangler, W., Visualizing class structure of multidimensional data, in Weisberg, S. (ed.), *Proc. 30th Symposium on the Interface: Computing Science and Statistics, Minneapolis, MN, May 13–6, 1998*, 1998.
3.43 Chang, K., and Ghosh, J., A unified model for probabilistic principal surfaces, *IEEE Trans. PAMI*, 23(1):22–41, Jan. 2001.
3.44 Torgerson, W. S., Multidimensional scaling, I: Theory and method, *Psychometrika*, 17:401–19, 1952.
3.45 Kohonen, T., The self-organizing map, in *New Concepts in Computer Science: Proc. Symp. in Honour of Jean-Claude Simon*, pp. 181–90, Paris, France, 1990, AFCET.
3.46 Berry, M. W., Hendrickson, B., and Raghavan. P., Sparse matrix reordering schemes for browsing hypertext, in *Lectures in Applied Mathematics (LAM)*, vol. 32, pp. 99–123. American Mathematical Society, 1996.
3.47 Keim, D. A., and Kriegel, H.-P., Visualization techniques for mining large databases: A comparison, *IEEE Transactions on Knowledge and Data Engineering*, 8(6):932–8, 1996, special issue on Data Mining.

4. Segmentation of Continuous Data Streams Based on a Change Detection Methodology

Gil Zeira[1], Mark Last[2], and Oded Maimon[3]

[1] Department of Industrial Engineering, Tel-Aviv University, Tel Aviv 69978, Israel; email: gil.zeira@ness.com

[2] Department of Information Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel; email: mlast@bgu.ac.il

[3] Department of Industrial Engineering, Tel-Aviv University, Tel Aviv 69978 Israel; email: maimon@eng.tau.ac.il

Most data mining algorithms assume that the historic data are the best estimator of what will happen in the future. As more data are accumulated in a database, one should examine whether the new data agrees with the model induced from previous instances. The problem of recognizing the change of the underlying model is known as a *change detection* problem. Once all change points have been detected, a data stream can be represented as a series of nonoverlapping *segments*.

This work presents a new methodology for change detection and segmentation based on a set of statistical estimators. While traditional segmentation methods are aimed at analyzing univariate time series, our methodology detects statistically significant changes in incrementally built classification models of data mining. In our previous work, we have shown the methodology to be valid for change detection in a set of artificial and benchmark data sets. In this work, we apply the change detection procedure to real-world data sets from two distinct domains (education and finance), where we detect significant changes between succeeding segments and compare the quality of alternative segmentations.

4.1 Introduction

The problems of *event detection* are concerned with recognizing either the change of parameter(s) in the model or the change of the model itself. The most common representation of a univariate time series is piecewise linear approximation. A straight line representing each segment can be found by linear interpolation or linear regression.

Change detection in time-series regression models has always been a topic of interest. For instance, Jones et al. [18] have developed a change-detection model mechanism for serially correlated multivariate data. Yao [38] has estimated the number of change points in time series using the BIC criterion. The bottom-up segmentation algorithm of Keogh [20] starts with a large number of equal-size segments and proceeds by merging two adjacent segments. Guralnik and Srivastava [12] use likelihood criteria to perform recursive binary partitioning of

the time segment. A sliding window bottom-up (SWAB) segmentation algorithm based on piecewise linear representation is presented in [21]. All these methods produce segmentation of a time series based on changes in a single parameter.

One known approach to deal with changes in classification models is using incremental or semi-incremental learning methods, which desire to achieve the following goals: (a) Decrease computational complexity as a function of time; (b) use prior knowledge to determine future conclusions; (c) update or completely re-train the model to improve its accuracy; and (d) reduce model complexity, for instance, network size, decision tree size, and depth or a set of extracted rules. These methods include: incremental concept learning with bounded example memory (Case et al. [2]), Utgoff 's [35], [36] method for incremental induction of decision trees (ITI), Shen's [33] semi-incremental learning method (CDL4), Cheung's [5] technique for updating association rules in large databases, Gerevini's [11] network constraints updating technique, Zhang's [40] method for feed-forwarding neural networks (SELF), incrementally trained connectionist networks (Martinez [27]), and a simple backpropagation algorithm for neural networks (Mangasarian and Solodov [28]).

The main topic in most incremental learning methods has been how the model (this could be a set of rules, a decision tree, neural networks, and so on) is refined or reconstructed efficiently as new amounts of data are encountered. This problem has been challenged by many of the preceding algorithms. However, the real question is: W*hen* should we discard the current model and reconstruct a new one, because something in our notion of the model has significantly changed? Hence, the problem is not *how* to reconstruct better, but alternatively, how to detect a change in a model based on accumulated data.

Learning in the presence of change is not a new concept in the data mining research area. Some researchers have studied various aspects of mining massive nonstationary data streams, including:

- defining robustness and discovering robust knowledge from databases (Hsu and Knoblock [16]) or learning stable concepts in domains with hidden changes in concepts (Harries and Horn [13]);
- identifying and modeling a persistent drift in a database (Freund and Mansour [10]);
- adapting to concept and population drift (Helmbold and Long [14], Hulten et al. [17], Kelly et al. [19], Lane and Brodley [23], Widmer and Kubat [37]);
- activity monitoring (Fawcett and Provost [8]); and
- improving accuracy, algorithm run time and noise reduction by Partitioning, Arbitering or Combining Models methods (Ali and Pazzani [1], Chan and Stolfo [3-4], Domingos [6]).

These methods do not challenge directly the problem of detecting significant changes. Rather they deal with environment, which changes over time.

This chapter introduces a novel methodology for detecting a significant change in a classification model of data mining by implementing a set of statistical estimators. Detection of a change implies that the model induced from a sufficiently large data set is no longer valid for use such as prediction or rule induction and alternatively, a new model, representing a subsequent time segment, must be constructed.

The rest of the chapter is organized as follows. In Section 4.2 we present the methodology for a change-detection procedure in data-mining models. Section 4.3 describes empirical evaluation of the change detection and segmentation method on real-world data sets and Section 4.4 concludes the chapter by summing up the main contributions of our method and presenting several options for future implementation and extension of our methodology.

4.2 Change Detection in Classification Models

4.2.1 Characteristics of the Classification Task in Data Mining

Classification is the task that involves constructing a model (generally consisting of categorical class labels) in a classifying attribute and using it to classify new data (Fayyad et al. [9]).

A classification model can be represented as a set of rules (see the RISE system [7], the BMA method for rule induction [34], PARULEL and PARADISER [32], etc.), a decision tree (see [35], [36]), Quinlan's C4.5 and ID3 and their variations), neural networks (see [28], [40]), information-theoretic connectionist networks (see [26]), and so on.

Given a database D that consists of X sets of records, the data-mining classification model M (induced by an algorithm G) is a set of hypotheses H within the available hypothesis space that generalizes the relationship between a set of predicting variables and the target variable. The following notation is a general description of the data-mining classification modeling task.

Given a database D containing a complete set of records $X = (A|T)$, where A is a vector of candidate variables (attributes from the examined phenomenon, which might have some influence on the target concept) and T is a target variable (i.e., the target concept), **find** the best set of hypotheses H within the available hypotheses space, which generalizes the relationship between a set of candidate variables and the target variable (e.g., the model M) using some data-mining algorithm.

We regard each record as a complete set of conjunction between attributes and a target concept (or variable), such as $X_i = (A_{1i} = a_{1j(1)} , A_{2i} = a_{2j(2)} , \ldots , A_{ni} = a_{nj(n)} , T_i = t_j')$. $A = (A_1 , A_2 . . . A_n)$ is a known set of attributes of the desired phenomenon and T (also noted in DM literature as Y) is a known discrete or continuous target variable or concept. It is obvious that not all attributes have to be incorporated in the set of generated hypotheses.

In most algorithms the database D is divided into two parts – learning ($D_{(learn)}$) and validation ($D_{(val)}$) sets. One is supposed to hold enough information to assemble a statistically significant and stable model based on the DM algorithm. The second part is supposed to ensure that the algorithm performs its goals by validation of the built models on unseen records. Evaluating the predictive accuracy of a model M, which is built by a classification algorithm G, is

commonly performed by estimating the validation error rate of the examined model. This is calculated by the following equation:

$$\frac{\sum_{X_{val}} H(G)_{X_{val}} \text{ is wrong}\big(T(\text{predicted}) \neq T(\text{actual})\big)}{N_{X_{val}} (\text{number of records in the validation set})} \tag{4.1}$$

When the database D is not fixed but is accumulated over time, the data-mining task should be altered: In every period K, a new set of records X_K is added to the database; d_K is the set of records X_K that was added in the start of period K, and D_K is the accumulated database $D_K = \bigcup_{k=1}^{K} d_k$. Therefore, Given a database D_K containing a complete set of records X_K, generate the best set of hypotheses H_K to describe the accumulated model M. At the end of time period K+1, a new question is encountered: Is $M_{K,G} = M_{K+1,G}$, for every K?

As noted before, several methods dealing with this problem have already been proposed by researchers. Most methods have dealt with "how the model M can be updated efficiently when a new period K is encountered" or "how we can adapt to the time factor," rather than asking the following questions:

1. Was the model significantly changed during the period K?
2. What was the nature of the change?
3. Should we consider several of the past periods as redundant or not required for an algorithm G to generate a better model M?

Hence, the objective is to define and evaluate a change-detection methodology for identifying a significant change that happened during period K in a data-mining model, which was incrementally built in periods 1 to $K-1$, based on the data that was accumulated during period K.

4.2.2 Variety of Changes

There are various significant changes that might occur when building the model M based on the algorithm G. There are several possible causes for significant changes in a data-mining classification model:

1. A change in the distribution of one or more of the candidate attributes (A). For example, if a database in periods 1 to K-1, consists of a 45% male and 55% female examples and in period K all records of male samples.
2. A change in the distribution of the target variable (T). For example, consider the case of examining the rate of failures in a seminar examination based on the characteristics of the students in the course in past consecutive years. If in 1999 the average was 20% and in 2000 it was 40%, then a change in the target distribution has occurred.

3. A change in the "patterns" (rules) that define the relationship of the inputs to the target variable, that is, a change in the model M. For instance, in the case of examining the rate of failures in a seminar exam based on the characteristics of the students in the course in consecutive years : if in 1999 male students produced 60% of the failures and female students produced 5%, and in 2000 the state was opposite, then it is obvious that there was a change in the patterns of behavior of the data-mining model. This suggests a significant change in a data-mining model M by the following definition: A change *C* is encountered during period *K* if the validation error of the model M_{K-1} (the model that is based on D_{K-1}) on the database D_{K-1} is significantly different from the validation error rate of the model M_{K-1} over d_K. d_K consists of the accumulated set of records in period *K*.
4. Instability of the DM algorithm. The basic assumption is that the DM algorithms the users choose to achieve their goal (their decision) were proven to be stable and therefore are not likely to be a major cause for a significant change in the model. Although this option should not be omitted, this work does not intend to deal with stability or instability measures of existing DM algorithms, rather than point out that an assumption of the stability of a chosen algorithm should be set before implementing any decisions based on the induced model. As indicated in the next section, the DM algorithm that was used in our experiments (IFN) was proven to be stable.

As noted in Section 4.2.1 the data-mining classification model is generally described as: $M_G : A \neg T$, that is, finding the "right" connection between *A* and *T*. The first cause is explained via a change in the set of attributes *A* and can also cause a change in the target variable (for example, if the percentage of women rises from 50% to 80% and women drink more white wine than men, then the overall percentage of white wine consumption (*T*) will also rise) and can also affect the overall error rate of the data-mining model if most "laws" were generated for men. Also, if the cause of a change in the relevant period is a change in the target variable (for example, France has stopped producing white wine, and the percentage of white wine consumption has dropped from 50% to 30%), it is possible that the laws between the relevant attributes to the target variable(s) will be affected.

Therefore, it is not obvious to state which cause of change will be more significant than others. Alternatively, this work states that there are a variety of possible changes that might occur in a relevant period. This change can be caused by one of the three causes mentioned earlier or by a simple combination of them.

Table 4.1 consists of the possible combinations of significant causes in a relevant period.

Table 4.1. Definition of the variety of changes in a data-mining model.

"Rules"	A	T	Details
-	-	-	No change.
-	-	+	A change in the target variable.
-	+	-	A change in the attribute variable(s).
-	+	+	A change in the target and in the attribute variable(s).
+	-	-	A change in "patterns" (rules) of the data-mining model.
+	-	+	A change in "patterns" (rules) of the data-mining model and a change in the target variable.
+	+	-	A change in "patterns" (rules) of the data-mining model and a change in the attribute variable(s).
+	+	+	A change in "patterns" (rules) of the data-mining model and a change in the target and the attribute(s) variable.

The definition of the variety of possible changes in a data-mining model is a new concept. As noted, several researchers tended to deal with concept change (e.g., target), population change (e.g., candidate effecting target), activity monitoring (e.g., model), etc. The new notion that all three major causes interact and affect each other is tested and validated in this work.

4.2.3 Statistical Hypothesis Testing

To determine whether a significant change has occurred during period *K*, a set of statistical estimators is presented in this chapter. The use of these estimators is subject to several conditions: (a) Every period contains a sufficient amount of data to rebuild a model for that specific period. The decision of whether a period contains sufficient data (records) should be based on the relationship between the training and validation error rate of every period and is subjective for different users, for acceptable range in difference, overfitting, and so on. (b) The same DM algorithm is used in all periods (i.e., the data-mining model in every period *K* was constructed based on the same DM algorithm). (c) The same validation method is used in all periods (e.g., one of the following: five-fold, 10-fold, 1/3 of the set of records).

4.2.3.1 Statistical Hypothesis for the Model Change Detection

The first estimator for the change-detection methodology is designed to detect a change in the "patterns" (rules) that defines the relationship of the candidate input to the target variable, that is, a change in the model M, derived from a change in a set of hypothesis in H. Because we assume that huge amounts of data are involved in building the incremental model, it is simple to assume that the true validation

error rate of a given incremental model is accurately estimated by the previous *K*-1 periods. Therefore, a change in the rules (R) is encountered during period *K* if the validation error of the model $M_{K\text{-}1}$ (the model based on $D_{K\text{-}1}$) on the database $D_{K\text{-}1}$ is significantly different from the validation error rate of the model $M_{K\text{-}1}$ over d_K.

Therefore, the parameter of interest for the statistical hypothesis testing is the true validation error rate, and the null hypothesis for testing is as follows:

$$\begin{aligned} H_0 &: \hat{e}_{M_{K-1},K} = e_{Val} \left(= \hat{e}_{M_{K-1},K-1}\right) \\ H_1 &: \hat{e}_{M_{K-1},K} \neq e_{Val} \left(= \hat{e}_{M_{K-1},K-1}\right), \end{aligned} \tag{4.2}$$

where $\hat{e}_{M_{K-1},K-1}$ is the validation error rate of $D_{K\text{-}1}$ set of records on model $M_{K\text{-}1}$ (the standard validation error of the model); $\hat{e}_{M_{K-1},K}$ is the validation error rate of the set of records d_K on the aggregated model $M_{K\text{-}1}$; and e_{Val} is the true validation error rate of the incremental model, based on *K*-1 periods.

To detect a significant difference between two error rates (see also Mitchell [29]), it is needed to use the Eq. (4.2). The objective of this test is to test the difference between two independent proportions based on the approximation to the normal distribution.

The hypothesis decision is measured by the following equations (two-sided hypothesis):

$$\begin{aligned} &\hat{\sigma}_d^2 = \frac{\hat{e}_{M_{K-1},K}\ (\,1\ -\ \hat{e}_{M_{K-1},K}\,)}{n_K} + \frac{\hat{e}_{M_{K-1},K-1}\ (\,1\ -\ \hat{e}_{M_{K-1},K-1}\,)}{n_{K-1(val)}} \\ &\left|\hat{d}\right| = \left|\hat{e}_{M_{K-1},K}\ -\ \hat{e}_{M_{K-1},K-1}\right| \\ &\text{If } \left|\hat{d}\right| \geq z_{1-\alpha/2} \cdot \sqrt{\hat{\sigma}_{\ddot{d}}^2} \text{ then do not accept } H_0. \end{aligned} \tag{4.3}$$

where $\hat{e}_{M_{K-1},K-1}$ is the validation error rate of $D_{K\text{-}1}$ set of records on model $M_{K\text{-}1}$ (the standard observed validation error of the model); $n_{K\text{-}1(\text{val})} = |D_{K\text{-}1(\text{val})}|$ is the number of records selected for validation from periods 1, ... , $K-1$; $\hat{e}_{M_{K-1},K}$ is the observed validation error rate of the set of records d_K on the aggregated model $M_{K\text{-}1}$; and $n_K = |d_K|$ is the number of records in period *K*.

4.2.3.2 Statistical Hypothesis for the Distribution Change Detection

The objective of the second estimator is to validate the assumption that a variable(s)'s population (target or candidate) has significantly changed in a statistical sense. For this purpose, we use Pearson's estimator for testing matching

proportions of variables (Hines and Montgomery [15]). This estimator examines whether an empirical distribution of a variable matches a known probability distribution of that variable. Again, because we assumed that a huge amount of data were involved in building the incremental model, it is reasonable to assume that the true population distribution of any variable in the given incremental model is accurately estimated by the previous $K-1$ periods.

The objective is to decide whether to accept the following null hypothesis for every variable *X* of interest:

H_0: the variable *X*'s population is stationary between periods.
H_1: otherwise.

The decision is made by the following equation:

$$X_p^2 = n_K \cdot \sum_{i=1}^{j} \frac{\left(\frac{x_{iK}}{n_K} - \frac{x_{iK-1}}{n_{K-1}}\right)^2}{\frac{x_{iK-1}}{n_{K-1}}}. \tag{4.4}$$

If $X_p^2 > \chi_{1-\alpha}^2(j-1)$ then the base assumption that the variable *X*'s distribution has been stationary in period *K* is not accepted (see also Montgomery and Runger [30]).

In Eq. (4.4); n_K is the number of records in the *K*th period; n_{K-1} is the number of records in periods 1, ... , $K-1$; x_{iK} is the number of records belonging to the *i*th class of variable *X* in the *K*th period; x_{iK-1} is the number of records belonging to the *i*th class of variable *X* in periods 1,..., $K-1$.

4.2.4 Methodology

This section describes the algorithmic usage of the previous estimators.

Inputs:	G	is the algorithm used for the DM classification model.
	M	is the model used for DM representation.
	V	is the validation method in use.
	K	is the number of periods.
	α	is the desired confidence level for the procedure.
Outputs:	CD (α)	is the Change Detection estimator 1 – Pvalue.
	XP (α)	is the Pearson's estimator 1 – Pvalue.

Change-Detection Procedure

Stage 1:

For periods $K-1$ build M_{K-1} using DM algorithm G.

Define $D_{K-1(\text{val})}$.

Count $n_{K-1} = \left|D_{K-1(\text{val})}\right|$.

Calculate $\hat{e}_{M_{K-1},K-1}$ according to V.

Calculate x_{iK-1}, n_{K-1} for every candidate and target variable existing in periods (1,..., $K-1$).

Stage 2:

For period *K*, define d_K.

Count $n_K = \left|d_K\right|$.

Calculate $\hat{e}_{M_{K-1},K}$ according to V.

Calculate $d = \text{ABS}(\hat{e}_{M_{K-1},K} - \hat{e}_{M_{K-1},K-1})$, $\hat{\sigma}_{\ddot{d}}^2$, $H_0 = z_{1-\alpha/2} \cdot \sqrt{\hat{\sigma}_{\ddot{d}}^2}$.

Calculate and return CD(α).

Stage 3:

For every candidate and target variable existing in periods (1,..., $K-1$) and in period *K* calculate: x_{iK}, n_K, and X_p^2.

Calculate and return XP(α).

It is obvious that the complexity of this procedure is at most O(n_K). It is very easy to store information about the various distributions of the target and candidate variables to simplify the change-detection methodology.

Using the outputs of the methodology the user can make a distinction among the eight possible variations of a change in the data-mining classification model. According to this new information the user of the new methodology can act in several ways : For example, the user can reapply the algorithm from scratch absorbing the new period and using the same incremental algorithm, making $K' = K+1$ and performing the procedure again. The user can also investigate the type of the change and its magnitude and effect on the other characteristics of the DM model, and incorporate other known methods dealing with the specific detected changes. One can also incorporate multiple-model approaches such as weighting, arbitrage, and combining methods, and use the prior knowledge of the change.

The methodology is not restricted to databases with a constant number of variables. The basic assumption is that if the addition of a new variable influences the connection between that target variable and the candidate variables in a manner that inflicts on the validation accuracy (V is the method to select

validation records and is not an equation), then it will be revealed as a significant change.

The procedure has three major stages. The first is designed to perform an initiation of procedure. The second stage is designated to detect a significant change in the "patterns" (rules) of the prebuilt data-mining model, as described in the previous section. The third stage is designated to evaluate whether one or some variable(s) in the group of candidate attributes or target variable(s) (A and T) show a significant change between periods.

The basic assumption for using the procedure is the availability of sufficient data for each run of the algorithm on every period. If this assumption is not valid, it is necessary to merge two or more periods to obtain statistically significant outcomes.

4.3 Application Evaluation

4.3.1 Data Set Description

The method was proven useful when run on artificially generated data sets. The method for change detection was also evaluated on several benchmark data sets (Zeira et al. [39]).

An example of the implementation of the change-detection methodology is illustrated in the first set of experiments, which were performed on a database obtained from a network of colleges in Israel. This data set describes yearly (e.g., the time periods) dropouts of students from technicians and technical engineering colleges (we refer to this data set as "Dropout"). The candidate attributes are: regional area of the colleges (REGION), a discrete categorical variable; number of divisions of studies in the institute (DIVISIONS), a discrete variable; number of students in the institute (SUMP), a discretized variable where each value X describes the interval $[40(X-1), 40X]$; average number of students in class (AVEP), discretized to two intervals (low and high); percent of technological reserve students in the institute (TR_PER), discretized to two intervals (high and low); and class of students (CLASS), a discrete categorical variable (technicians studies and technical engineers studies). The target factor (DROPOUT) describes dropout percentage in the institute (high, low, negative). Dropout represents students who have not finished their studies according to the pre-defined curriculum of their class.

The "Dropout" database represents data for a five-year period. It is common that due to organizational and social trends in the society, some changes in the data-mining model are expected after the model becomes stable. Therefore, the base assumption for this data set is that significant changes would be observed over time.

The second set of experiments has been performed on a stock market data set, initially used in Last et al. [24] for evaluation of the IFN algorithm. The raw data represent the daily stock prices of 373 companies from the Standard & Poor's 500

index over a five-year period (from 8/29/94 to 8/27/99) and it has been obtained from the Microsoft™ MoneyCentral Web site. In Last et al. [24], we have applied signal-processing techniques to partition each series of daily stock values into a sequence of intervals having distinct slopes (trends). An average of 15.64 intervals per company has been identified. The classification problem has been defined as predicting the correct length of the current interval based on the known characteristics of the current and preceding intervals. Consequently, we have converted every sequence of *m* intervals related to a specific stock into *m–1* interval-pairs, each containing information about two consecutive intervals. This resulted in a total of 5462 records of interval-pairs. The candidate input attributes include the duration, slope, and fluctuation measured in each interval, as well as the major sector of the corresponding stock (a static attribute). The target attribute, which is the duration of the second interval in a pair, has been discretized to five subintervals of nearly equal frequency. These subintervals have been labeled very short, short, medium, etc. To restore the original order of data arrival, we have sorted the records by the starting date of each interval (we refer to this data set as "Stock").

4.3.2 Results – "Dropout" Data Set

This section summarizes five consecutive yearly periods processed by the IFN algorithm, which have proven to produce stable data-mining models (Last et al. [25]). The base assumption for this data set was that significant changes would be observed over time, due to organizational changes, increasing demand for technological degrees, etc. Table 4.2, Table 4.3, Table 4.4, and Fig. 4.1 describe the outcomes of implementing the IFN algorithm on five consecutive years in the database "Dropout" using the change-detection methodology to detect significant changes that have occurred during these years.

Table 4.2. Results of the CD hypothesis testing on the "Dropout" database.

Year	CD				
	$^{e}M_{K-1,K}$	$^{e}M_{K-1K-1}$	d	H(95%)	1 – Pvalue
1996	-	-	-	-	-
1997	42.2%	17.4%	24.8%	6.5%	100.0%
1998	54.8%	31.2%	23.6%	6.2%	100.0%
1999	35.0%	29.9%	5.1%	5.4%	87.7%
2000	42.3%	21.9%	20.4%	4.9%	100.0%

Table 4.3. Results of the XP hypothesis testing on the "Dropout" database (1–*P*value).

Year	Class	Reg ion	Divisions	Sump	*A*vep	Tr_Per	Dropout
1997	98.6%	100%	67.1%	100.0%	99.6%	96.6%	99.8%
1998	99.8%	76.7%	88.6%	99.6%	51.1%	100%	100%
1999	99.1%	99.8%	99.6%	100%	97.7%	99.9%	100%
2000	93.1%	98.9%	100%	100%	96.4%	65.9%	100%

Table 4.4. Outcomes of implementing the change detection methodology on "Dropout" database (1–*P*value).

Year	CD	XP(target)
1996	-	-
1997	100.0%	99.8%
1998	100.0%	100.0%
1999	82.2%	100.0%
2000	100.0%	100.0%

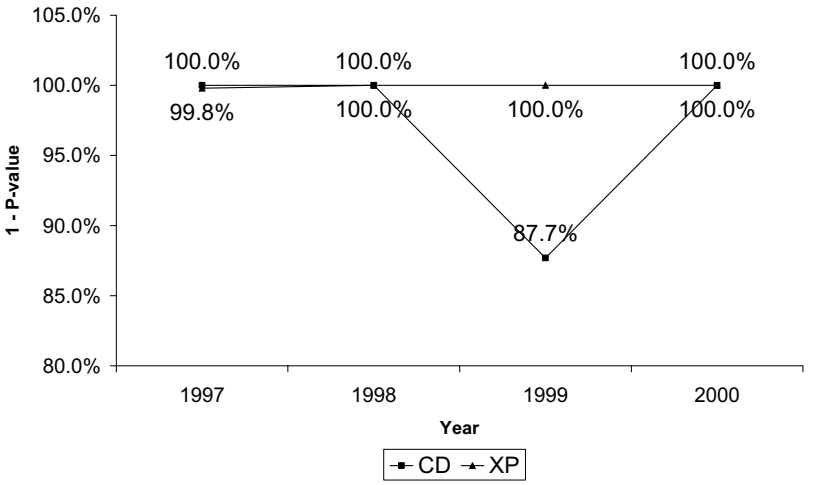


Fig. 4.1. Summary of implementing the change detection methodology on "Dropout" database (1–*P*value).

A statistical analysis of the first four years, which in most cases reveals statistically significant result over most common hypothesis tests and analysis, will derive a *stable* model. This is obvious as the change-detection metric falls to only 87% significance for the four years of accumulated data. Usually the basic inductive learning hypothesis states that any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over unobserved examples.

But analysis of the fifth year data does not support this hypothesis. The change-detection method reveals that the model, which was observed over the first four years, is not stationary. All three metrics, which are combined through the change-detection procedure, detect a highly significant process change (more than 1% confidence level).

To Illustrate how our method detects real-world significant changes, we quote the manager of science and technology adminstration in the Ministry of Education, Culture and Sports in Israel, as delivered at the end of year 2000: "The administration has finished the new planning of organizing studies for the technological road, many courses has been altered . . . We are preparing for a pioneer experiment, which involves about 40 educational institutes, in which the programs, their implementation and integration will be evaluated."

To further demonstrate the impact of the change in the models from years 1996 to 1999, opposed to the model using the five years of data, illustrated in Fig. 4.2 is the decision tree, which is derived by the IFN algorithm from the data set that includes 1996 to 2000 (according to the largest connection weight of the extracted set of fuzzy rules (M. Last, et al. [25]). The bolded nodes of the tree represent states, which involve different forecasts from including or excluding the year 2000 from the database.

The expected error rate from using the same set of rules, based on 1996 to 1999 over the year 2000 and beyond, will produce at least 22% error on average, as shown in Table 4.5.

Table 4.5. Comparison of decision trees induced from "Dropout" data set including and excluding year 2000.

Layer no.	No. of rules	Mismatches	Mismatch percentage
0	1		0%
1	2		0%
2	12	3	25%
3	27	5	19%
4	19	7	37%
5	6		0%
6	2		0%
sum	69	15	22%

Class = 1: 1
Region = 0: 1
Region = 1: 1
Region = 2: 1
Region = 3: 1
Population = 0: 1
Population = 1: 0
Population = 2: 1
Population = 3: 1
Population = 4: 1
Region = 4: 1
Population = 0: 1
Divisions = 0: 1
Divisions = 1: 2
Population = 3: 2
Divisions = 1: 1
Divisions = 2: 2
Population = 4: 1
Region = 5: 1

Class = 0: 1
Region = 0: 1
Region = 1: 1
Population = 0: 2
Divisions = 0: 2
Divisions = 1: 0
Population = 1: 1
Divisions = 0: 1
Divisions = 1: 1
Divisions = 2: 2
Population = 2: 1
Divisions = 1: 2
Divisions = 2: 1
Divisions = 3: 1
Region = 2: 2
Population = 0: 2
Population = 1: 1
Population = 2: 2
Divisions = 2: 2
Divisions = 3: 1
Region = 3: 1
Population = 0: 1
Population = 1: 1
Population = 2: 2
Population = 3: 1
Population = 4: 2
Region = 4: 1
Population = 0: 2
Population = 1: 1
Population = 2: 1
Divisions = 1: 1
Divisions = 2: 1
Divisions = 3: 1
Population = 3: 0
Population = 4: 1
Population = 5: 1
Region = 5: 2
Population = 0: 2
Divisions = 0: 2
Divisions = 1: 1
Population = 1: 1

Fig. 4.2. Decision tree based on years 1996 - 2000 from “Dropout” database.

4.3.3 Results – "Stock" Data Set

This section summarizes segmentation and analysis of the stock data set, which was analyzed by the IFN algorithm. The main expectation for this dataset was that significant changes would be observed over time, due to the segmentation of the full data sets into disjoint segments. This indicates that the full data stream can be evaluated as several disjoint data sets, and for each of them, a separate underlying model can be evaluated and implemented.

The full data set holds information about stocks in 5462 records. As there is no predefined way to segment the given data stream, three different ways of segmentation were implemented and evaluated based on the change-detection methodology.

The logical way of segmenting the data stream is using any "time field" as an indication of the accumulating knowledge, which was added incrementally to the database. Table 4.6 describes how the segments of data sets are divided according to the incremental date field in the stock data set.

Table 4.5. Segmentation of the "stock" data set.

Trial num.	Segment num.	Record interval
1	1	[1,1000]
	2	[1001,2000]
	3	[2001,3000]
	4	[3001,4000]
	5	[4001,5000]
2	1	[1,1500]
	2	[1501,3000]
	3	[3001,4500]
	4	[4501,5000]
3	1	[1,2000]
	2	[2001,2500]
	3	[2501,3000]
	4	[3001,3500]
	5	[3501,4000]
	6	[4001,4500]
	7	[4501,5000]

The first trial is a partition of 5000 accumulated records into five equally sized data sets. Figure 4.3 describes the outcome of applying the change-detection methodology to these segments of data.

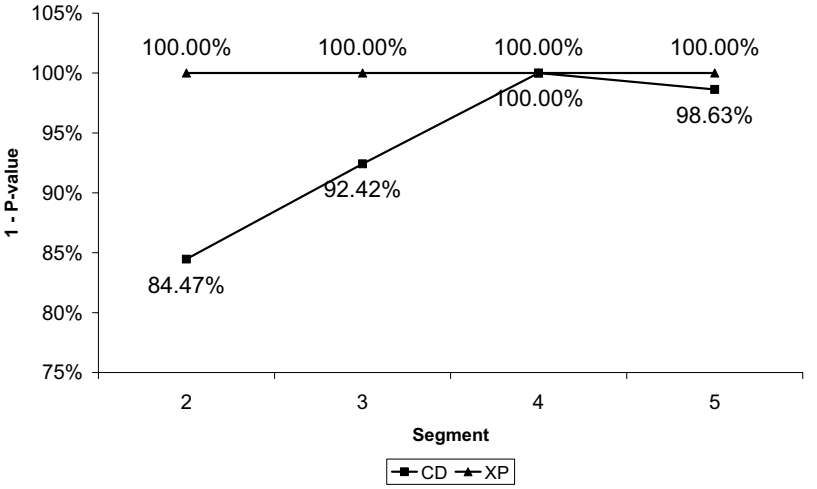


Fig. 4.3. Implementation of CD and XP on trial number 1 in "Stock" Database.

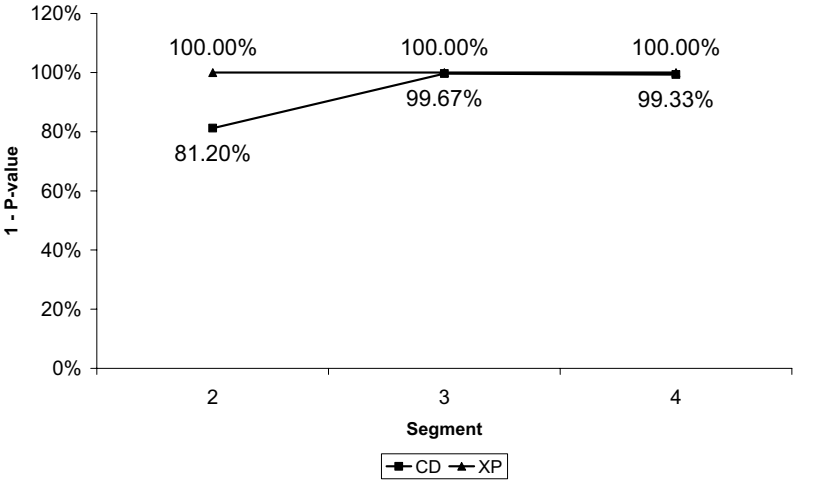


Fig. 4.4. Implementation of CD and XP on trial number 2 in "Stock" Database.

The second trial is a partition of 5000 accumulated records into four equally sized data sets. The assumption for this kind of segmentation, as opposed to the first one, is that the more data the segment holds, the better the incremental data-mining model is. Figure 4.4 describes the outcome of implementing the change-detection methodology on these segments of data.

Opposed to the first two trials, the third trial is based on the assumption that the first segment should hold enough information to validate the base data-mining

model and that the following segments of information should be smaller to indicate a segment of information that is not fully compatible or totally incompatible with the base data-mining model. Figure 4.5 depicts the outcome of implementing the change-detection methodology on these segments of data.

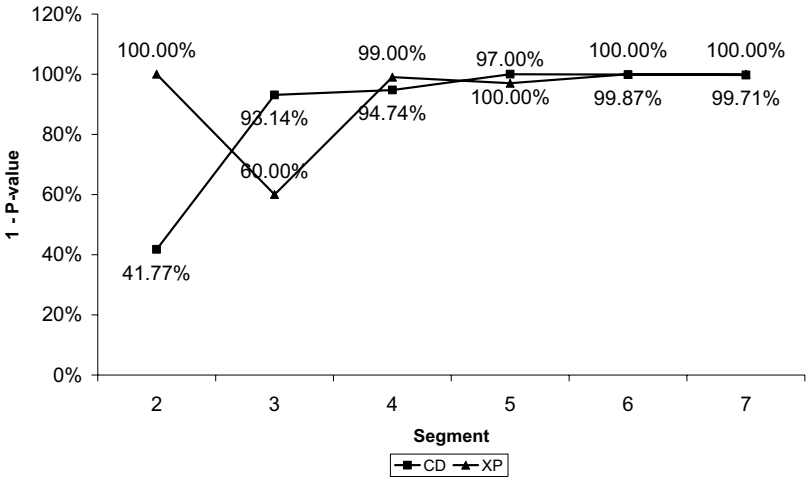


Fig. 4.5. Implementation of CD and XP on trial number 2 in 'Stock' Database.

Our objective is to find the best suitable segmentation. Therefore, we need to evaluate which of the three trials produced a "better" segmentation, based on the outcomes of the change-detection methodology. When evaluating these outcomes we mainly consider the parameter CD, which describes the "level of fitness" of a new data segment to previously built classification models of data mining.

Based on the major outcomes of the change-detection methodology, an evaluation of the best segmentation is provided by the accumulated parameters of the model detection. The summary is presented in Fig. 4.6.

In Fig. 4.6 a statistical analysis of the CD parameter in all three trials is described. The analysis describes three possible segmentations for the given data stream. Deciding what is the best possible segmentation is the user's choice. It is possible to use one or a combination of the statistics. The possible statistical parameters, which are used in this set of experiments, are:

1. Deciding on a better segmentation, based on the range of $1 - P_{\text{value}}(CD)$ over all segments of a data stream. Range is calculated by the following equation: $Range = Max(1 - P_{\text{value}}(CD)) - Min(1 - P_{\text{value}}(CD))$. A higher range denotes a better segmentation.
2. Deciding on a better segmentation, based on the standard deviation of $1 - P_{\text{value}}(CD)$ over all segments of a data stream. Greater standard deviation denotes a better segmentation.

3. Deciding on a better segmentation, based on the average of $1-P_{\text{value}}(CD)$ over all segments of a data stream. A higher average denotes a better segmentation.
4. Deciding on a better segmentation, based on the percentage of segments where $1-P_{\text{value}}(CD)$ exceeds the minimal 0.5% confidence level out of all the segments of a data stream. A higher percentage denotes a better segmentation.

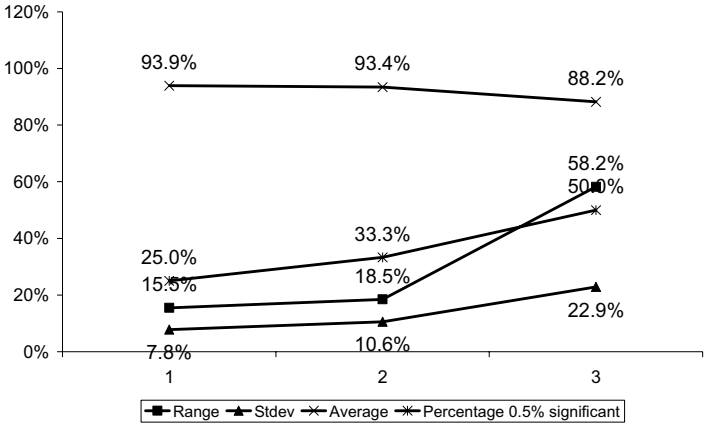


Fig. 4.6. Analysis of $1-P_{\text{value}}(CD)$ in three trials of segmentation in "stock" data set.

In this example, for instance, a relevancy rank is assigned to all statistical parameters; a higher rank (1 to 3) describes a better score in a statistical parameter. The overall score is calculated as a weighted average and the outcome is described in Table 4.7. The weighting schema of all the parameters should be a choice of the user of the segmentation model.

Table 4.6. Evaluating segmentation in the "stock" data set.

Trial	Range	Standard deviation	Average	Percentage 0.5% significant	Score
Weight	25%	25%	25%	25%	100%
1	1	1	3	1	1.5
2	2	2	2	2	2
3	3	3	1	3	2.5

It is obvious that in this case the third trial describes the best segmentation out of three trials based on the change-detection methodology. But the question whether or not this is the best possible segmentation within a range of various types and lengths of segmentation in the specific "stock" data, cannot be answered

by these outcomes. To answer this specific question, a preferred algorithm for searching the optimal (or suboptimal) segmentation should rely on the following assumptions and characteristics:

1. The complexity of the IFN algorithm for data mining like most classification data-mining algorithms is O(n). This should be taken into consideration.
2. There should be a limit on the number of possible segmentations and a minimal size for each segment. Otherwise, the change-detection method would not be useful due to insufficient information in each segment.
3. The choice of set of statistical parameters and weights should be considered.
4. An initial segmentation should be implemented (a simple partition of k segments) and then explored into a relevant segmentation method by merging and dividing segments.
5. The search algorithm should have a stopping criterion. Also, the search method can be one of many search methods available (greedy, golden section, genetic algorithms, etc.).
6. An automated segmentation procedure should have capabilities for user interaction in the segmentation process (for example, see Nouira and Fouet [31]).

4.3.4 Summary of Experiments

The following statements summarize the results:

1. In the "Dropout" database, the change-detection procedure reveals significant changes in the extracted data-mining model, which was built from the data accumulated during 2000, validating the base assumption for this database.
2. In the "Dropout" database, the expected error rate of using the same set of rules, based on 1996–1999 on the year 2000 and beyond, would produce at least 22% error on average.
3. By applying the change-detection approach to the "stock" data set, we have detected significant changes between succeeding segments and have compared the quality of two alternative segmentations to provide a better segmentation of the data set.
4. It is shown in the "stock" data set that a better segmentation of a data stream can be chosen based on a statistical analysis and ranking schema.
5. Our change-detection methodology may be utilized as a basis for an automated procedure aimed at finding the best segmentation of a given data stream but it may be computationally expensive.

4.4 Conclusions and Future Work

As mentioned earlier, many data-mining models are constructed based on the assumption that the data involved in building and verifying the model are the best estimators of what will happen in the future.

The important factor that must not be set aside is the time factor. As more data is accumulated into the problem domain, incrementally over time, one must examine whether the new data agree with the previous data sets and make the relevant assumptions about the future.

This work presented a novel change-detection method for detecting significant changes in data for building data-mining models. We also addressed the data-segmentation problem.

The major contributions of this research to the area of data mining and KDD are:

1. This work defines three main causes for a statistically significant change in a data-mining model:
   - a change in the distribution of one or more of the candidate variables attributes (A),
   - a change in the distribution of the target variable (T), and
   - a change in the "patterns" (rules) that define the relationship of the candidate input to the target variable. That is, a change in the model M.

   This work showed that although there are three main causes for significant changes in the data-mining classification models, it is common that these main causes will interact with each other, deriving eight possible combinations for a significant change in an aggregated data-mining model. Moreover, the effect of these causes is not the same for all databases and algorithms.
1. The change-detection method relies on the implementation of two statistical estimators: change-detection hypothesis testing (CD) of every period K and Pearson's estimator (XP) for testing matching proportions of variables.
2. The effect of a change is relevant and can be mainly detected at the period of the change. If not detected, the influence of a change in a database will be reflected in successive periods.
3. The change-detection method has a low computation cost. Its complexity is $O(n_K)$ due to only checking whether the new data agree with the prior aggregated model.
4. By implementing the change detection in a data stream, we can detect significant changes between succeeding segments and decide on a better segmentation of a data stream based on a statistical analysis and ranking schema.
5. Our change-detection methodology may be used as a basis for an automated procedure aimed at finding the best segmentation of a given data stream by integrating the change-detection methodology with a search algorithm.

The change detection procedure with the use of the statistical estimators can detect significant changes in classification models of data mining. These changes can be detected independently of the data-mining algorithm (e.g., C4.5 and ID3 by Quinlan; KS2, ITI and DMTI by Utgoff [35-36]; IDTM by Kohavi [22]; Shen's CDL4 [33]) or the DM classification model (rules, decision trees, networks, etc.), which are used for constructing the corresponding model.

The major contribution of the change-detection methodology as described, is the introduction of a new methodology for change detection and the implication of the eight possible changes in the data-mining models. The implication of this novel method in the field of data stream segmentation is described. These notions are defined, and a specific change-detection procedure is designed to solve the change-detection problem, based on a set of statistical hypothesis testing. Also, the methodology was implemented as a new method of finding segmentation in a data stream.

As change detection is quite a new application area in the field of classification models of data mining, many issues are left to be investigated:

1. Implementing voting techniques according to the cause(s) and magnitude(s) of a change(s) detected in period K for combining several models, such as exponential smoothing, voting weights based on the CD confidence level, neglecting old periods or problematic periods, etc.
2. Integrating the change-detection methodology in an existing data-mining method. As seen, the change-detection procedure's complexity is only O(n). One could integrate this procedure with an existing incremental learning algorithm, which will continue efficiently rebuilding an existing model if the procedure detects a significant change in the newly obtained data. This option is also applicable for meta-learning and combining methods.
3. Implementing the methodology in a new search algorithm for finding an optimal segmentation of a data stream with respect to a variety of data-mining models.
4. Using the CD statistical hypothesis testing for specific attribute monitoring.

References

[1] Ali, K., and Pazzani, M., Learning multiple descriptions to improve classification accuracy, *International Journal on Artificial Intelligence Tools*, 4, 1995.

[2] Case, J. et al., Incremental concept learning for bounded data mining, *Proc. Automata Induction, Grammatical Inference, and Language Acquisition,* Workshop at the 14th International Conference on Machine Learning (ICML-97), 1997, and *Information and Computation*, 152(1), 74–110, 1999.

[3] Chan, P. K., and Stolfo, S. J., A comparative evaluation of voting and meta-learning on partitioned data, in *Proc. 12th Intl. Conf. on Machine Learning*, 90–8, 1995.

[4] Chan, P. K., and Stolfo, S. J., Sharing learned models among remote database partitions by local meta-learning, in *Proc. 2nd Intl. Conf. on Knowledge Discovery and Data Mining*, 2–7, 1996.

[5] Cheung, D., Han, J., Ng, V., and Wong, C. Y., Maintenance of discovered association rules in large databases: An incremental updating technique, in *Proc. of 1996 Int'l Conf. on Data Engineering* (ICDE'96), New Orleans, LA, 1996.

[6] Domingos, P., Knowledge acquisition from examples via multiple models, in *Proc. 14th International Conference on Machine Learning*, 98–106, 1997.
[7] Domingos, P., Using partitioning to speed up specific-to-general rule induction, in *Proc. AAAI-96 Workshop on Integrating Multiple Learned Models*, 29–34, 1996.
[8] Fawcett, T., and Provost, F., Activity monitoring: Noticing interesting changes in behavior, in *Proc. 5th International Conference on Knowledge Discovery and Data Mining* (KDD-99), 1999.
[9] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P., Knowledge discovery and data mining: Towards a unifying framework, in *Proc. 2nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD-96), 1996.
[10] Freund, Y., and Mansour, Y., Learning under persistent drift, in *Proc. EuroColt 1997*, 109–18, 1997.
[11] Gerevini, A., Perini, A., and Ricci, F., Incremental algorithms for managing temporal constraints, *IEEE International Conference on Tools with Artificial Intelligence* (ICTAI'96), 1996.
[12] Guralnik, V., and Srivastava, J., Event detection from time series data, in *Proc. 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 33–42, 1999.
[13] Harries, M., and Horn, K., Learning stable concepts in domains with hidden changes in context, in *Proc. ICML-96 Workshop on Learning in Context-Sensitive Domains*, Bari, Italy, 21–31, 1996.
[14] Helmbold, D. P., and Long, P. M., Tracking drifting concepts by minimizing disagreements, *Machine Learning*, 14 (1), 27–46, 1994.
[15] Hines, W. H., and Montgomery, D. C., *Probability and Statistics in Engineering and Management Science,* 3rd edition, Wiley, 1990.
[16] Hsu, C. N, and Knoblock, C. A., Discovering robust knowledge from databases that change, *Journal of Data Mining and Knowledge Discovery*, 2(1), 1–28, 1997.
[17] Hulten, G., Spencer, L., and Domingos, P., Mining time-changing data streams, in *Proc. 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD-2001), 2001
[18] Jones, R. H., Crowell, D. H., Kapuniai, L. E., Change detection model for serially correlated multivariate data, *Biometrica*, 26:269–80, 1970.
[19] Kelly, M. G., Hand, D. J., and Adams, N. M., The impact of changing populations on classifier performance, in *Proc. 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD-99), 1999.
[20] Keogh, E., Fast similarity search in the presence of longitudinal scaling in time series databases, in *Proc. Tools with Artificial Intelligence*, 578–84, 1997.
[21] Keogh, E., Chu, S., Hart, D., and Pazzani, M., An online algorithm for segmenting time series, in *Proc. of IEEE International Conference on Data Mining*, 289–96, 2001.
[22] Kohavi, R., The power of decision tables, in *European Conference on Machine Learning* (ECML1995), 1995.

[23] Lane, T., and Brodley, C. E., Approaches to online and concept drift for user identification in computer security, in *Proc. 4th International Conference on Knowledge Discovery and Data Mining*, New York, 259–63, 1998.

[24] Last, M., Klein, Y., and Kandel, A., Knowledge discovery in time series databases, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 31: Part B, No. 1, 160–9, 2001.

[25] Last, M., Maimon, O., and Minkov, E., Improving stability of decision trees, *International Journal of Pattern Recognition and Artificial Intelligence*, 16 (2), 145–59, 2002.

[26] Maimon, O., and Last, M., *Knowledge Discovery and Data Mining, the Info-Fuzzy Network (IFN) Methodology*, Kluwer Academic Publishers, 2000.

[27] Martinez, T., Consistency and generalization in incrementally trained connectionist networks, in *Proc. International Symposium on Circuits and Systems*, 706–9, 1990.

[28] Mangasarian, O. L., and Solodov, M. V., Backpropagation convergence via deterministic nonmonotone perturbed mininization, in *Advances in Neural Information Processing Systems*, 6, 383–90, 1994.

[29] Mitchell, T. M., *Machine Learning*, Carnegie Mellon University, McGraw Hill, 1997.

[30] Montgomery, D. C., and Runger, G. C., *Applied Statistics and Probability for Engineers,* 2nd edition, Wiley, 1999.

[31] Nouira, R., and Fouet, J.M., A knowledge based tool for the incremental construction, validation and refinement of large knowledge bases, in *Preliminary Proceedings of Workshop on Validation, Verification and Refinement of BKS* (ECAI96), 1996.

[32] Ohsie, D., Hasanat, M. D., Stolfo, S. J., and Da Silva, S., Performance of Incremental Update in Database Rule Processing, In *J. Widom, S. Chakravarthy (Eds.): Proc. Fourth International Workshop on Research Issues in Data Engineering: Active Database Systems, Houston, Texas, February 14-15,* 10-18, 1994 .

[33] Shen, W. M., An active and semi-incremental algorithm for learning decision lists, Technical Report, USC-ISI-97, Information Sciences Institute, University of Southern California, 1997.

[34] Shen, W. M., Bayesian probability theory – A general method for machine learning, from MCC-Carnot-101-93, Microelectronics and Computer Technology Corp., Austin, TX, 1997.

[35] Utgoff, P.E., An improved algorithm for incremental induction of decision trees, in *Machine Learning: Proc. 11th International Conference*, 318–25, 1994.

[36] Utgoff, P. E., Decision tree induction based on efficient tree restructuring, Technical Report 95-18, Department of Computer Science, University of Massachusetts, 1995.

[37] Widmer, G., and Kubat, M., Learning in the presence of concept drift and hidden contexts, *Machine Learning*, 23(1), 69–101, 1996.

[38] Yao, Y., Estimating the number of change points via Schwartz' criterion, *Statistics and Probability Letters*, 181–9, 1988.

[39] Zeira, G., Maimon, O., Last, M., and Rokach, L., Change detection in classification models of data mining, in *Proc. 12th Israeli Conf. of Industrial Engineering and Management*, 2002.

[40] Zhang, B. T., An incremental learning algorithm that optimizes network size and sample size in one trial, in *Proc. of Int. Conf. on Neural Networks* (ICNN-94), IEEE, 215–20, 1994.

5. Instance Selection Using Evolutionary Algorithms: An Experimental Study

José Ramón Cano, [1] Francisco Herrera, [2] and Manuel Lozano[2]

[1] Dept. of Computer Science, Escuela Politecnica Superior de Linares, University of Jaén, 23700 Jaén, Spain; email: jrcano@decsai.ugr.es

[2] Dept. of Computer Science and Artificial Intelligence, Escuela Tecnica Superior de Ingenieria Informatica, University of Granada, 18071 Granada, Spain; email: herrera, lozano@decsai.ugr.es

In this chapter, we carry out an empirical study of the performance of four representative evolutionary algorithm models considering two instance-selection perspectives, the prototype selection and the training set selection for data reduction in knowledge discovery. This study includes a comparison between these algorithms and other nonevolutionary instance-selection algorithms. The results show that the evolutionary instance-selection algorithms consistently outperform the nonevolutionary ones, offering two main advantages simultaneously, better instance-reduction rates and higher classification accuracy.

5.1 Introduction

The digital technologies and computer advances with booming Internet use have led to massive data collection and information. Research in areas of science from astronomy to the human natural genome is facing the same problem choking on information. Raw data are rarely of direct use, and manual analysis simply cannot keep pace with the fast growth of data. Knowledge discovery (KD) [34] and data mining (DM) [1] help us; they aim to turn raw data into nuggets and create special edges.

KD processes include *problem comprehension*, *data comprehension*, *data preprocessing*, *DM*, *evaluation,* and *development* [1], [8], [35]. The first three processes (problem and data comprehension and data preprocessing) play a pivotal role in successful DM.

Due to the enormous amounts of data, much of the current research is based on *scaling up* DM algorithms. Research has also worked on *scaling down* data. The major issue of scaling down data is to select the relevant data and then present them to a DM algorithm [25]. This task is developed in the data-preprocessing phase in the KD process.

Data preprocessing presents the following strategies: *data reduction*, *data cleaning*, *data construction*, *data integration*, and *data format change*. Our attention is focused on *data reduction*. Data reduction can be achieved in many ways:

- By *selecting features* [6], [24], we reduce the number of columns in a data set.
- By *discretizing feature-values* [14], we reduce the number of possible values of discretized features.
- By *selecting instances* [6], [26], we reduce the number of rows in a data set. Instance selection (IS) is a focusing task in the data-preparation phase [8] of KD. IS may comprise following different strategies: *sampling*, *boosting*, *prototype selection* (PS), and *active learning*.

The topic of this chapter is precisely the IS [6], [27], [33], [40] by means of evolutionary algorithms (EAs) for data reduction in KD.

EAs [2], [3], [13] are general-purpose search algorithms that use principles inspired by natural genetic populations to evolve solutions for problems. The basic idea is to maintain a population of chromosomes, which represent candidate solutions for the specific problem, which evolves over time through a process of competition and controlled variation. EAs have been designed to solve the IS problem, given promising results [4], [19], [21], [28], [32], [41].

The goal of this chapter is to present the application of some representative EA models for data reduction and to compare them with nonevolutionary IS algorithms (classical ones in the following). To do this, we carry out our study from two different points of view:

- *IS-PS*: Analyzing the results when they are used for prototype selection for classification, where 1-NN is applied to evaluate the classification percentage offered by the training set selected. We are going to denote IS-PS (*Instance Selection – Prototype Selection*) to this point of view.
- *IS-TSS*: Analyzing the behavior of EAs as instance selectors for data reduction, applying C4.5 [31] to evaluate the training set selected. We are going to denote IS-TSS (*Instance Selection – Training Set Selection*) to this approach.

The second one is, really, the most important aspect and novelty in this chapter, to analyze the behavior of EAs for data reduction in KD. In particular, we introduce a stratified EA model for evaluating this approach.

The chapter is organized as follows. In Section 5.2, we explain the main ideas of IS, introduce a brief historical review, and describe two processes where IS algorithms take part, the IS-PS and the IS-TSS. In Section 5.3, we survey the main classical IS algorithms. In Section 5.4, we introduce the foundations of the EAs and summarize the basic features of the models considered in this chapter. In Section 5.5, we provide details on the way EAs may be applied to the IS problem. In Section 5.6, we deal with the methodology for the experiments. In Section 5.7, we include the results of the experiments and their analysis. Finally, in Section 5.8, we present some concluding remarks.

5.2 Instance Selection

In IS we want to isolate the smallest set of instances that enable us to predict the class of a query instance with the same (or higher) accuracy as the original set [26]. By reducing the "useful" data set size, which can reduce both space and time complexities of subsequent processing phases. One can also hope to reduce the size of formulas obtained by a subsequent induction algorithm on the reduced and less noisy data sets. This may facilitate interpretation tasks.

IS raises the problem of defining relevance for a prototype subset. From the statistical viewpoint, relevance can be partly understood as the contribution to the overall accuracy, which would be obtained by a subsequent induction. We emphasize that removing instances does not necessarily lead to a degradation of the results: We have observed experimentally that a small number of instances can have performances comparable to those of the whole sample, sometimes higher. Two reasons come to mind to explain such an observation. First, some noises or repetitions in data could be deleted by removing instances. Second, each instance can be viewed as a supplementary degree of freedom. If we reduce the number of instances, we can sometimes avoid over-fitting situations. With this definition there are two types of irrelevant instances we should remove:

- The first are instances belonging to regions with very few elements; their vote is statistically a poor estimator, and a little noise might affect their vote dramatically. It is also common in statistical analysis to search and remove such points, in regression, parametric estimations, etc. Removing them does not necessarily brings a great reduction in the size of the retained instances, but it may be helpful for future prediction tasks.
- The second are instances belonging to regions where votes can be assimilated as being randomized. Local densities are approximately evenly distributed with respect to the overall class distributions. These instances are not necessarily harmful for prediction, but a great reduction in size can be obtained after removing some of them if they are numerous. Depending on the IS method, all or some of them would be removed.

5.2.1 Related Work

Historically, IS has been aimed first at improving the efficiency of the nearest neighbor (NN) classifier [9]. The NN algorithm is one of the most venerable algorithms in machine learning [10]. To classify a new instance, the Euclidean distance (possibly weighted) is computed between this instance and each training neighboring instance, and the new instance is assigned the class of the nearest neighboring instance. More generally, the k nearest neighbor (k-NN) is computed, and the new instance is assigned the class that is most frequent among these k neighbors. The use of the k-NN classifier was also spread and encouraged by early theoretical results linking its generalization Bayes error.

However, from a practical point of view, the NN algorithm is not suited to very large data sets because of the storage requirements it imposes and the computational overhead involved. Actually, this approach involves storing all the instances in memory. Pioneer work in IS firstly tired to reduce the storing size. Next, we introduce a brief historical review about this topic. The algorithms used in this study will be described in Section 5.3.

Hart [17] proposes a *condensed NN rule* to find a consistent subset, which correctly classifies all of the remaining points in the sample set. However, this algorithm will not find a minimal consistent subset [39].

The *reduced NN rule* proposed by Gates [15] tries to overcome this drawback, searching in Hart's consistent set for the minimal subset that correctly classifies all the learning instances. However, this approach is efficient if and only if Hart's consistent set contains the minimal consistent set of the learning set, which is not always the case. It is worthwhile remarking that in these two approaches, the IS algorithm deduces only one instance subset. It is impossible to save more or fewer instances and to control the size of the subset.

Wilson [38] proposes *edited NN* and *repeated edited NN*. Edited NN edits out noisy instances and close border cases, leaving smoother decision boundaries. It also retains all internal points, which keeps it from reducing the storage requirements as much as most other reduction algorithms. The repeated edited NN continues to widen the gap between classes and to smooth the decision boundary.

Kibbler and Aha [20] propose an algorithm similar to the reduced NN. It retains border points, but unlike reduced NN, it is sensitive to noise. It is called the *shrink algorithm*.

Brodley [7] proposes the *MCS system* (model class selection system) to deal with the IS problem. MCS systems tend to avoid noise.

Wilson and Martinez [39] present a family of editing algorithms guided by the sets for each instance: the k nearest neighbors and the associates of the instance. An associate of the instance P is each of the instances that has P as one of its k nearest neighbors. The family is composed of five algorithms, from *DROP1* to *DROP5*. The first three methods, *DROP1–3*, were previously introduced by the authors under the name RT1–3, respectively. These algorithms pretend to produce instance reductions that provide noise tolerance, high-generalization accuracy, insensitivity to the order of presentation of instances, and significant storage reduction, which in turn improves the generalization and speed.

5.2.2 Strategies for Instance Selection

In this section, we describe two strategies for IS followed in the study presented in this chapter.

5.2.2.1 Instance Selection for Prototype Selection (IS-PS)

This strategy consists of prototype selection having the classification as objective.

Prototype selection (PS): The 1-NN classifiers predict the class of a previously unseen instance by computing its similarity to a set of stored instances called *prototypes*. PS – storing a well-selected, proper subset of available training instances – has been shown to increase classifier accuracy in many domains [10]. At the same time, using prototypes dramatically decreases storage and classification-time costs.

A PS algorithm is an IS algorithm that attempts to obtain a subset of the training set that allows the 1-NN classifier to achieve the maximum classification rate. Figure 5.1 shows the process where a PS algorithm acts.

A large number of approaches for PS algorithms have tried to identify these salient instances that are stored by a 1-NN classifier; some are surveyed in Section 5.3.

5.2.2.2 Instance Selection for Training Set Selection (IS-TSS)

In this section, we describe the IS as data reduction having to obtain a training set selection as objective.

Training set selection (TSS). There may be situations where there are too many data points; almost always these data are not equally useful in the training phase of a learning algorithm [32]. It is intuitively clear that those data points that fall near the decision boundary between two classes are likely to be more influential on the DM algorithm than points that are well inside. Similarly, if several points from the same class are very close to each other, the information they convey is virtually the same, so are they all necessary? IS mechanisms have been proposed for choosing the most suitable points in the data set that should become instances for the training data set used by a learning algorithm. For example, in [32], a genetic algorithm is used for training data selection in radial basis function networks. In [30], there is a comprehensive study on the general question of training data selection in the context of function approximation (i.e., regression problems).

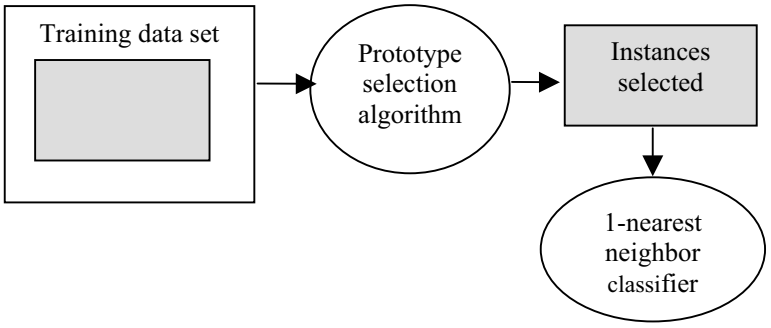


Fig. 5.1. IS-PS strategy.

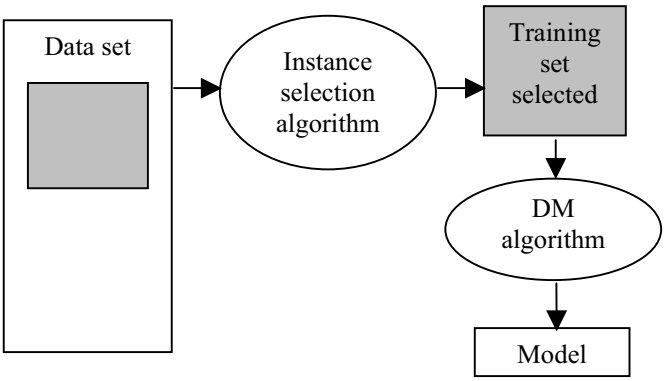


Fig. 5.2. IS-TSS strategy.

Figure 5.2 shows a general framework for the application of an IS algorithm for TSS. Starting from the data set *D*, the IS algorithm finds a suitable training set selected, *S*.

In this work, we use EAs for IS-TSS following a *stratified* approach [27], [33], which is outlined in Figure 5.3. The data set, *D*, is divided into two non-overlapping sets with the 50% of the elements, *T1* and *T2* ($D = T1 \cup T2$ and $T1 \cap T2 = \varnothing$), which are classically called *strata*. Then an IS algorithm may be applied on them independently, obtaining two sets with the selected instances, *S1* and *S2*. The final training set will be the union of these sets ($S = S1 \cup S2$).

This technique seems adequate for applying EAs as IS algorithms to DM problems with large data sets (see Section 5.5.1).

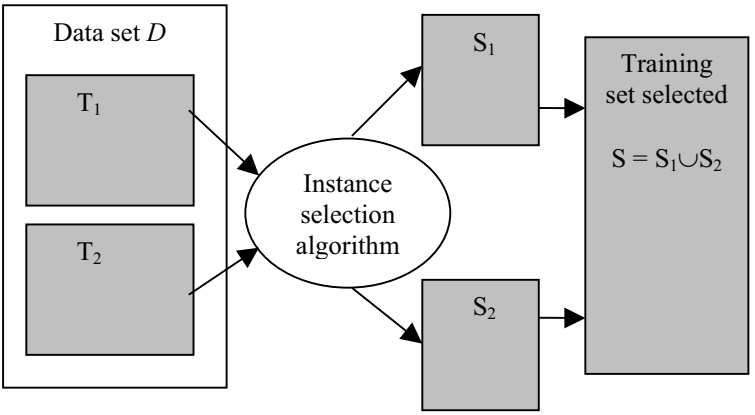


Fig. 5.3. Stratified approach for IS-TSS.

5.3 Survey of Instance Selection Algorithms

This section surveys several techniques, discusses them in light of the framework presented in Section 5.2.1, and points out their interesting differences. Most of the models also tend to use $k = 1$ (1–NN), where k is the number of neighbors evaluated, except where noted, although in most cases the algorithms can be modified to use $k > 1$. These algorithms use a subset S of the original instances in the training set T as their representation and primarily use the Euclidean distance function.

5.3.1 Nearest Neighbor Editing Rules

In this section, we find the classic algorithms based on the nearest neighbor rule.

5.3.1.1 Condensed Nearest Neighbor (CNN) [17]

This algorithm finds a subset S of the training set T such that every number of T is closer to a member of S of the same class than to a member of S of a different class. In this way, the subset S can be used to classify all the instances in T correctly.

This algorithm begins by randomly selecting one instance belonging to each output class from T and putting them in S. Then each instance in T is classified using only instances in S. If an instance is misclassified, it is added to S, thus ensuring that it will be classified correctly. This process is repeated until there are no instances in T that are misclassified. This algorithm ensures that all instances in T are classified correctly, but it does not guarantee a minimal set.

5.3.1.2 Edited Nearest Neighbor (ENN) [11]

In this algorithm S starts out the same as T, and then each instance in S is removed if it does not agree with the majority of its k nearest neighbors (with $k = 3$, typically). This edits out noisy instances and close border cases, leaving smoother decision boundaries. It also retains all internal points, which keeps it from reducing the storage requirements as much as most other reduction algorithms.

5.3.1.3 Repeated Edited Nearest Neighbor (RENN) [38]

The repeated ENN (RENN) applies the ENN algorithm repeatedly until all instances remaining have a majority of their neighbors with the same class, which continues to widen the gap between classes and smooths the decision boundary.

5.3.2 Instance-Based Learning Algorithms

Instance-based learning techniques essentially work by keeping typical attribute examples for each class.

5.3.2.1 Model Class Selection (MCS) [7]

The idea of this algorithm is to keep track of how many times each instance was one of the k nearest neighbors of another instance and whether its class matched that of the instance being classified. If the number of times it was wrong is greater than the number of times it was correct, then it is thrown out.

5.3.2.2 Shrink Algorithm [20]

Kibbler and Aha presented an algorithm that starts with $S = T$ and then removes any instances that would still be classified correctly by the remaining subset. This is similar to the reduced nearest neighbor (RNN) rule, except that it only considers whether the removed instance would be classified correctly, whereas RNN considers whether the classification of other instances would be affected by the instance's removal. Like RNN and many of the other algorithms, it retains border points, but unlike RNN, this algorithm is sensitive to noise.

5.3.3 Ordered Removal

This section presents a collection of new heuristics used to decide which instances to keep and which instances to remove from a training set. Unlike most previous methods, these algorithms take careful note of the order in which instances are removed. The first three methods, DROP1–3, were previously introduced by the authors under the name RT1–3, respectively.

5.3.3.1 Decremental Reduction Optimization Procedure 1 (DROP1) [39]

DROP1 uses the following basic rule to decide if it is safe to remove an instance from the instance set (where $S = T$ originally):

Remove P if at least as many of its associates in S would be classified correctly without P.

To see if an instance P can be removed using this rule, each associate (each instance that has P as one of its neighbors) is checked to see what effect the removal of P would have on it. Removing P causes each associate $P.A_i$ to use its $k + 1$ nearest neighbor ($P.A_i.N_{k+1}$) in place of P. If P has the same class as $P.A_i$ and $P.A_i.N_{k+1}$ has different than $P.A_i$, this weakens its classification and could cause $P.A_i$ to be misclassified by its neighbors. On the other hand, if P is a

different class from $P.A_i$ and $P.A_i.N_{k+1}$ is the same class as $P.A_i$, the removal of P could cause a previously misclassified instance to be classified correctly.

In essence, this rule tests to see if removing P would degrade leave-one-out cross-validation generalization accuracy, which is an estimate of the true generalization ability of the resulting classifier.

5.3.3.2 Decremental Reduction Optimization Procedure 2 (DROP2) [39]

DROP2 resolves the problem of noisy instances that affects DROP1. It solves this problem by considering the effect of the removal of an instance on all the instances in the original training set T instead of considering only those instances remaining in S. In other words, an instance P is removed from S only if at least as many of its associates – including those that may have already been removed from S – are classified correctly without it.

Thus, the removal criterion can be restated as:

Remove P if at least as many of its associates in T would be classified correctly without P.

DROP2 also changes the order of removal of instances. It initially sorts the instances in S by the distance to their nearest enemy. Instances are then checked for removal beginning at the instance furthest from its nearest enemy. This tends to remove instances furthest from the decision boundary first, which in turn increases the chance of retaining border points.

5.3.3.3 Decremental Reduction Optimization Procedure 3 (DROP3) [39]

DROP2 has one problem. Noisy instances are also "border" points and cause the order of removal to be drastically changed. One noisy point in the center of a cluster causes many points in that cluster to be considered border points, and some of these can remain in S even after the noisy point is removed. DROP3 uses a noise-filtering pass before sorting the instances in S. This is done using the rule:

Any instance misclassified by its k nearest neighbors is removed.

There are DROP4 and DROP5 algorithms but they offer minimal variations from DROP3.

5.4 Evolutionary Algorithms

EAs [2], [3], [13] are stochastic search methods that mimic the metaphor of natural biological evolution. All EAs rely on the concept of a *population* of individuals (representing search points in the space of potential solutions to a given problem), which undergo probabilistic operators such as *mutation*, *selection*,

and (sometimes) *recombination* to evolve toward increasingly better fitness values of the individuals. The *fitness* of an individual reflects its objective function value with respect to a particular objective function to be optimized. The mutation operator introduces innovation into the population by generating variations of individuals, and the recombination operator typically performs an information exchange between different individuals from a population. The selection operator imposes a driving force on the process of evolution by preferring better individuals to survive and reproduce when the members of the next generation are selected.

The reason for a great part of the success of EAs is their ability to exploit the information accumulated about an initially unknown search space in order to bias subsequent searches into useful subspaces, i.e., their adaptation. This is their key feature, particularly in large, complex, and poorly understood search spaces, where classical search tools (enumerative, heuristic, etc.) are inappropriate. In such cases, they offer a valid approach to problems requiring efficient and effective search techniques.

Next, we describe the four EA models that will be used in this chapter as evolutionary IS algorithms. They are:

- two standard GA models: the *generational* GA (GGA) and the *steady-state* GA (SGA) [36];
- the *CHC* algorithm [12], which has been tested in many GA works against other different GA approaches, giving better results, especially on hard problems [37]; and
- the *population based incremental learning* (PBIL) algorithm [5], which is an algorithm that uses a probabilistic model for driving the search toward the most promising regions. This idea constitutes a profitable research topic in the EA field by using probabilistic models [29].

5.4.1 Generational Genetic Algorithm (GGA) [18], [16]

GAs are general purpose search algorithms that use principles inspired by natural genetic populations to evolve solutions to problems. The basic idea is to maintain a population of chromosomes, which represent candidate solutions to the concrete problem, that evolves over successive iterations (generations) through a process of competition and controlled variation. Each chromosome in the population has an associated fitness to determine which chromosomes are to be used to form new ones in the competition process. This is called selection. The new ones are created using genetic operators such as *crossover* and *mutation*.

Although there are many possible variants of the basic GA, the classical model is the GGA, which consists of three operations:

1. Evaluation of individual fitness.
2. Formation of an intermediate population through a selection mechanism.
3. Recombination through crossover and mutation operators.

The next algorithm shows the structure of a basic GGA. *P(t)* denotes the population at generation *t*:

```
Generational Genetic Algorithm
  Begin
     t=0;
     Initialize P(t);
     Evaluate P(t);
     While (Not termination-condition) do
          Begin
               t=t+1;
               Select P(t) from P(t-1);
               Recombine P(t);
               Evaluate P(t);
           End;
  End;
```

The selection mechanism produces a new population, *P(t)*, with copies of chromosomes in *P(t-1)*. The number of copies received for each chromosome depends on its fitness; chromosomes with higher fitness usually have a greater chance of contributing copies to *P(t)*. Then the crossover and mutation operators are applied on *P(t)*.

Crossover takes two individuals called parents and produces two new individuals called the offspring by swapping parts of the parents. In its simplest form, the operator works by exchanging substrings after a randomly selected crossover point. The crossover operator is not usually applied to all pairs of chromosomes in the new population. A random choice is made, where the likelihood of crossover being applied depends on the probability defined by a *crossover rate*.

Mutation serves to prevent premature loss of population diversity by randomly sampling new points in the search space. Mutation rates are kept small however, otherwise the process degenerates into a random search. To bit strings, mutation is applied by flipping one or more random bits in a string with a probability equal to the *mutation rate*.

Termination may be triggered by reaching a maximum number of generations or by finding an acceptable solution by some criterion.

5.4.2 Steady-State Genetic Algorithm (SGA) [36]

In SGAs usually only one or two offspring are produced in each generation. Parents are selected to produce offspring and then a replacement/deletion strategy defines which member of the population will be replaced by the new offspring. The basic algorithm steps of SGA are the following:

1. Select two parents from the population *P*.
2. Create an offspring using crossover and mutation.
3. Evaluate the offspring with the fitness function.
4. Select an individual in *P*, which may be replaced by the offspring.
5. Decide if this individual will be replaced.

In Step 4, one can choose the replacement strategy (e.g., replacement of the worst, the oldest, or a randomly chosen individual). In Step 5, one can choose the replacement condition (e.g., replacement if the new individual is better or unconditional replacement). A widely used combination is to replace the worst individual only if the new individual is better (this is the one we have assumed in our experiments).

The major difference between SGAs and GGAs is that for each *P* members of the population generated in the GGA there are 2•*P* selections. Consequently, the selection strength and generic drift for an SGA is twice that of the GGA. The SGA, therefore, appears twice as fast, although it can lose out in the long term because it does not explore the landscape as well as the GGA.

5.4.3 CHC Algorithm [12]

During each generation, the CHC algorithm uses a parent population of size *N* to generate an intermediate population of *N* individuals, which are randomly paired and used to generate *N* potential offspring. Then a survival competition is held where the best *N* chromosomes from the parent and offspring populations are selected to form the next generation.

CHC also implements a form of heterogeneous recombination using HUX, a special recombination operator. HUX exchanges half of the bits that differ between parents, where the bit positions to be exchanged are randomly determined. CHC also employs a method of incest prevention. Before applying HUX on two parents, the Hamming distance between them is measured. Only those parents that differ from each other by some number of bits (*mating threshold*) are mated. The initial threshold is set at *L*/4, where *L* is the length of the chromosomes. When no offspring are inserted into the new population the threshold is reduced by 1.

No mutation is applied during the recombination phase. Instead, when the population converges or the search stops making progress (i.e., the difference threshold has dropped to zero and no new offspring are being generated that are better than any members of the parent population), the population is reinitialized to introduce new diversity to the search. The chromosome representing the best solution found over the course of the search is used as a template to reseed the population. Reseeding of the population is accomplished by randomly changing 35% of the bits in the template chromosome to form each of the other *N*-1 new chromosomes in the population. Search is then resumed.

The CHC generally does well with small populations. Limited resources and the computational cost of the simulations led to our use of small populations and selection of the CHC for this work.

5.4.4 Population-Based Incremental Learning (PBIL) [5]

PBIL is a combination of GAs and competitive learning. It was designed for binary search spaces. The PBIL algorithm attempts to explicitly maintain statistics about the search space to decide where to sample next.

The objective of the algorithm is to create a real-valued probability vector, V_p, which, when sampled, reveals high-quality solution vectors with high probability. For example, if a good solution can be encoded as a string of alternating 0's and 1's, a suitable final V_p would be 0.01, 0.99, 0.01, 0.99, etc. The values of V_p are initialized to 0.5. Sampling from this vector yields random solution vectors because the probability of generating a 1 or 0 is equal. As the search progresses, the values in V_p gradually shift to represent high evaluation solution vectors through the following process:

1. A number of solution vectors (N_{samples}) are generated based on the probabilities specified in V_p.
2. V_p is pushed toward the generated solution vector with the highest evaluation, S_{best}. This is accomplished as follows:

$$V_p[i]=V_p[i] \bullet (1\text{-}LR) + S_{\text{best}}[i] \bullet LR, \tag{5.1}$$

where LR is the learning rate, which specifies how large the steps toward the best solution are.

3. After the probability vector is updated, a new set of solution vectors is produced by sampling from the updated probability vector, and the cycle is continued.

Furthermore, PBIL applies mutations on V_p, with a purpose analogous to mutation in GAs: to inhibit premature convergence. Mutations perturb V_p with a small probability, P_m, in a random direction, *Mut_Shif*.

5.5 Evolutionary Instance Selection

EAs may be applied to the IS problem, because it may be formulated as a search problem. In this chapter, these EAs have been called *evolutionary IS algorithms*. Examples of these algorithms may be found in [4], [21], [22], [23], [19], [41], which are concerned with the application of the GAs to PS, and in [32], which is concerned with the application of the GAs to TSS.

As we have mentioned, the objective of this chapter is to study the performance of the four EAs described in the previous section as IS algorithms applied to PS and to TSS, comparing their results with the ones obtained by the classical algorithms introduced in Section 5.3.

The application of EAs to these two approaches is accomplished by tackling two important issues: the specification of the representation of the solutions and the definition of the fitness function.

5.5.1 Representation

Let us assume that T is a data set with m instances. The search space associated with the IS of T is constituted by all the subsets of T (the subsets are denoted S). Then the chromosomes should represent subsets of T. This is accomplished by using a binary representation. A chromosome consists of m genes (one for each instance in T) with two possible states: 0 and 1. If the gene has 1, then its associated instance is included in the subset S represented by the chromosome. If it has 0, then this does not occur.

This representation may not be effective for the application of the IS to TSS problems with large databases because the chromosomes will have too many genes. In this case, the EAs may have problems driving the search toward the better regions. For these cases, we propose using a *stratified evolutionary model* based on Figure 5.3 (Section 5.2.2), which applies an evolutionary IS algorithm, independently, on two nonoverlapping partitions of the data set (even, more than two disjoint partitions may be considered). In this way, the chromosomes will have fewer genes, and the effectiveness of the EAs may be improved. Furthermore, this mechanism may be generalized by considering a greater number of partitions, which will depend on the number of instances of the data set.

5.5.2 Fitness Function

Let $S \subseteq T$ be a subset of instances to evaluate. We define a fitness function that combines two values: the classification performance associated with S and the percentage of reduction of instances of S with regard to T:

$$\text{Fitness}(S) = \alpha \cdot clas\_per + (1-\alpha) \cdot porc\_red. \tag{5.2}$$

The 1-NN classifier (Section 5.2.1) is used for measuring the classification rate, *clas_per*, associated with S. It denotes the percentage of correctly classified objects from T using only S to find the nearest neighbor. For each object x in T the nearest neighbor is searched among those in the set S, without considering the proper x when $x \in S$. On the other hand, *porc_red* is defined as:

$$porc\_red = 100 \cdot \frac{|T| - |S|}{|T|}. \tag{5.3}$$

The objective of the EA is to maximize the fitness function defined, i.e., maximize the classification performance and minimize the number of instances obtained. In the experiments, we have considered $\alpha = 0.5$.

5.6 Methodology for the Experiments

In this section, we present the methodology followed for the experiments. Section 5.6.1 describes the data sets used, Section 5.6.2 explains the partitions of the data sets that were considered for applying the algorithms, and finally, Section 5.6.3 introduces the parameters associated with the algorithms.

5.6.1 Data Sets

A different group of data sets have been contemplated for each problem.

5.6.1.1 Instance Selection – Prototype Selection

We have evaluated 10 classical data sets used in machine learning for the PS [39] shown in Table 5.1.

Cleveland: This database contains 76 attributes, but all published experiments refer to using a subset of 13 of them. In particular, the Cleveland database is the only one that has been used by machine learning researchers to this date. The "goal" field refers to the presence of heart disease in the patient. It is integer-valued from 0 (no presence) to 4. Experiments with the Cleveland database have concentrated on simply attempting to distinguish presence (values 1, 2, 3, 4) from absence (value 0).

Table 5.1. Data sets for IS-PS.

Data set	Num. instances	Num. features	Num. classes
Cleveland	297	13	2
Glass	214	9	6
Iris	150	4	3
LED24Digit	200	24	10
LED7Digit	500	7	10
Lymphography	148	18	4
Monk	432	6	2
Pima	768	8	2
Wine	178	13	3
Wisconsin	683	9	2

Glass: The study of classification of types of glass was motivated by criminological investigation. At the scene of the crime, the glass left can be used as evidence if it is correctly identified.

Iris: The data set contains three classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other two; the latter are *NOT* linearly separable from each other.

LED24Dig: This simple domain contains 24 Boolean attributes and 10 concepts, the set of decimal digits. Recall that LED displays contain 24 light-emitting diodes -- hence the reason for seven attributes. The problem would be easy if not for the introduction of noise.

LED7Dig: This simple domain contains seven Boolean attributes and 10 concepts, the set of decimal digits. Recall that LED displays contain seven light-emitting diodes -- hence the reason for seven attributes. The problem would be easy if not for the introduction of noise.

Lymphography: This lymphography domain was obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia.

Monk: The MONK's problem was the basis of a first international comparison of learning algorithms. The result of this comparison is summarized in “the MONK's problems.”

Pima: The diagnostic, binary-valued variable investigated is whether the patient shows signs of diabetes according to World Health Organization criteria (i.e., if the two-hour postload plasma glucose was at least 200 mg/dl at any survey examination or if found during routine medical care). The population lives near Phoenix, Arizona, USA.

Wine: These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

Wisconsin: This breast cancer database was obtained from the University of Wisconsin Hospitals, Madison, from Dr. William H. Wolberg.

Table 5.2. Data sets for IS-TSS.

Data set	Num. instances	Num. features	Num. classes
Pen-based recognition	10992	16	10
SatImage	6435	36	6
Thyroid	7200	21	3

5.6.1.2 Instance Selection – Training Set Selection

To adequately study the behavior of the IS algorithm on the TSS, we should consider data sets with a larger number of instances than the data sets in Table 5.1. Therefore, we have chosen three databases that contain more than 6000 individuals, and up to 11,000, which allow an analysis of the scaling up associated with the IS algorithms to be made. They are shown in Table 5.2.

Pen-Based Recognition: A digit database was created by collecting 250 samples from 44 writers. A WACOM PL-100V pressure-sensitive tablet with an integrated LCD display and a cordless stylus were used. The input and display areas are located in the same place. Attached to the serial port of an Intel 486-based PC, it allows us to collect handwriting samples. These writers are asked to write 250 digits in random order inside boxes of 500-by-500 tablet pixel resolution. The raw data that we capture from the tablet consist of integer values between 0 and 500.

SatImage: The database consists of the multispectral values of pixels in 3x3 neighborhoods in a satellite image, and the classification associated with the central pixel in each neighborhood. The aim is to predict this classification, given the multispectral values. In the sample database, the class of a pixel is coded as a number.

Thyroid: The aim is to determine whether a patient referred to the clinic is hypothyroid. Therefore three classes are built: normal (not hypothyroid), hyperfunction, and subnormal functioning.

5.6.2 Partitions

Due to the different strategy followed in IS-PS and IS-TSS, we have taken into account different models of partitions for each one.

5.6.2.1 Instance Selection – Prototype Selection

The sets considered for IS-PS are partitioned using the ten-*fold cross-validation procedure*. Each data set, D, is randomly divided into ten disjoint sets of equal size, $D_1 \ldots D_{10}$. We then conduct ten pairs of training and test sets, (Ti ti), i=1, …,

10. For each pair i, the test set, t_i, is D_i, and the training set, T_i, is the union of all the other D_j, $j \neq i$ (clearly, $D = T_i \cup t_i$ and $T_i \cap t_i = \emptyset$).

Ten trials were run for each data set and IS algorithm. During the ith trial, the algorithm is applied to Ti, and then the resulting reduced set is used by the 1-NN algorithm for classifying the elements of t_i, obtaining a test accuracy.

5.6.2.2 Instance Selection – Training Set Selection

We have followed the stratified approach for IS-TSS shown in Figure 5.3 for carrying out the experiments on the application of the IS algorithms to the TSS. In particular, for each data set, D, two partitions are randomly made, each consisting of two nonoverlapping sets with 50% of the elements: $D = T_{11} \cup T_{12}$ and $D = T_{21} \cup T_{22}$. The IS algorithms are applied to these sets, returning four sets with a reduced number of instances: S_{11}, S_{12}, S_{21}, and S_{22}. Then two different training sets are calculated:

$$S_1 = S_{11} \cup S_{12} \text{ and } S_2 = S_{21} \cup S_{22}. \tag{5.4}$$

Their associated test sets are $s_i = D \backslash S_i$, $i = 1,2$. The training sets are used during the IS process, while the test sets are used to calculate the test accuracy of the model learned. To determine the quality of the training sets obtained, two learning algorithms, the classical 1-NN classifier and the C4.5 [31], were used on these sets.

5.6.3 Algorithms and Parameters

5.6.3.1 Instance Selection – Prototype Selection

We have executed the following classical IS algorithms: CNN, ENN, RENN, MCS, Shrink, and Drop1–3. Moreover, we have carried out experiments with a 1-NN classifier that considers all instances in the training sets.

The parameters used for EAs are:

- GGA considers a population with 10 chromosomes. The crossover rate is 1, and two mutation rates were considered: 0.01 for changing 1 to 0, and 0.001 in the contrary case. This asymmetry in mutation rates is considered to favor the presence in the population of solutions with a few instances, which is a desirable feature. GGA was run during 1000 generations.
- SGA employs these parameters, as well, but considers 10000 offspring evaluations.
- The population size of the CHC algorithm was 10 chromosomes, and it was performed during 1000 generations.
- The parameters associated with PBIL were: $N_{\text{samples}} = 10$, $LR = 0.005$, $P_m = 0.01$, and $Mut\_Shif = 0.01$; 1000 iterations for this algorithm were completed.

All the EAs use the solution representation and fitness function presented in Sections 5.5.1 and 5.5.2. All the algorithms assume $k = 1$.

5.6.3.2 Instance Selection – Training Set Selection

For IS-TSS, we have run two classical IS algorithms: MCS and DROP1. MCS is chosen because it presents the best classification accuracy on test data, and DROP1 is selected because it has the best reduction of training set.

The parameters used for EAs are:

- The population size of GGA is 20 chromosomes. The crossover rate is 1, and two mutation rates were considered: 0.01 for changing 1 to 0, and 0.001 in the contrary case. GGA was run during 500 generations.
- The parameters of SGA are the same but considering 10,000 offspring evaluations.
- The population size of the CHC algorithm was 20 chromosomes, and it was executed during 500 generations.
- The parameters for PBIL were: $N_{\text{samples}} = 20$, $LR = 0.005$, $P_m = 0.01$, and $Mut\_Shif = 0.01$. This algorithm completed 500 iterations.

For IS-TSS, we have increased the population size due to the greater complexity of the search space.

The solution representation and fitness function used by all the EAs are the ones in Sections 5.5.1 and 5.5.2, respectively. All the algorithms use the 1-NN for the fitness function.

5.7 Analysis of the Experiments

5.7.1 Analysis and Results for Prototype Selection

Tables 5.3 and 5.4 show the results obtained by the classical algorithms and the evolutionary IS algorithms, respectively:

- The average test accuracy over the 10 trials is reported for each algorithm on each data set.
- The average reduction percentage from the initial training sets is also reported for each experiment under the column "%."

Furthermore, to observe the level of robustness achieved by all the EAs, we have included in Table 5.4 two columns with the average results of all of them.

Table 5.3. Results for the classical algorithms on PS.

Database	1-NN	%	CNN	%	ENN	%
Cleveland	41.34	0	33.98	34.9	48.83	59.03
Glass	71.65	0	69.29	55.51	69.69	28.09
Iris	96.67	0	94.67	90.37	96.67	4.15
LED24Digit	40.07	0	38.20	24.78	37.66	61.33
LED7Digit	40.41	0	41.00	37.48	36.61	59.73
Lymphography	41.54	0	41.09	38.39	49.41	55.41
Monk	66.44	0	64.82	60.54	64.59	33.44
Pima	67.59	0	61.46	61.72	69.40	31.89
Wine	78.14	0	68.53	64.04	74.74	24.16
Wisconsin	95.46	0	91.80	92.45	96.63	4.38
Average	63.93	0	60.48	56.02	64.42	36.16

Database	RENN	%	MCS	%	Shrink	%
Cleveland	50.59	62.33	42.05	34.08	50.19	51.14
Glass	66.87	31.93	70.50	15.47	68.73	24.82
Iris	96.67	4.15	96.67	2.18	96.67	3.26
LED24Digit	35.52	68.11	37.32	33.61	40.98	52.67
LED7Digit	34.45	72.16	51.18	3.49	36.79	61.82
Lymphography	46.74	59.84	48.04	29.36	47.41	53.38
Monk	66.69	43.44	61.59	7.54	65.97	33.51
Pima	69.52	35.27	69.01	16.16	68.35	27.14
Wine	71.31	27.65	75.29	12.61	73.01	21.41
Wisconsin	96.63	4.52	96.92	2.21	96.19	2.88
Average	63.5	40.94	64.86	15.67	64.43	33.2

Database	Drop1	%	Drop2	%	Drop3	%
Cleveland	48.17	86.87	41.01	21.44	43.44	77.63
Glass	62.72	80.43	66.49	53.90	61.19	71.69
Iris	92.67	97.33	90.00	88.15	93.33	95.28
LED24Digit	36.52	76.33	39.02	21.61	34.15	43.49
LED7Digit	47.57	96.89	52.74	33.38	48.23	71.09
Lymphography	42.30	77.71	46.11	28.68	47.81	52.69
Monk	62.51	80.97	58.36	54.09	55.81	71.77
Pima	67.20	83.68	68.88	49.20	64.71	76.58
Wine	73.56	80.28	74.67	57.06	72.48	74.12
Wisconsin	95.32	98.42	95.90	92.42	94.43	98.21
Average	62.85	85.89	63.32	49.99	61.56	73.26

Table 5.4. Results for the evolutionary IS algorithms on PS.

Database	GGA	%	SGA	%	CHC	%	PBIL	%	Avg	Avg%
Clevel.	49.21	96.00	47.65	94.22	52.90	98.69	51.57	97.61	50.33	96.63
Glass	71.80	89.93	69.98	88.41	69.14	93.15	65.39	92.84	69.08	91.08
Iris	96.00	95.56	95.15	95.84	95.33	96.59	96.00	96.59	95.62	96.15
LED24D.	32.05	88.67	30.08	86.12	31.87	92.05	37.17	90.61	32.79	89.36
LED7D.	64.56	95.58	63.66	94.72	65.39	96.04	62.80	90.18	64.10	94.13
Lymph.	48.33	89.72	48.25	87.84	42.67	94.30	49.40	93.32	47.16	91.30
Monk	64.09	91.18	63.41	91.34	67.37	98.05	62.28	96.32	64.29	94.22
Pima	69.79	94.89	68.01	93.99	73.17	97.80	71.74	95.10	70.68	95.45
Wine	71.96	94.88	69.97	94.83	74.77	96.82	69.74	96.63	71.61	95.79
Wiscon.	97.07	98.93	96.12	96.43	95.91	99.40	96.05	98.32	96.29	98.27
Average	66.49	93.53	65.23	92.37	66.85	96.29	66.21	94.75	66.20	94.24

We wish to point out the following conclusions about the evolutionary IS algorithms for PS:

- We should highlight the high values reached by the EAs for the reduction percentage (around 94%). This is a great advantage with regard to the percentage of the best classical algorithm, DROP3 (73.26%).
- The average behavior of all the EAs is 66.20 for accuracy and 94.24 for reduction percentage. These values are better than the ones for all the classical algorithms.
- Although, in general, the results of all the EAs are very similar, the CHC algorithm arises as the best one.

5.7.2 Analysis and Results for Training Set Selection

As we mentioned in Section 5.6.2, for each data set and IS algorithm executed, two pairs of training selected set and test set are obtained (S_i s_i), I =1, 2. These pairs are then used by two learning algorithms, the 1–NN classifier and the C4.5. Tables 5.5 and 5.7 show the results of these algorithms on the pairs (S_i s_i) obtained from the classical IS algorithms. Tables 5.6 and 5.8 have the same information for the case of the pairs (S_i s_i) obtained from the evolutionary IS algorithms:

- The column "1–NN" has the test accuracy obtained from the classification of the elements in s_i by an 1-NN classifier that uses the set of prototypes S_i.
- The column "C4.5" contains the test accuracy achieved by classifying the elements in s_i by means of the decision tree learned by the C4.5 from S_i.
- The column "%" reports the reduction percentage of S_i with regards to the corresponding complete data set.

Again, for the case of the EAs, three columns were included with the average results for all of these algorithms.

Table 5.5. Results using the S_1 and s_1 sets obtained from the classical algorithms.

	MCS			DROP1		
Database	1-NN	C4.5	%	1-NN	C4.5	%
Pen-based	99.00	98.90	50.22	86.58	99	98.82
Satimage	88.81	95.40	52.37	77.67	97.2	96.48
Thyroid	91.37	99.80	52.36	52.20	99.9	99.9
Average	93.06	98.03	51.65	72.15	98.70	98.40

Table 5.6. Results using the S_1 and s_1 sets obtained from the EAs.

	CHC			PBIL		
Database	1-NN	C4.5	%	1-NN	C4.5	%
Pen-based	98.29	98.90	87.35	83.33	99.10	99.68
Satimage	86.22	97.00	91.48	85.94	96.60	96.76
Thyroid	91.15	99.90	89.31	88.91	99.90	96.38
Average	91.89	98.60	89.38	86.06	98.53	97.61

	CHC			PBIL		
Database	1-NN	C4.5	%	1-NN	C4.5	%
Pen-based	98.29	98.90	87.35	83.33	99.10	99.68
Satimage	86.22	97.00	91.48	85.94	96.60	96.76
Thyroid	91.15	99.90	89.31	88.91	99.90	96.38
Average	91.89	98.60	89.38	86.06	98.53	97.61

	Average		
Database	1-NN	C4.5	%
Pen-based	92.27	99.00	94.38
Satimage	83.74	97.20	95.54
Thyroid	91.02	99.90	95.11
Average	89.01	98.70	95.01

Table 5.7. Results using the S_2 and s_2 sets obtained from the classical algorithms.

	MCS			DROP1		
Database	1-NN	C4.5	%	1-NN	C4.5	%
Pen-based	99.05	95.20	50.16	85.53	99.1	98.83
Satimage	89.64	97.40	52.37	82.18	96.8	96.31
Thyroid	91.77	99.50	51.99	43.59	99.9	99.9
Average	93.49	97.37	51.51	70.43	98.60	98.35

Table 5.8. Results using the S_2 and s_2 sets obtained from the EAs.

	GGA			SGA		
Database	1-NN	C4.5	%	1-NN	C4.5	%
Pen-based	95.60	99.10	96.13	96.12	99.10	94.18
Satimage	85.41	96.90	97.46	86.65	97.10	95.82
Thyroid	90.52	99.90	97.58	88.46	99.90	96.15
Average	90.51	98.63	97.06	90.41	98.70	95.38

	CHC			PBIL		
Database	1-NN	C4.5	%	1-NN	C4.5	%
Pen-based	98.32	99.10	85.70	79.37	99.10	99.75
Satimage	86.48	97.30	91.96	84.88	97.10	96.87
Thyroid	91.36	100.00	88.58	87.64	99.90	96.64
Average	92.05	98.80	88.75	83.96	98.70	97.75

	Average		
Database	1-NN	C4.5	%
Pen-based	92.35	99.10	93.94
Satimage	85.86	97.10	95.53
Thyroid	89.50	99.93	94.74
Average	89.23	98.71	94.74

We want to offer the following conclusions:

- The C4.5 algorithm obtains a very good test accuracy (approximately 98%), for all the IS algorithms. This indicates that the stratified mechanism used in this chapter for the TSS (Fig. 5.3) is a suitable and robust method for finding training sets for this learning algorithm.
- The EAs return (S_1 s_1) and (S_2 s_2) pairs with a better mix of 1–NN and C4.5 test accuracy and reduction percentage than the ones reached with the classical algorithms.
- The best EA with regard to the 1–NN and C4.5 test accuracy is CHC.

5.8 Concluding Remarks

This chapter presented the analysis of the evolutionary IS algorithms and its use for data reduction in KD. An experimental study has been carried out for comparing the results of four EA models against classical IS algorithms on two particular applications, the PS and the TSS. The principal conclusions reached are the following:

- EAs outperform the classical algorithms, offering two main advantages simultaneously, better data reduction percentages and higher classification accuracy.
- The CHC algorithm is particularly appropriate as an IS algorithm.

Furthermore, the stratified model applied to TSS seems adequate for applying EAs to large data sets. It has arisen as a powerful tool for obtaining training sets for the C4.5 algorithm, independent of the IS algorithm used.

Finally, we wish to point out that future works may be directed at studying the behavior of the evolutionary IS algorithms on databases with a large number of instances.

References

[1] Adriaans, P., and Zantinge, D., *Data Mining*, Addison-Wesley, 1996.
[2] Back, T., *Evolutionary Algorithms in Theory and Practice*, Oxford, 1996.
[3] Back, T., Fogel, D., and Michalewicz, Z., *Handbook of Evolutionary Computation*, Oxford University Press, 1997.
[4] Babu, T. R., and Murty, M. N., Comparison of genetic algorithms based prototype selection schemes, *Pattern Recognition*, 34, 523–5, 2001.
[5] Baluja, S., Population-based incremental learning, Technical Report CMU-CS-94-163. Carnegie Mellon University, 1994.
[6] Brighton, H., and Mellish, C., Advances in instance selection for instance-based learning algorithms, *Data Mining and Knowledge Discovery*, 6, 153–72, 2002.
[7] Brodley, C. E., Addressing the selective superiority problem: Automatic algorithm/model class selection, in *Proc. 10th International Machine Learning Conference*, Amherst, MA, 17–24, 1993.
[8] Chapman, P., Clinton, J., Khabaza, T., Reinart, T., and Wirth, R., The CRISP-DM Process Model. www.crisp-dm.org, 1999.
[9] Cover, T. M., and Hart, P. E., Nearest neighbor pattern classification, *IEEE Transactions on Information Theory*, 13, 21–7, 1967.
[10] Dasarathy, B. V., *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*, IEEE Computer Society Press, Los Alamitos, CA, 1991.
[11] Devijver, P. A., and Kittler, J., On the edited nearest neighbor rule, in Proc. 5th Internat. Conf. on Pattern Recognition, 72–80, 1980.
[12] Eshelman, L. J., The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination, in *Foundations of Genetic Algorithms-1*, Rawlins, G. J. E., (ed.), Morgan Kauffman, 265–83, 1991.
[13] Fogel, D. B., *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press: Piscataway, 1995.
[14] Frank, E., and Witten, I. H., Making better use of global discretization, in *Proc. 16th International Conference on Machine Learning*, Bratko, I., and Dzeroski, S., (eds.), Morgan Kaufmann, 115–23, 1999.

[15] Gates, G. W., The reduced nearest neighbor rule, *IEEE Transactions on Information Theory*, 18(3), 431–3, 1972.
[16] Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, New York, 1989.
[17] Hart, P. E., The condensed nearest neighbor rule, *IEEE Transaction on Information Theory*, 14, 515–6, 1968.
[18] Holland, J. H., *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, 1975.
[19] Ishibuchi, H., Nakashima, T., and Nii, M., Genetic-algorithm-based instance and feature selection, in *Instance Selection and Construction for Data Mining*, Kluwer Academic Publishers, 95–112, 2001.
[20] Kibbler, D., and Aha, D. W., Learning representative exemplars of concepts: An initial case study, in *Proc. 4th International Workshop on Machine Learning*,. Irvine, CA, Morgan Kaufmann, 24–30, 1987.
[21] Kuncheva, L. I., Editing for the *k*-nearest neighbors rule by a genetic algorithm, *Pattern recognition Letters*, 16, 809–14, 1995.
[22] Kuncheva, L. I., Fitness functions in editing *k*-NN reference set by genetic algorithms, *Pattern Recognition*, 30(6), 1041–9, 1996.
[23] Kuncheva, L. I., and Jain, L. C., Nearest neighbor classifier: Simultaneous editing and feature selection, *Pattern Recognition Letters*, 20, 1149–56, 1999.
[24] Liu, H., and Motoda, H., *Feature Selection for Knowledge Discovery and Data Mining*, Kluwer Academic Publishers, 1998.
[25] Liu, H., and Motoda, H., *Instance Selection and Construction for Data Mining*, Kluwer Academic Publishers, 2001.
[26] Liu, H., and Motoda, H., Data reduction via instance selection, in *Instance Selection and Construction for Data Mining*, Liu, H., and Motoda, H., (eds.), Kluwer Academic Publishers, 3–20, 2001.
[27] Liu, H., and Motoda, H., On issues of instance selection, *Data Mining and Knowledge Discovery*, 6, 115–30, 2002.
[28] Nakashima, T., and Ishibuchi, H., GA-based approaches for finding the minimum reference set for nearest neighbor classification, in Proc. 1998 IEEE Conf. on Evolutionary Computation, IEEE Service Center, 709–14, 1998.
[29] Pelikan, M., Goldberg, D. E., and Lobo, F., A survey of optimization by building and using probabilistic models, *IlliGAL Report No.99018*, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL, 1999.
[30] Plutowski, M., *Selecting Training Exemplars for Neural Network Learning*, PhD Dissertation, University of California, San Diego, 1994.
[31] Quinlan, J. R., *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA, 1993.
[32] Reeves, C. R., and Bush, D. R., Using genetic algorithms for training data selection in RBF networks, Liu, H., and Motoda, H., (eds.), *Selection and Construction for Data Mining*, Kluwer Academic Publishers, 339–56, 2001.
[33] Reinartz, T., A unifying view in instance selection, *Data Mining and Knowledge Discovery*, 6, 191–210, 2002.

[34] Shanahan, J. G., *Soft Computing for Knowledge Discovery*, Kluwer Academic Publishers, 2000.
[35] Witten, I. H., and Frank, E., *Data Mining*, Morgan Kaufmann Publishers, 2000.
[36] Whitley, D., The GENITOR algorithm and selective pressure: Why rank based allocation of reproductive trials is best, *Proc. 3rd International Conference on GAs*, Schaffer, J. D., (Ed.), Morgan Kaufman, 116–21, 2000.
[37] Whitley, D., Rana, S., Dzubera, J., and Mathias, E., Evaluating evolutionary algorithms, *Artificial Intelligence*, 85, 245–76, 1996.
[38] Wilson, D. L., Asymptotic properties of nearest neighbor rules using edited data, *IEEE Trans. on Systems, Man, and Cybernetic*, 2, 408–21, 1972.
[39] Wilson, D. R., and Martinez, T. R., Reduction techniques for exemplar-based learning algorithms, *Machine Learning*, 38, 257–68, 2000.
[40] Zhang, J., Selecting typical instances in instance-based learning, in *Proc. 9th International Conference on Machine Learning*, Sleeman, D., Edwards, P., (eds.), Morgan Kaufmann, 470–9, 1992.
[41] Zhao, Q., and Higuchi, T., Minimization of nearest neighbor classifiers based on individual evolutionary algorithm, *Pattern Recognition Letters*, 17, 125–31, 1996.

6. Using Cooperative Coevolution for Data Mining of Bayesian Networks

Man Leung Wong,[1] Shing Yan Lee,[2] and Kwong Sak Leung[2]

[1] Department of Information Systems, Lingnan University, Tuen Mun, Hong Kong.
[2] Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong.

Bayesian networks are formal knowledge representation tools that provide reasoning under uncertainty. The applications of Bayesian networks are widespread, including data mining, information retrieval, and various diagnostic systems. Although Bayesian networks are useful, the learning problem, namely to construct a network automatically from data, remains a difficult problem. Recently, some researchers have adopted evolutionary computation for learning. However, the drawback is that the approach is slow. In this chapter, we propose a hybrid framework for Bayesian network learning. By combining the merits of two different learning approaches, we expect an improvement in learning speed. In brief, the new learning algorithm consists of two phases: the conditional independence (CI) test phase and the search phase. In the CI test phase, we conduct dependency analysis, which helps to reduce the search space. In the search phase, we perform model searching using an evolutionary approach, called cooperative coevolution. When comparing our new algorithm with an existing algorithm, we find that our algorithm performs faster and is more accurate in many cases.

6.1 Introduction

Bayesian networks, or Bayesian belief networks, are popular in dealing with uncertainty for designing intelligent systems. Basically, a Bayesian network is a graph that depicts conditional independence among random variables in the domain. In Fig. 6.1, a Bayesian network example is shown. By definition, a Bayesian network also encodes the joint probability distribution of the random variables. With a network at hand, we can perform probabilistic inference for various uses. For instance, we can predict the most likely outcome of certain variables based on the observation of others. In light of this, Bayesian networks are widely used in diagnostic systems. For example, in medical diagnosis, there is MUNIN, which is used for diagnosing diseases in muscles and nerves, and PATHFINDER, which is used for diagnosing lymph node disease [6.1]. They are also used in information retrieval [6.2] and printer troubleshooting [6.3].

The literature on Bayesian networks concentrates on two major issues: the learning problem and the inference problem. Here, we focus our attention on

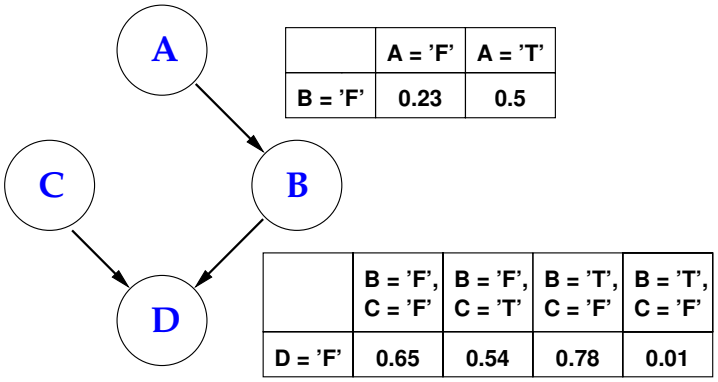


Fig. 6.1. A Bayesian network example.

the learning problem, which is an intractable problem. In the learning problem, the objective is to construct a Bayesian network that best describes a given set of observations about the domain. There are two major approaches to tackle the problem: the *dependency analysis* and the *search-and-scoring* approaches [6.4]. In short, the dependency analysis approach constructs a network by discovering the dependency information from data. The search-and-scoring approach, on the other hand, searches for the optimal network according to a metric that evaluates the goodness of a candidate network with reference to the data. While the two approaches try to learn Bayesian networks differently, they both suffer from their respective drawbacks and shortcomings. For the former, a straightforward implementation would require an exponential number of tests [6.5]. Worse still, some test results may be unreliable [6.5]. For the latter, the search space is often huge, and it is difficult to find a good solution.

In this chapter, we propose a hybrid learning framework that combines the merits of both approaches. The hybrid framework consists of two phases: the conditional independence test (CI test) phase and the search phase. In essence, the main idea is that we exploit the information discovered by dependency analysis (CI test phase) to reduce the search space in the search phase. To tackle the search problem, the idea of cooperative coevolution [6.6], [6.7], [6.8], which is a modular decomposition evolutionary search approach, is employed. Our new approach for learning Bayesian networks is called the cooperative coevolution genetic algorithm (CCGA).

We compare CCGA with another existing learning algorithm, MDLEP [6.9]; CCGA executes much faster. Moreover, CCGA usually performs better in discovering the original structure that generates the training data.

This chapter is organized as follows. In Section 6.2, we present the backgrounds of Bayesian networks, the MDL metric, and cooperative coevolution.

Different methods of applying evolutionary computation to learn Bayesian networks are presented in Section 6.3. In Section 6.4, we describe our algorithm in detail. In Section 6.5, we present a comparison between the new algorithm and another existing algorithm (MDLEP). We conclude the chapter with Section 6.6.

6.2 Background

6.2.1 Bayesian Network Learning

It was not until Pearl's work [6.10] that Bayesian networks were given a solid foundation. Basically, Bayesian networks are directed acyclic graphs (DAG), which describe conditional independency relations. Each node in the graph corresponds to a discrete random variable in the domain, U. Each edge designates a parent-and-child relation. For a given node $X \in U$, all of its parents constitute the parent set of X, which is denoted by Π_X. In addition to the graphical structure, there are conditional probability tables (CPT) specifying the conditional probability distribution of each domain variable given its parent set.

Because Bayesian networks are founded on the notion of conditional independency, it is necessary to give a brief description of the subject. A conditional independence relation is a three-place relationship among distinct subsets of variables X, Y, and Z, denoted by $I(X, Z, Y)$. Equivalently, we say X and Y are conditionally independent given the *conditioning set*, Z. Formally speaking, the following relationship holds [6.10]:

$$P(x, y \mid z) = P(x \mid z) \quad \text{whenever} \quad P(y, z) > 0, \tag{6.1}$$

where x, y, and z are any instantiations of the sets X, Y, and Z, respectively, and P is the probability distribution. A conditional independence relation is characterized by its order, which is simply the size of the conditioning set Z.

By definition, a Bayesian network encodes the joint probability distribution of the domain variables $U = \{N_1, \ldots, N_n\}$:

$$P(N_1, \ldots N_n) = \prod_i P(N_i \mid \Pi_{N_i}). \tag{6.2}$$

The Dependency Analysis Approach. As mentioned before, researchers treat the network learning problem in two very different ways. The first approach, called the dependency analysis approach, takes the view that Bayesian networks depict conditional independence relations among the variables. Hence, the approach relies on discovering conditional independence relations from the data for network construction. Work belonging to this category include [6.5], [6.11], and [6.4]. Typically, the existence of a *perfect map*

is presumed for a given distribution P [6.5]. In other words, it is assumed that there exists a Bayesian network, G, that captures all the conditional independence relations implied by P. Consequently, this suggests a general learning methodology: Construct a network G by testing the validity of any independence assertions $I(X, Z, Y)$. In practice, we can use what is collectively called the conditional independence test (CI test). If the statement $I(X, Z, Y)$ is supported by the data, it follows that X should be d-separated [6.10] with Y by Z in G; otherwise, X is not d-separated with Y by Z.

As a digression from the ongoing discussion, we give a brief description of the CI test. A common approach is to use a hypothesis testing procedure discussed in the statistical literature [6.5], [6.12], [6.13]. To begin, the conditional independence assertion (i.e., $I(X, Z, Y)$) is modeled as the *null hypothesis*. Suppose we use the likelihood-ratio χ^2 test; the χ^2 statistics is calculated by:

$$G^2 = -2 \sum \text{observed} * \log(\text{expected}/\text{observed}). \tag{6.3}$$

Simply put, the statistic calculate the discrepancies between the real occurrence, *observed*, and the expected count followed from the hypothesis, *expected*, over every distinct event. In our case, because $I(X, Z, Y)$ implies

$$\begin{aligned} P(x, y, z) &= P(x \mid y, z)\, P(y, z) \\ &= P(x \mid y)\, P(y, z) \qquad \text{(by Eq. 6.1)}, \end{aligned}$$

the statistics are computed by:

$$G^2 = -2 \sum_{x,y,z} P(x, y, z) \log \frac{P(x, y, z)}{P(y, z) P(x \mid z)}. \tag{6.4}$$

Suppose the number of possible instantiations of the variables X, Y, and Z are, respectively v_X, v_Y, and v_Z; G^2 follows a χ^2 distribution with $(v_X - 1) \times (v_Y - 1) \times v_Z$ degrees of freedom. Checking our computed G^2 against the distribution, we obtain the p-value, which is "the smallest level of significance for which the data lead to the rejection of the null hypothesis" [6.14]. If the p-value is less than a predefined *cutoff value* α, the test shows strong evidence to reject the hypothesis; otherwise, the hypothesis cannot be rejected.

Take the SGS algorithm [6.5] as an illustration. The algorithm begins with a completely connected undirected graph. In other words, dependence between every pair of variables is assumed. Then CI tests between all pairs of connected nodes are conducted. When two nodes X and Y are found to be conditionally independent given Z, the undirected edge between them is removed so that $I(X, Z, Y)$ is not violated. When no more edges can be removed, the undirected edges in the graph are oriented according to some rules that conform to the conditional independence relations discovered previously. This produces the final Bayesian network.

In general, there are three problems typical to the dependency analysis approach. First, it is difficult to determine whether two nodes are dependent.

Quoting from [6.5]: "In general, two variables X and Y may be conditionally dependent given a set Z while independent on the subset or superset of Z." In the worst case, like in SGS, every possible combination of the conditioning set should be examined, which would require an exponential number of tests. Second, results from CI tests may not be reliable for high-order CI tests (when the size of the conditioning set is high) [6.5], [6.15]. Hence, for algorithms that require high-order CI tests, the results may be inaccurate. Third, because a network is constructed in a step-by-step manner, the construction algorithm may be *unstable* in the sense that an earlier mistake during construction is consequential [6.5], [6.16]. Moreover, this suggests that the order of testing the CI relations is important, which will be a concern when one pursues optimal performance.

The Search-and-Scoring Approach. The second approach is called the search-and-scoring approach. Recall that a Bayesian network encodes a joint probability distribution (Eq. 6.2); we can derive a measure to assess the goodness of such encoding. For instance, such a measure could be derived from Bayesian statistics, information theory, or the minimum description length (MDL) principle [6.17]. Though their theoretical foundations are different, some studies [6.18], [6.19] show that different metrics are asymptotically equivalent under certain conditions.

Because we employ the MDL metric [6.20] in our work, we take it as an example for illustration. Basically, the metric is derived from information theory and incorporates the idea of the minimum description length principle. The metric has two components: the network description length and the data description length. An optimal network is the one that minimizes both simultaneously.

Formally, let $U = \{N_1, \ldots, N_n\}$ be the set of discrete variables, Π_{N_i} be the parent set of a node N_i in the candidate network, and v_i be the number of possible states of the variable N_i. The network description length is given by

$$\sum_{i=1}^{n} \left[|\Pi_{N_i}| \log_2(n) + d(v_i - 1) \prod_{N_j \in \Pi_{N_i}} v_j \right],$$

where d is a constant denoting the number of bits used to store a numerical value. Intuitively, the network description length represents the structural complexity of the network, which is evaluated by the number of bits required to encode the graphical structure and store the conditional probability table at each node.

The data description length is given by

$$\sum_{i=1}^{n} \sum_{N_i,\, \Pi_{N_i}} M(N_i, \Pi_{N_i}) \log_2 \frac{M(\Pi_{N_i})}{M(N_i, \Pi_{N_i})},$$

where $M(.)$ is the count of the particular instantiation in the data set. In essence, the data description length evaluates the proximity of the distributions implied by the data and the candidate network, which is a measure of the accuracy of the candidate network.

Because the MDL metric is simply the sum of the two description lengths, it puts a balance between model complexity and model accuracy. In other words, the optimal network, with regard to the metric, should be simple and accurately represent the joint distribution.

As a property common to other metrics, the MDL metric is node-decomposable and could be written as in Eq. (6.5). One can observe that the score is simply the sum of the independent evaluation on the parent set, Π_{N_i}, of every node N_i in the domain U:

$$\mathrm{MDL}(G) = \sum_{N_i \in U} \mathrm{MDL}(N_i, \Pi_{N_i}). \tag{6.5}$$

With the defined metric, the network learning problem can be formulated as a search problem. The objective is to search for the network structure that has the optimal score. However, the problem is difficult as the search space, which contains all possible network structures, is huge. Chickering et al. proved that the search problem is NP-hard with the use of a particular metric [6.21]. Some research, therefore, resorts to greedy search heuristics [6.22], [6.20]. However, the drawback of these approaches is that suboptimal solutions may be obtained. Some others use systematic and exhaustive search, like branch-and-bound [6.23], to find the optimal solution. In the worst case, the time consumed would be considerable. Recently, some researchers attempt [6.24], [6.9] to use evolutionary computation to tackle the problem.

6.2.2 Evolutionary Computation

Evolutionary computation is a general stochastic search methodology. The principal idea is borrowed from evolution mechanisms proposed by Charles Darwin. Evolutionary computation is becoming popular as it often gives satisfactory results for various optimization problems in different areas. For example, it is applied in data mining, image processing, pattern recognition, and signal processing [6.25], [6.26], [6.27], [6.28], [6.29], [6.30].

In essence, evolutionary computation is a group search algorithm with guidance. In Fig. 6.2, we show the typical steps during searching. A candidate solution in the search space is called a *chromosome*. A chromosome consists of a number of *genes*, which correspond to the elements constituting a solution. At the beginning, a pool of chromosomes, also called the *population*, is created randomly. As such, a number of search points are maintained. For each generated individual, the fitness value, which stands for the quality of the candidate solution it encodes, is computed according to a predefined *fitness function*. In subsequent iterations, or *generations*, new chromosomes

1. Set t, the generation count, to 0.
2. Create an initial population, Pop(t), randomly.
3. While the termination criterion is not matched,
   - select individuals for reproduction according to their fitness values.
   - apply genetic operators to produce offspring.
   - evaluate the fitness values of the offspring.
   - replace members in Pop(t) with offspring, which gives Pop($t + 1$).
   - increment t by 1.
4. Return the best-so-far individual as the solution.

Fig. 6.2. Procedures of evolutionary computation.

(the *offspring*) are created by genetic operators that alter the genetic composition of the parental chromosomes. Intuitively, this could be regarded as the exploration of the search space by exploiting previous search results. Then selection comes into play, where the weaker ones will vanish and stronger ones will have a higher chance to survive into the next generation. This process is repeated until a termination criterion is satisfied. Because better ones will have a better chance of surviving, it is expected that a good, or near optimal, solution can be obtained ultimately.

Cooperative Coevolution. Coevolution is the evolution of different species in the same environment, where the interactions among them affect the genetic composition. There are two kinds of coevolution: competitive and cooperative. In nature, the kind of coevolution we often see is competitive coevolution. For instance, the "arm race" between two species is a good demonstration of competitive coevolution. In cooperative coevolution, the natural selection pressure will prefer individuals that could have good collaboration with other species.

Based on the work of Potter and DeJong [6.6], [6.7], [6.8], cooperative coevolution represents a problem breakdown methodology. A problem instance is divided into a number of subcomponents that correspond to different species. The analogy is that once species (i.e., solutions to subcomponents) can cooperate among themselves, the collaboration (i.e., the assembled solution) will be a good solution.

In each species population, evolutionary search is conducted separately. During fitness evaluation, an individual is assigned a fitness value so that cooperation is promoted. To achieve this, a collaborative structure $\mathcal{S}$ is first assembled from representatives of each species population. Note that $\mathcal{S}$ now is a complete solution to the original problem. When an individual is subject to fitness evaluation, it replaces its representatives in $\mathcal{S}$ and forms $\mathcal{S}'$. As such, the individual is assigned with the fitness of $\mathcal{S}'$, which reflects, to a certain degree, how well it cooperates with individuals in other species. Figure 6.3 shows the cooperative coevolution algorithm.

By using cooperative coevolution, a hard and complex problem can be handled in a systematic and efficient manner. For example, cooperative co-

1. Set t, the generation count, to 0.
2. For each species k,
   - create an initial population, $\mathsf{Pop}_k(t)$, randomly.
3. For each species k,
   - evaluate the fitness of individuals in $\mathsf{Pop}_k(t)$.
4. Compose the collaborative structure $\mathcal{S}$ by combining the best individual from each species.
5. While the termination criterion is not matched:
   - For each species k,
     - evaluate the fitness values of individuals in $\mathsf{Pop}_k(t)$ with respect to the collaborative structure $\mathcal{S}$.
     - select individuals for reproduction according to their fitness values.
     - apply genetic operators to produce offspring.
     - evaluate the fitness values of the offspring with respect to the collaborative structure $\mathcal{S}$.
     - replace members in $\mathsf{Pop}_k(t)$ with offspring, which gives the new population $\mathsf{Pop}_k(t+1)$.
   - Update $\mathcal{S}$.

Fig. 6.3. Cooperative coevolution algorithm.

evolution is applied in learning neural networks [6.6] and in learning sequential decision rules [6.7].

6.3 Learning Using Evolutionary Computation

Recently, a few attempts [6.24], [6.9] were made that apply evolutionary computation to tackle the problem of learning Bayesian networks using the search-and-scoring approach. In [6.24], genetic algorithms (GAs) are used, but [6.9] uses evolutionary programming (EP).

6.3.1 Using GA

Larrañaga et al. [6.24] proposed using genetic algorithms [6.30], [6.31] to search for the optimal Bayesian network structure. In their research, the network structure (composed of n nodes) is represented by an $n \times n$ connectivity matrix C which is, in effect, the transpose of the adjacency matrix. Each element C_{ij} in the matrix is defined as:

$$C_{ij} = \begin{cases} 1, & \text{if node } j \text{ is a parent of node } i \\ 0, & \text{otherwise.} \end{cases}$$

With this representation, the ith row in the matrix encodes the parent set of node N_i (i.e., Π_{N_i}). An illustration is given in Fig. 6.4.
By flattening the matrix, the bit-string representation is obtained:

$$C_{11}C_{21}C_{31}\ldots C_{n1}C_{21}C_{22}\ldots C_{nn}.$$

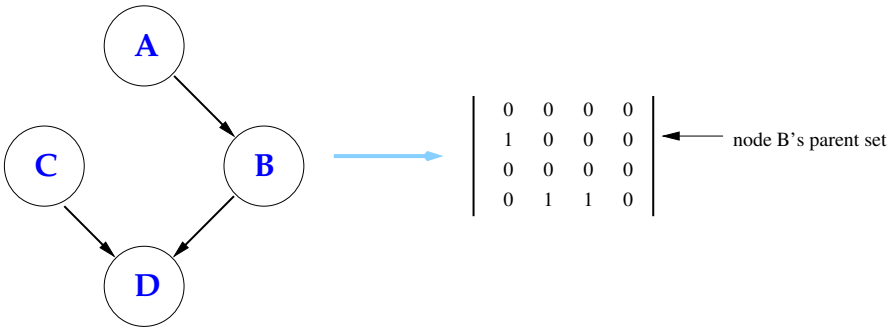


Fig. 6.4. Matrix representation of a DAG.

A traditional GA (with one-point crossover and mutation) is applied to search for the optimal solution represented in the bit-string representation. For the fitness function, they adopted the Bayesian score (called the BD score in [6.21]), which was used in the K2 algorithm [6.22]. Note that because the genetic operators can generate illegal offspring structures (i.e., networks that are not DAG), cycle repairing is needed after an offspring is produced.[3]

Because it is rare to have a densely connected network in real-world problems, they imposed a limit on the number of parents a node could have in its implementation.[4] Even though such a restriction greatly reduces the possible search space, the problem is still NP-hard, as suggested by the work of Höffgen [6.32].

They conducted a number of experiments to test the GA approach with different implementations under different parameter settings. Based on the results, several recommendations regarding the choice of implementation and parameters are made.

6.3.2 Using EP

Recently, Wong et al. [6.9] used evolutionary programming to tackle the learning problem. Because they used the minimum description length metric [6.20] to evaluate fitness, they called their approach MDLEP.

EP is different from GAs mainly in the format of solution representation and the genetic operators used [6.33], [6.26]. Unlike the restricted use of string in GAs, EP does not have any restriction in solution representation. An individual in MDLEP is simply the connectivity matrix of the network.

[3] In a later work, they considered the case that a node ordering is given. With a node ordering, the genetic operators become closed operators, which means no repairing is required.

[4] The same limit is imposed in MDLEP and all of our algorithms.

1. Set t, the generation count, to 0.
2. Create an initial population, Pop(t) of m random DAGs (m is the population size).
3. Each DAG in the population is evaluated using the MDL metric.
4. While t is less than the maximum number of generations,
   - each DAG in Pop(t) produces one offspring by performing a number of mutation operations. If there are cycles, a randomly picked edge in each cycle is removed.
   - the DAGs in Pop(t) and all new offspring are stored in the intermediate population Pop$'(t)$. The size of Pop$'(t)$ is $2 \times m$.
   - conduct a number of pairwise competitions over all DAGs in Pop$'(t)$. For each G_i in the population, q other individuals are selected. Then the fitness of G_i and the q individuals are compared. The score of G_i is the number of individuals (out of q) that have lower fitness than G_i.
   - select the m highest-score individuals from Pop$'(t)$ with ties broken randomly. The individuals are stored in Pop($t+1$).
   - increment t by 1.
5. Return the individual that has the lowest MDL metric in any generation of a run as the output of the algorithm.

Fig. 6.5. The MDLEP algorithm.

Furthermore, there is no crossover operation in EP, and the only operation is mutation. An outline of the MDLEP algorithm is given in Fig. 6.5.

In essence, MDLEP uses four mutation operators that include simple, reversion, move, and knowledge-guided mutations. The simple mutation operator randomly picks an edge and either adds the edge to (when it is absent) or removes it from (when it is already present) the network. The reverse mutation operator randomly picks an edge from the network and reverses its direction. The move mutation operator modifies the parent set of a node by replacing one of the parents with a nonparent. The knowledge-guided mutation operator is similar to simple mutation except that an edge is selected with certain guidance. Briefly, each edge, $X \rightarrow Y$, is weighted by evaluating the MDL score of node Y having only X as its parent. These scores are computed and stored at the beginning. When the knowledge-guided mutation operator determines that an existing edge should be removed, it retrieves the stored MDL scores of all edges in the network and those edges with higher scores are deleted with higher probabilities. On the other hand, if the knowledge-guided mutation operator decides to add an edge to the network, it gets the stored MDL scores of the edges that are absent and those edges with lower scores will have higher probabilities of being added.

In their experiments, they tested their algorithm with data sets generated from two benchmark networks, ALARM and PRINTD. They compared their algorithm with the GA approach using the MDL metric, and they found that MDLEP performs better in many aspects. In general, those networks generated from MDLEP have smaller structural differences (in comparison with the original network) and smaller MDL scores. In addition, MDLEP is also faster, as it requires fewer generations to converge and generates less invalid structures.

6.3.3 Criticism of the Previous Approaches

As reported in Wong et al.'s work, the EP formulation seems to have a clear advantage over the GA one. To account for this, we conjecture that the performance gain is largely due to the different choice of genetic operators in producing the offspring, which, in effect, influences the exploration of search space. On the one hand, the success of EP readily suggests that sheer mutations, which correspond to adding or removing an edge or the combination of the two, are good enough for generating new search points. On the other hand, the crossover operation, which plays an important role in GAs, seems to be ineffective. The reason is not difficult to understand because the one-point crossover, in our case, effectively recombines two graphs arbitrarily. In most cases, this could result in an invalid structure. In this regard, such recombination would seem insignificant and offspring do not properly *inherit*, which is supposedly the merit of the crossover operator. Although the intention to exchange information among the population is good, the traditional one-point crossover fails to achieve the purpose.

Despite the EP approach performs better, we note that it often requires a long execution time. For instance, to learn a network with 37 nodes from a given data set of 10,000 cases, MDLEP needs about an hour to find the solution,[5] which is intolerable for practical use. At closer inspection, we find that a major cause of its long execution time is that there are much more worse offspring (comparing an offspring with its parent) produced than better offspring on average. From our experience, if the population size is 50, we would have, on average, fewer than five better offspring produced in each generation. This implies that most of the mutation operations generate inferior network structures. Hence, we conjecture that MDLEP is not efficient enough in finding good solutions.

6.4 Proposed Algorithm

It is straightforward to combine the two approaches for various purposes. In the literature, there are several attempts that show different realizations of the idea.[6] We propose a new hybrid framework that targets improving the learning efficiency. In essence, information from dependency analysis is exploited in the search-and-scoring process so that the searching will be more efficient.

[5] We use 5000 generations as the termination criterion.

[6] For instance, the CB algorithm [6.34] employs a search-and-score approach (i.e., K2), which takes as input a node ordering given by the network constructed by a dependency analysis approach. In another scenario, the BENEDICT algorithm [6.35] uses a metric definition that reflects the discrepancy between the independence assertions implied by the network and the data.

6.4.1 A Hybrid Framework

In dependency analysis, it is assumed that a *perfect map* exists for a given data set D. In particular, we assume the existence of a Bayesian network G, which depicts every conditional independence relation as implied by D (i.e., $I(X, Z, Y)$) through d-separation. To check the validity of a conditional independence assertion $I(X, Z, Y)$ of any two nodes X, Y and a conditioning set Z, CI tests are often used. In the simplest sense, if the assertion $I(X, Z, Y)$ is valid, X and Y cannot be connected because this will violate that X and Y are d-separated. In other words, neither the edge $X \rightarrow Y$ nor the edge $X \leftarrow Y$ will present in the resultant network G. In the SGS algorithm, the same rationale is used to construct a Bayesian network in the initial steps, where X—Y is removed from an undirected connected graph for each verified assertion $I(X, Z, Y)$.

With such observations, we formulate a general hybrid framework for Bayesian network learning that consists of two phases. In the first phase, we conduct low-order CI tests so we know which edge can be removed. Note that we limit ourselves to use only low-order CI tests because their results are more reliable than higher order CI tests and the time complexity is bounded. In the second phase, we use a search-and-score approach with the knowledge obtained previously. In particular, we refine the search space by excluding networks that contain the edges $X \rightarrow Y$ and $X \leftarrow Y$ for every verified assertion $I(X, Z, Y)$. Consequently, the search space is reduced that would imply a speedup for the search process because we would not waste time adding (or removing) potentially *wrong* edges. The proposed framework is general in the sense that it does not necessitate a particular testing procedure for the CI test phase or a particular search method in the search phase.

In our work, we propose using cooperative coevolution for searching because the problem contains certain characteristics that would benefit a modular decomposition technique. In Fig. 6.6, we provide an outline of the algorithm. Because we use cooperative coevolution and genetic algorithms, we call our new algorithm CCGA for short.

6.4.2 CI Test Phase

Initially, we let the possible parent set of each node contain all other nodes. Using the hybrid approach discussed before, we attempt to reduce the size of the parent set of each node by discovering low-order CI relations. For example, if the node X is found to be conditionally independent of node Y in a test, X will be removed from Y's parent set and vice versa. Alternatively, it could be viewed as though the edges $X \leftarrow Y$ and $X \rightarrow Y$ are both excluded for further consideration. Because higher-order CI tests may be unreliable, we only use order-0 and order-1 tests to discern possible conditional independence relations.

- For each node, initialize the possible parent set to contain every other node.
- CI Test Phase:
  - Perform CI test (up to order- 1) between all pairs of nodes.
  - If the nodes are found to be conditionally independent, remove each from its possible parent sets.
- Cooperative Coevolution Search Phase:
  1. Set t, the generation count, to 0.
  2. For each species population,
     - randomly initialize each chromosome in accordance with the possible parent set.
     - evaluate the fitness of each chromosome.
  3. Compose $\mathcal{S}$ by combining the best chromosome from each species population.
  4. Pass the constraints from $\mathcal{S}$ to each species population.
  5. While t is less than the maximum number of generations,
     - inside each species population,
       - temporarily change the possible parent set with the given constraints.
       - perform selection.
       - evolve new offspring using crossover and mutation.
       - evaluate the fitness of each chromosome.
     - Update $\mathcal{S}$.
     - Produce a node ordering from $\mathcal{S}$ and pass the constraints to each species population.
     - Update the best-so-far structure.

Fig. 6.6. The CCGA algorithm.

In our implementation, we use the likelihood-ratio χ^2 test for testing. For a given assertion $I(X,\ Z,\ Y)$, a p-value is returned from the test (see Section 6.2.1 for details). If the p-value is greater than a predefined cutoff value α, the assertion cannot be rejected and we assume $I(X,\ Z,\ Y)$ is valid.

Suppose there are n variables. For a given pair of variables, we need to, in the worst case, conduct the order-0 test (i.e., $I(X, \Phi, Y)$) and all order-1 tests (i.e., test $I(X, Z, Y)$ for every $Z \in U \setminus \{X, Y\}$). Hence, the overall complexity of the CI test phase is bounded by $O(n^3)$ test.

Although CI tests are very useful, incorrect results could be detrimental. In particular, if a crucial edge (which appears in the optimal network) is excluded due to our findings in the CI test phase, it is impossible to obtain the optimal solution in the subsequent search phase. In our implementation, we choose a *moderate* α-value to lessen the reliance on the test results.

6.4.3 Cooperative Coevolution Search Phase

As mentioned before, cooperative coevolution is a kind of problem breakdown. In our case, we divide the network learning problem of n variables into n subproblems, the goal of which is to find the "optimal" parent set for each node. Alternatively, it could be viewed as though we divide the matrix representation into rows as illustrated in Fig. 6.7. Consequently, each candidate solution to the subproblems is represented as a bit-string, which readily

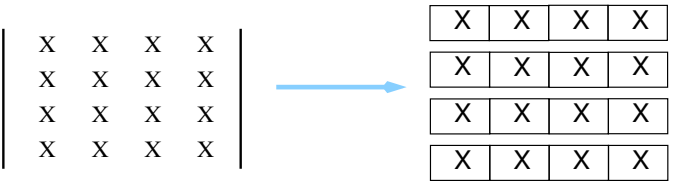


Fig. 6.7. Decomposition of the matrix representation into rows.

suggests the use of genetic algorithms for solving the problems. Following the convention, we call the search population of each subproblem *a species population*. Suppose there are n variables: there will be n different species populations. Inside each population, we use a simple GA to search for the optimal solution.

Although such a problem breakdown seems plausible, it is required that the composite network must be acyclic. Obviously, illegal solutions could be avoided only if each species population has the knowledge of others and works cooperatively to prevent cycle formation. To realize this idea, we propose an approach that uses the topological ordering of a graph as guidance. In the following sections, we discuss our algorithm in detail.

A Feedback Mechanism. Noting that every legal network (i.e., an acyclic graph) conforms to a topological ordering, it follows that we could use an ordering as constraints for each species population to avoid cycle formation. In particular, the possible parent set of a node, which defines the search space of the corresponding species population, could only consist of nodes preceding it in the given ordering. Consequently, a composite of the solutions from the species populations will be acyclic (as it conforms to the given ordering). Alternatively, it could be viewed as if we are to search the optimal network for a given ordering.

Using this idea, we propose a feedback mechanism. Essentially, we use the node ordering implied by the collaborative structure, $\mathcal{S}$, to produce constraints for each species population such that each candidate solution conforms to the ordering. Next, we update $\mathcal{S}$ with results from the species populations and start another cycle. We show this idea in Fig. 6.8.

Nevertheless, there is a problem in the feedback mechanism, namely, $\mathcal{S}$ will conform to the same ordering for whatever update is made. Eventually, this will drive the search process to return the optimal network for an initial ordering, which is fine only if the ordering is optimal.

To tackle this problem, we therefore use $\mathcal{S}$ only to *approximate* an ordering. In particular, every directed edge $X \rightarrow Y$ in $\mathcal{S}$ is associated with a certain degree of belief[7] that specifies the likelihood of finding the edge in the

[7] The value of the degree of belief is selected randomly from a uniform distribution between 0.0 and 1.0.

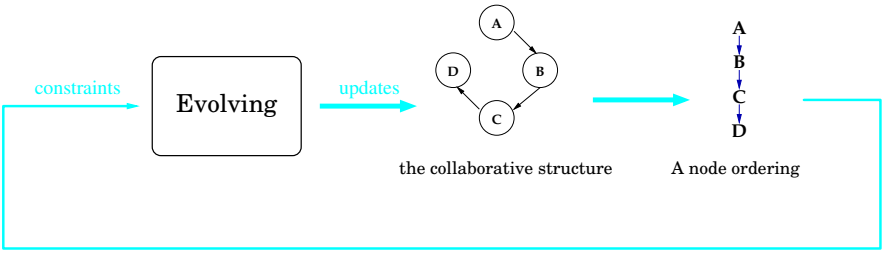


Fig. 6.8. The feedback mechanism.

optimal structure. If our degree of belief is less than a fixed threshold, the *belief factor*, it casts doubt on the correctness of the edge. Hence, we allow the presence of Y in X's possible parent set, which is otherwise forbidden. As a result, a new $\mathcal{S}$ could exhibit an ordering that is different from the original. However, the drawback is that we should watch out for possible cycles formation.

Initialization. In the beginning, the chromosomes in the species populations are randomly initialized. After the populations are initialized, we assemble $\mathcal{S}$ using the best individual from each population. Note that in this way, $\mathcal{S}$ will probably contain cycle(s). Next, for each node N_i in the network, we create a new network $\mathcal{S}'$ that copies $\mathcal{S}$ except that N_i is a root node in $\mathcal{S}'$ (i.e., its parent set is empty). Then the network $\mathcal{S}'$ is repaired for cycles. Using $\mathcal{S}'$ as a reference, we produce constraints on the species population of node N_i such that we assure each candidate solution, when substituted to $\mathcal{S}'$, will create a network that is still acyclic.

Searching Inside the Species Populations. For every species population, the search space is equivalent to the possible parent set of the corresponding node. As mentioned earlier, the possible parent set of a node is subject to different changes during the course of searching. Consequently, the corresponding search process of the node faces both *permanent* and *temporary* constraints. For the permanent constraints, we refer to the reduction of the possible parent set due to the result from CI test. For the temporary constraints, we refer to the changes due to the aforementioned ordering implied by $\mathcal{S}$. As a result of these constraints, the length of the bit-string and the mapping (i.e., which bit corresponds to which parent) are varying.[8]

With the varying bit-string representation, a simple GA with crossover and mutation is used to create a new population. However, instead of using

[8] In the special case when the size of a possible parent set of a node is too small, it would clearly be unwise to search for the optimal combination using genetic operators. Hence, for such cases, we simply list all the possibilities with the entire population, and we inhibit any genetic operation to be performed on the population.

one-point or two-point crossover, we devise a modified crossover operator. Because we have a limit k on the size of the parent set, many of the bit-string are 0s. As an illustration, suppose that the number of all possible parents of a node is 30 and that k equals 5, the bit-string will contain many 0s and a few 1s. As a result, one-point or two-point crossover will be likely to exchange segments of 0s and create nothing new.

To circumvent this problem, we use an approach similar to uniform crossover [6.36]. For the two bit-strings that take part in crossover, we create two different masks for each of them. As an illustration, Fig. 6.9 shows the mask defined and the bit-strings. Here, suppose the number of all possible parents of a node is six and hence the bit-strings are six bits long. A mask of equal length is created for each bit-string. In a position where the original string is 0, the mask is undefined. In a position where the original string is 1, the mask has a value of either 1 or 0. Consider the upper bit string in Fig. 6.9, it has 1s only at the second, fourth, and sixth bits. Hence, the mask value is undefined for the first, third, and fifth bits. If the mask value is 0, the corresponding bit is copied to its offspring. Otherwise, if the mask value is 1, the corresponding bit is copied to the offspring of the other partner. Referring to Fig. 6.9, the upper mask has the value 1 in the fourth bit position, thus the corresponding bit is copied to the offspring of the other partner. The action is indicated by the arrow in the figure.

With this crossover operator, we hope to have a uniform combination of two given parent sets. From our experience, this modified crossover operator indeed improves over the two-point crossover operator.

Update of $\mathcal{S}$. After the new population is created and evaluated at each species population, an individual that has a better score than its correspondent in $\mathcal{S}$ will be used to update $\mathcal{S}$. If a better parent set is found for node N_i, then it will replace the parent set of node N_i in $\mathcal{S}$. However, due to the relaxation of the ordering constraint, such substitution may create an edge

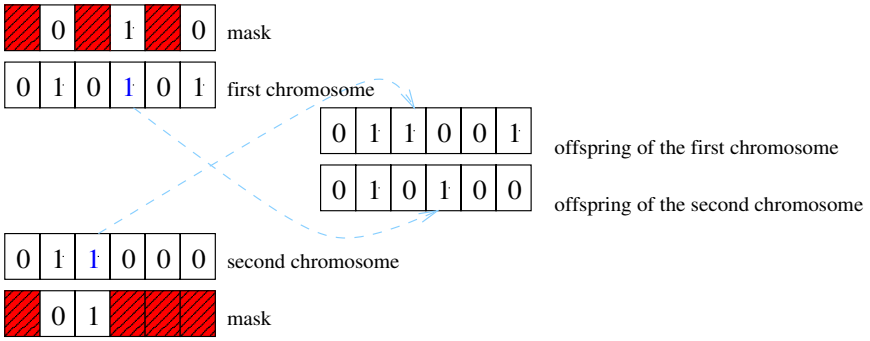


Fig. 6.9. The modified crossover operator.

conflict such that $X \leftarrow Y$ and $X \rightarrow Y$ coexist in $\mathcal{S}$.[9] Hence, we differentiate those edge additions that will create a conflict from those that will not. For those that are not in conflict, they are incorporated directly to $\mathcal{S}$.

To determine which edge ($X \leftarrow Y$ and $X \rightarrow Y$) should be kept is equivalent to determining the proper orientation of the undirected edge X—Y. For this, we use the approach: When we know more about the rest of the network, we would know how to orient the edge. Simply put, with the remaining part of the network known and fixed, we try all possibilities of the unoriented part. Finally, the best configuration is incorporated into $\mathcal{S}$.

For the set of conflicting edges, *related* ones are first grouped together. By related edges, we mean edges that share a common set of nodes, W. Next, different configurations of orientating the group of related edges are tried and evaluated. Because the MDL metric is node-decomposable, it suffices to evaluate the total MDL score of W. Note that although the present configuration is tried, the parent set of each node contains every other known parents. Because each conflicting edge has two orientation possibilities, there are 2^m configurations to try for a group of m edges (m is usually small). Finally, the configuration that gives the best score is used to update $\mathcal{S}$. This process is then repeated for the other groups. Because the best configuration of a group may contain cycles, the resultant $\mathcal{S}$ is repaired for cycles. Although it is possible to check for cycles when trying different orientation possibilities, we suggest avoiding it, as the involved cost will be great.

Update of the Best-So-Far Structure. To obtain the best structure during the course of searching, we have maintained a best-so-far structure separately. In each generation, we try to *merge* the currently best-so-far structure with $\mathcal{S}$. Because $\mathcal{S}$ may contain some good partial structures, it is expected that a good solution can be obtained by accumulating the essential components of $\mathcal{S}$ into the best-so-far structure. If a better structure is created, the new structure will become the best-so-far structure.

To perform the recombination, we invent a heuristic that attempts to exchange parent sets between the two network structures. Because the score of a network structure is simply the summation of the scores of the parent sets, it enables us to perform greedy operations so that better ones (i.e., parent sets) will substitute the worse. Furthermore, our heuristic algorithm performs cycle checking for each substitution so that the outcome has a legal structure.

[9] To be specific, there are two cases of conflict. First, $X \rightarrow Y$ and $Y \rightarrow X$ are both found on the list of update. Second, $X \rightarrow Y$ is found and there is no update for node X and $Y \rightarrow X$ is contained in $\mathcal{S}$.

6.5 Performance of CCGA

6.5.1 Experimental Methodology

A common practice to assess the performance of a Bayesian network learning algorithm is to test the algorithm on data sets that are generated from known network structures by probabilistic logic sampling [6.37]. We follow this practice and test our algorithm on seven different data sets. All of the data sets are generated from well-known benchmark Bayesian networks, which include the ALARM, the ASIA, and the PRINTD networks. Table 6.1 gives a summary of the data sets that we used in our experiments.

Five of the data sets are generated from the ALARM network obtained from different sources. Originally, the ALARM network was used in the medical domain for potential anesthesia diagnosis in the operating room [6.38]. Because the network, with 37 nodes and 46 directed edges, has a complex structure, it is widely used to evaluate the performance of a learning algorithm. The PRINTD network is primarily constructed for troubleshooting printer problems in the Windows^{TM} operating system [6.3]. It has 26 nodes and 26 edges. The ASIA-1000 data set is generated from the ASIA network, which is a relatively simple structure that contains eight nodes and eight edges. The network is also known as the "chest-clinic" network, which describes a "fictitious medical example whether a patient has tuberculosis, lung cancer, or bronchitis, related to their X-ray, dyspnea, visit-to-Asia, and smoking status" [6.39], [6.40]. The data set contains 1000 cases.

In our experiment, we compare the performance of our algorithm with MDLEP. All algorithms (including the implementation of MDLEP, obtained from the authors) are implemented in the C++ language and compiled using the same compiler.[10] The same MDL metric evaluation routine is used so that the difference among implementations is minimized. For all algorithms, the maximum size of a parent set is five. Because the algorithms are stochastic in nature, they are executed 40 times for each test instance. All our experiments are conducted on Sun Ultra-5 workstations.

[10] We use the g++ compiler with -O2 optimization level.

Data set	Original network	Size	MDL score of original network
ALARM-1000	ALARM	1000	18,533.5
ALARM-2000	ALARM	2000	34,287.9
ALARM-5000	ALARM	5000	81,223.4
ALARM-10000	ALARM	10,000	158,497.0
ALARM-O	ALARM	10,000	138,455.0
ASIA-1000	ASIA	1000	3416.9
PRINTD-5000	PRINTD	5000	106,541.6

Table 6.1. Data sets used in the experiments.

We estimate the performance of the learning algorithms using five measures, which include:

- the average MDL score of the final solutions, the smaller the better (AFS);
- the average MDL score of the best network obtained in the first generation (AIS);
- the average execution time in seconds (AET);
- the average generation that the best-so-far solution is obtained (ANG);
- the average number of MDL metric evaluations in a run (AME);
- the average structural difference, i.e., number of edges added, omitted, and reversed, between the final solution and the original network (ASD).

Recall that the algorithms are executed 40 times for each data set; the figures are, therefore, average values of 40 trials. Without any fine-tuning, we adopt the parameter values as the default setting:

- For MDLEP, we adopt the same parameter settings that appear in the original publication [6.9]: the population size is 50, the tournament size is seven, and the maximum number of generations is 5000.
- For CCGA, the cutoff value for the CI test phase is 0.3. For each species population in search phase, the population size is 20 with the crossover and mutation rate set to 0.7 and 0.2, respectively. The belief factor is 0.2. We use 1000 generations as the termination criterion.

6.5.2 Comparing CCGA with MDLEP

We provide a summary of the results in Table 6.2. In the table, the MDL score of the original network is shown under the name of the data set for reference. Besides the averaged measures, we include the standard deviations of the respective measure, which appear in parentheses.

Except for the ASIA-1000 data set, we can observe that CCGA is often able to find a better or as good network compare with MDLEP. For two of the six cases, the difference is statistically significant at the 0.05 level using the Mann-Whitney test.[11] Using the MDL score of the original network as a reference, we observe that although MDLEP performs well (competitive with CCGA) for smaller data sets, it clearly needs a longer running time to compete with our approach for larger data sets. For the ALARM-O data set, we regard it as a harder problem instance. The data set is relatively large, and both algorithms fail to approximate the score of the original network. However, CCGA still has better performance. For the PRINTD 5000-data set, both algorithms could recover the original network structure and hence the two have identical performance. For the ASIA 1000-data set, we find that MDLEP outperforms CCGA in terms of the final score. On closer inspection, we find that this is because an important edge in the network has been cut away during the CI test phase. Consequently, CCGA is stuck at a local

[11] The Mann-Whitney test is a nonparametric test that suits our need as the final score is observed not to follow a normal distribution.

Data Set		AFS	AIS	AET	ANG	AME	ASD
ALARM-1000 (18533.5)	CCGA	17,877.3 (38.5)	23,527.7 (885.6)	44.9 (1.1)	398.6 (290.7)	5978.2 (110.1)	12.6 (2.3)
	MDLEP	17,990.5 (73.1)	30,831.0 (795.6)	1003.9 (70.8)	4301.2 (654.3)	22,133.8 (619.3)	19.4 (4.2)
ALARM-2000 (34287.9)	CCGA	33,836.5 (92.8)	45,720.0 (1,750.3)	68.1 (1.7)	384.4 (265.6)	8710.8 (139.9)	8.2 (1.2)
	MDLEP	33,932.6 (215.8)	56,896.6 (1,259.5)	1307.8 (125.1)	4046.6 (634.1)	25,905.8 (911.3)	12.9 (4.9)
ALARM-5000 (81233.4)	CCGA	81,033.1 (64.4)	111,738.0 (4,389.9)	114.1 (1.5)	422.1 (264.9)	9118.1 (139.5)	6.1 (0.5)
	MDLEP	81,287.6 (419.9)	134,487.2 (1,836.0)	1843.2 (359.0)	3946.3 (651.2)	29,570.8 (1,016.3)	10.7 (4.9)
ALARM-10000 (158497.0)	CCGA	158,432.4 (16.3)	224,246.3 (7,070.0)	204.6 (3.6)	422.8 (286.8)	10,531.5 (96.2)	3.4 (0.7)
	MDLEP	158,704.4 (513.1)	256,946.2 (3,843.7)	2435.1 (350.1)	3596.7 (720.0)	32,160.8 (1,538.0)	8.7 (5.1)
ALARM-O (138455.0)	CCGA	138,854.3 (564.1)	217,386.1 (12,898.4)	269.8 (32.5)	462.7 (247.1)	13,635.7 (186.4)	11.9 (5.0)
	MDLEP	138,913.4 (460.8)	252,818.4 (5,862.0)	4,09.9 (2,021.3)	4523.8 (482.1)	34,309.5 (1,327.5)	17.5 (6.9)
ASIA-1000 (3416.9)	CCGA	3413.4 (0.0)	3650.7 (104.1)	2.9 (0.1)	5.9 (5.3)	42.1 (0.5)	3.7 (0.5)
	MDLEP	3398.6 (0.0)	3590.2 (48.5)	76.3 (0.4)	79.6 (30.2)	656.8 (9.2)	3.5 (0.5)
PRINTD-5000 (106541.6)	CCGA	106,541.6 (0.0)	114,967.2 (900.3)	66.4 (7.5)	10.1 (3.9)	2552.1 (43.8)	0.0 (0.0)
	MDLEP	106,541.6 (0.0)	116,089.6 (546.4)	704.5 (13.8)	512.1 (95.8)	17,688.4 (373.7)	0.0 (0.0)

Table 6.2. Performance comparison between CCGA and MDLEP.

optimal solution. In another setting, when we set the cutoff value to 1 (i.e., ignore the test results), we find that the performance of CCGA equals that of MDLEP.

Regarding the structural difference measure (i.e., ASD), we observe that CCGA consistently performs better than MDLEP. There are two possibilities that can account for this observation: One directly relates to the CI tests, another relates indirectly. On the one hand, the CI test phase possibly helps to focus the search on dealing with only the *correct* edges (that appear in the original structure) so that the result returned will be similar to the original one. Although MDLEP may be successful in finding structures with low scores, such structures may contain some *wrong* edges. Hence, it will be the merit of the entire hybrid learning framework, which helps to recover structures that closely resemble the original one. On the other hand, the observation could also be explained by the fact that better results are obtained because searching is efficient. In this regard, we assume that the metric directly relates to the structural difference measure. Hence, networks with good scores will also resemble the original network. Because the searching is made efficient as a consequence of the reduction of search space by CI tests, we can often find these good solutions that make ASD small.

Apart from the quality of the final solutions, we observe that CCGA has a tremendous speedup over MDLEP. In general, the gain is more than ten-fold (from 10.6 to 26.5). We conjecture that there are two important reasons: (1) Due to the hybrid framework, the search space is reduced. Therefore, CCGA uses significantly less MDL metric evaluations (i.e., AME) than MDLEP, which is crucial because evaluation of the MDL metric is a time-consuming operation.[12] Despite the fact that less evaluation is made, CCGA is still effective in finding good solutions. (2) Because CCGA requires much fewer cycle-repairing operations (only for the collaborative structure $\mathcal{S}$ and the merged best-so-far structure) than MDLEP, it can save time.

Because CCGA executes faster and the results can still be competitive with that of MDLEP, we conclude that CCGA is more efficient than MDLEP.

6.6 Conclusion

In this chapter, we proposed a new algorithm, CCGA, for learning Bayesian networks more efficiently. The algorithm was tested on a number of network learning problems, and good Bayesian networks were obtained. We compared CCGA with MDLEP and found that CCGA is much more efficient. Moreover, CCGA usually discovers better Bayesian networks. With an efficient algorithm, it enables us to explore interesting applications of Bayesian networks on real-world data-mining problems.

Acknowledgment

This research was supported by the Earmarked Grant LU3012/01E from the Research Grant Council of the Hong Kong Special Administrative Region.

References

6.1 Jensen, F. V.,*An Introduction to Bayesian Network*, University College of London Press, 1996.
6.2 Heckerman, D., and Horvitz, E., Inferring informational goals from free-text queries: A Bayesian approach, in Cooper, G. F., and Moral, S. (eds.), *Proc. 14th Conference of Uncertainty in Artificial Intelligence*, Wisconsin, pp. 230–7, Morgan Kaufmann, July 1998.
6.3 Heckerman, D., and Wellman, M. P., Bayesian networks, *Communications of the ACM*, 38, pp. 27–30, March 1995.

[12] In our current implementation, every record needs to be examined once for each query.

6.4 Cheng, J., Bell, D. A., and Liu, W., Learning Bayesian networks from data: An efficient approach based on information theory, in *Proc. 6th ACM International Conference on Information and Knowledge Management*, Las Vegas, NV, pp. 325–31, ACM, Nov. 1997.
6.5 Spirtes, P., Glymour, C., and Scheines, R., *Causation, Prediction, and Search*, vol. 81 of *Lecture Notes in Statistics*, New York: Springer-Verlag, 1993.
6.6 Potter, M. A., and De Jong, K. A., Evolving neural networks with collaborative species, in *Proc. 1995 Summer Computer Simulation Conference*, Ottawa, Canada, 1995.
6.7 Potter, M. A., De Jong, K. A., and Grefenstette, J., A coevolutionary approach to learning sequential decision rules, in *Proc. 6th International Conference (ICGA'95)*, pp. 366–72, July 1995.
6.8 Potter, M. A., *The Design and Analysis of a Computational Model of Cooperative Coevolution*, Ph.D. thesis, George Mason University, 1997.
6.9 Wong, M. L., Lam, W., and Leung, K. S., Using evolutionary programming and minimum description length principle for data mining of Bayesian networks, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21, pp. 174–8, Feb. 1999.
6.10 Pearl, J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, 1988.
6.11 Fung, R. M., and Crawford, S. L., Constructor: A system for the induction of probabilistic models, in *Proc. 7th National Conference on Artificial Intelligence*, pp. 762–9, 1990.
6.12 Spatz, C., and Johnston, J. O., *Basic Statistics: Tales of Distribution*, 2nd ed., Monterey, CA: Brooks/Cole Publishing Company, 1981.
6.13 Agresti, A., *Categorical Data Analysis*, New York: Wiley, 1990.
6.14 Beaumont, G. P., and Knowles, J. D., *Statistical Tests: An Introduction with MINITAB Commentary*, Prentice-Hall, 1996.
6.15 de Campos, L. M., and Huete, J. F., Approximating causal orderings for Bayesian networks using genetic algorithms and simulated annealing, Tech. Rep., Department of Computer Science and Artificial Intelligence, University of Granada, Spain, May 1999.
6.16 Dash, D., and Druzdzel, M. J., A hybrid anytime algorithm for the construction of causal models from sparse data, in Laskey, K. B., and Prade, H., (eds.), *Proc. 15th Conference on Uncertainty in Artificial Intelligence*, Stockholm, Sweden, pp. 142–9, Morgan Kaufmann, July–Aug. 1999.
6.17 Rissanen, J., Modelling by shortest data description, *Automatica*, 14, pp. 465–71, 1978.
6.18 Bouckaert, R. R., *Bayesian Belief Networks: From Inference to Construction*, Ph.D. thesis, Utrecht University, June 1995.
6.19 Suzuki, J., Learning Bayesian belief networks based on the minimum description length principle: Basic properties, in *IEICE Transactions Fundamentals*, E82-A, Nov. 1999.
6.20 Lam, W., and Bacchus, F., Learning Bayesian belief networks–an approach based on the MDL principle, *Computational Intelligence*, 10(4), pp. 269–93, 1994.
6.21 Chickering, D. M., Geiger, D., and Heckerman, D., Learning Bayesian network is NP-Hard, Tech. Rep., Microsoft Research, 1994.
6.22 Herskovits, E., and Cooper, G., A Bayesian method for the induction of probabilistic networks from data, *Machine Learning*, 9(4), pp. 309–47, 1992.

6.23 Tian, J., A branch-and-bound algorithm for MDL learning Bayesian networks, in Boutilier, C., and Goldszmidt, M. (eds.), *Proc. 16th Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA, pp. 580–8, Morgan Kaufmann, June–July 2000.
6.24 Larrañaga, P., Kuijpers, C., Mura, R., and Yurramendi, Y., Structural learning of Bayesian network by genetic algorithms: A performance analysis of control parameters, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18, pp. 912–26, Sept. 1996.
6.25 Bäck, T., *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Agorithms*, Oxford University Press, 1996.
6.26 Fogel, D. B., *Evolutionary Computation: Toward a New Philosohpy*, IEEE Press, 2000.
6.27 Koza, J. R., *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992.
6.28 Bäck, T., Fogel, D. B., and Michalwicz, Z. (eds.), *Evolutionary Computation 1: Basic Algorithms and Operators*, Institute of Physics Publishing, 2000.
6.29 Bäck, T., Fogel, D. B., and Michalwicz, Z. (eds.), *Evolutionary Computation 2: Advanced Algorithms and Operators*, Institute of Physics Publishing, 2000.
6.30 Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.
6.31 Holland, J. H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.
6.32 Höffgen, K.-U., Learning and robust learning of product distributions, *ACM COLT*, pp. 77–83, 1993.
6.33 Fogel, L., Owens, A., and Walsh, M., *Artificial Intelligence Through Simulated Evolution*, John Wiley and Sons, 1966.
6.34 Singh, M., and Valtorta, M., An algorithm for the construction of Bayesian network structures from data, in Heckerman, D., and Mamdani, E. H. (eds.), *Proc. 9th Conference of Uncertainty in Artificial Intelligence*, San Mateo, CA, pp. 259–65, Morgan Kaufmann, 1993.
6.35 Acid, S., and de Campos, L. M., BENEDICT: An algorithm for learning probabilistic belief networks, in *Proc. IPMU-96 Conference*, pp. 979–84, 1996.
6.36 Beasley, D., Bull, D. R., and Martin, R. R., An overview of genetic algorithms: Part 1, fundamentals, *University Computing*, 15(2), pp. 58–69, 1993; ftp://ralph.cs.cf.ac.uk/pub/papers/GAs/ga_overview1.ps.
6.37 Henrion, M., Propagating uncertainty in Bayesian networks by logic sampling, in Lemmar, J., and Kanal, L. (eds.), *Proc. 2nd Conference on Uncertainty in Artificial Intelligence*, Amsterdam, North Holland), pp. 149–63, 1988.
6.38 Beinlinch, I., Suermondt, H., Chavez, R., and Cooper, G., The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks, in *Proc. 2nd European Conference Artificial Intelligence in Medicine*, pp. 247–56, 1989.
6.39 Norsys Bayes net library, http://www.norsys.com/net_library.htm.
6.40 Lauritzen, S. L., and Spieglhalter, D. J., Local computations with probabilities on graphical structures and their application to expert systems, *Journal of Royal Statistics Society*, 50(2), pp. 157–94, 1988.

7. Knowledge Discovery and Data Mining in Medicine

Takumi Ichimura,[1] Shinichi Oeda,[2] Machi Suka,[3] Akira Hara,[1]
Kenneth J. Mackin,[4] and Katsumi Yoshida[3]

[1] Faculty of Information Sciences, Hiroshima City University, 3-4-1, Ozuka-higashi, Asaminami-ku, Hiroshima 731-3194, Japan; email: {ichimura, ahara}@its.hiroshima-cu.ac.jp

[2] Department of Information and Computer Engineering, Kisarazu National College of Technology, 2-11-1, Kiyomidai-higashi, Kisarazu, Chiba 292-0041, Japan; email: oeda@j.kisarazu.ac.jp

[3] Department of Preventive Medicine, St. Marianna University School of Medicine, 2-16-1, Sugao, Miyamae-ku, Kawasaki 216-8511, Japan; email: {suka,k2yosida}@marianna-u.ac.jp

[4] Department of Information Systems, Tokyo University of Information Sciences, 1200-2, Yatohcho, Wakaba-ku, Chiba 265-8501, Japan; email: mackin@rsch.tuis.ac.jp

Medical databases store diagnostic information based on patients' medical records. Because of deficits in patients' medical records, medical databases do not provide all the required information for learning algorithms. Moreover, we may meet some contradictory cases, in which the pattern of input signals is the same, but the pattern of output signals is different. Learning algorithms cannot correctly classify such cases. Even medical doctors require more information to make the final diagnosis. In this chapter, we describe three methods of classifying medical databases based on neural networks and genetic programming (GP). To verify the effectiveness of our proposed methods, we apply them to real medical databases and prove their high classification capability. We also introduce techniques for extracting If-Then rules from the trained networks.

7.1 Introduction

Medical databases store diagnostic information based on patients' medical records. Medical information such as laboratory tests, lifestyles, and chief complaint is often ambiguous, and it is difficult to distinguish between normal and pathological values.

Because of deficits in patients' medical records, medical databases do not provide all the required information for learning algorithms. There may be some contradictory cases, in which the pattern of input signals is the same, but the pattern of output signals is different. Learning algorithms cannot correctly classify such cases. Even medical doctors require more information to make the final diagnosis.

Many algorithms for neural networks have been developed. Different algorithms focus on different facets of learning, such as finding the optimal

network structure, reducing the effect of initial values of parameters, deciding on the optimal learning parameters, and improving classification accuracy. However, we may face the following problem in a typical neural network learning environment. If we use only good examples without noise and missing data (i.e., some data items are missing), the trained network reasonably classifies target patterns but may not have the ability to classify many random patterns into optimal categories. On the contrary, if we train the network to classify random patterns, the classification capability of target patterns may not be very good.

To avoid such a disadvantage in the learning, we propose the learning method of immune multiagent neural networks (IMANNs) [1]. The IMANN has macrophage agents, T-cell agents, and B-cell agents. The macrophage and T-cell agents employ the planar lattice neural networks (PLNN) with a neuron generation/annihilation algorithm [2]. The PLNN consists of hidden neurons in the lattice and works a similar way to self-organized map (SOM) [3]. B-cell agents employ Darwinian neural networks (DNN) [4], which have a structural learning algorithm based on Darwin's theory of evolution.

On the other hand, we use genetic programming (GP) combined with multiagent systems for rule extraction. We assume that some agents extract general rules from frequently observed data and other agents extract exceptional rules from rarely observed data. With cooperation between multiple agents, the extracted rules can cover all the data. We propose an improved GP method, automatically defined groups (ADG). The ADG aims to realize effective cooperative behaviors among heterogeneous agents [5], [6]. This method can optimize both the group structure of agents and the action rule of each group.

Moreover, there is a learning method for knowledge-based artificial neural networks (KBANN) with structure-level adaptation of the NN. The KBANN represents the knowledge structure of experts in the initial network structure and generates or annihilates hidden neurons to reach a suitable structure during learning. Extracting rules from KBANN may be easier than from usual neural networks, because the initial network structure is transformed from If-Then rules given by experts.

Sections 7.2, 7.3, 7.4, and 7.5 describe KBANN with structure-level adaptation, ADG, and IMANN, respectively. To verify the effectiveness of our proposed methods, we apply them to real medical databases and prove their high classification capability. We also introduce techniques for extracting If-Then rules from the trained networks.

7.2 KBANN with Structure Level Adaptation

We propose a method of knowledge based artificial neural network with structure-level adaptation of the NN. This method can determine network structure to represent the knowledge structure of medical experts. The network structure translates the intermediate assumption that medical doctors usually set up before their final diagnosis. This method can generate new neurons or annihilate redundant ones. The neuron generation/annihilation algorithm, that is,

structure-level adaptation of the NN [7], makes a suitable configuration of a network on the basis of the observation of change in connection weights and output activities of hidden neurons during tanning. Using the structure-level adaptation of the NN, we can give a clear explanation of the relation between input and output signals. To verify the effectiveness of this method, we develop a model of the occurrence of hypertension [8].

7.2.1 KBANN

Towell and Shavlik proposed the rule-to-network algorithm [9] shown in Table 7.1. This algorithm is abstracted from the seven-step rules-to-network translation algorithm, which consists of rewriting, mapping, numbering, adding hidden neurons, adding input neurons, adding links, and perturbing.

1. Rewriting. The first step of the algorithm transforms the set of approximately correct rules into a format that clarifies its hierarchical structure and makes it possible to directly translate the rules into a neural network. If there is more than one rule for a consequence, then every rule for this consequence is rewritten as two rules, as shown in Fig. 7.1. B' and B'' are newly created antecedents terms. Rules do not map from inputs to outputs directly. Sometimes the rules provide intermediate conclusions, which may be used by other rules to reach the final conclusion or other intermediate conclusions. The rule sets form the hierarchical

Table 7.1. The rule-to-network algorithm.

1.	Rewrite rules so that disjunctions are expressed as a rule set that has only one antecedent.
2.	Directly map the rule structure into a neural network.
3.	Label neurons in the network according to their "level."
4.	Add hidden neurons to the network at user-specified levels.
5.	Add neurons to known input features that are not referred to in the rules.
6.	Add links that are not specified by translation between all neurons in topologically contiguous levels.
7.	Perturb the network by adding near-zero random numbers to each of the link weights.

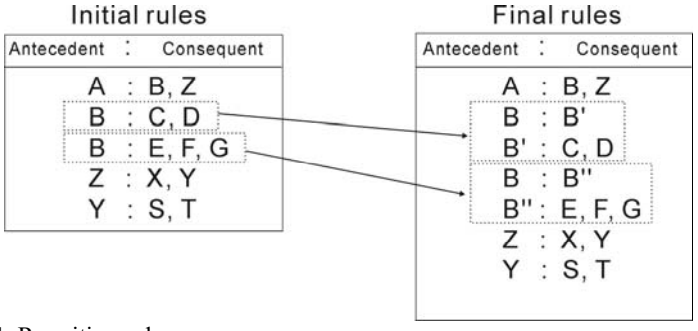


Fig.7.1. Rewriting rules.

Table 7.2. Correspondences between knowledge and networks.

Knowledge base	Neural network
Final conclusions	Output neurons
Facts	Input neurons
Intermediate conclusions	Hidden neurons
Dependencies	Weight connections

structure, which creates the feature used in the example-based learning system. Therefore, the created networks will have no derived feature that indicates contextual dependencies or other useful conjunctions within example descriptions.

2. Mapping. The second step establishes a mapping between a transformed rule set and a neural network, as shown in Table 7.2. This step can create a network that has a one-to-one correspondence with the rule set.

3. Numbering. This step numbers neurons in the network to define the level of each neuron as the length of the longest path to the input neurons, as shown in Fig. 7.2.

4. Adding hidden neurons. This step adds hidden neurons to the network, giving the network the ability to learn derived features that are not specified in the initial rule set but are suggested by the expert, as shown in Fig. 7.3.

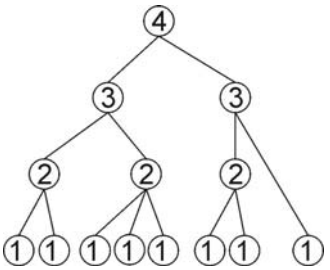


Fig.7.2. Numbering neurons.

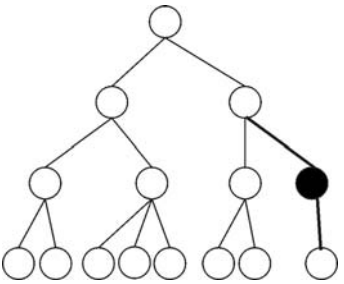


Fig.7.3. Adding hidden neurons.

5. Adding input neurons. This step adds neurons to known input features that are not referred to in the rules, because the rule set that is not perfectly correct may not identify every input feature required to correctly learn the concept.

6. Adding links. This step adds links with weight zero to the network using the neuron numbering established in Step 4.

7. Perturbing. The final step is to perturb the network by adding a small random number to each of the link weights.

However, the rule-to-network algorithm of KBANN cannot change its network structure according to the situation of training the records. Even if there are some records with new features in the database, we cannot find a new rule that goes beyond the knowledge of medical experts. To find a new rule in the database, we use the structure-level adaptation of the NN, which develops a suitable network structure in the Step 7.

7.2.2 Structure-Level Adaptation of the NN [10]

Usual neural networks shows the following behaviors during learning:

- If a neural network does not have enough neurons to infer, the input weight vector of each neuron may fluctuate greatly, even after an long enough period of learning.
- If a neural network has more neurons than what is required to infer, then even after the input weight vector of each neuron converges, the network may have unnecessary neurons.

In the first case, the network should generate a new neuron inheriting the attributes of its parent neuron. In the second case, redundant neurons should be deleted from the network through weight calculation. Based on these cases, we determine the suitable conditions for neuron generation or annihilation in the learning process.

7.2.2.1 Neuron Generation

Neuron generation occurs when the representation power of the network is insufficient. Given enough neurons in a hidden layer, a neural network can map with precision. Therefore, we used a stabilized error as the index to determine whether the network needs to generate a new neuron. If a stabilized error after an adequate period of learning is larger than the desired value, a new neuron is generated.

The next problem is how to determine the position of the new neuron in the hidden layer. The optimal position can be found through monitoring the behaviors of the neurons in the hidden layer. The neuron needs adequate representation power to contribute to the final system error through the fluctuation in its input weight vector, because the input weight vector of each neuron fluctuates greatly even after an adequate period of learning. When the neuron dose not have enough

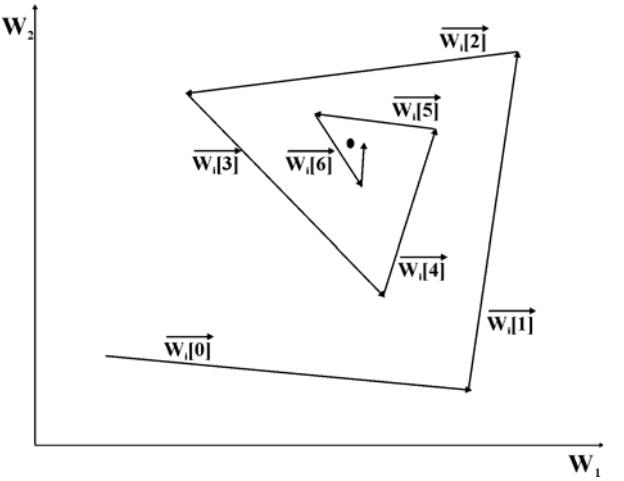


Fig.7.4. Convergence of the input weight of a neuron.

representation power to learn the subproblem, the neuron was split into two, i.e., we added another neuron to the same interconnection of the network as its parent neurons' attributes are inherited.

Figure 7.4 shows an idea for converging an input weight of a neuron through searching optimal spaces of weights W_1, W_2.

During the convergence process, the average length of movement between adjacent temporal weights is gradually decreased. On the basis of this observation, we define the following measure, called the walking distance (WD) of the neuron:

$$WD_j[n] = \sum_{m=1}^{n} Met(\vec{W}_j[m], \vec{W}_j[m-1]), \tag{7.1}$$

where $WD_j[n]$ is the WD of neuron j in the hidden layer at iteration n, $\vec{W}_j[m]$ is the input weight of neuron j at arbitrary iteration m, and Met is a metric that measures the distance between vectors in a metric space. In this chapter, Met is the Euclidean metric:

$$EuMet(\vec{X}, \vec{Y}) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}, \tag{7.2}$$

where

$$N = Dim(\vec{X}) \text{ or } Dim(\vec{Y}). \tag{7.3}$$

In Eq. (7.1), WD_j is a measure for the time variance of the stored information for neuron j; it can be considered the activity of neuron j in the parameter space. Too large a WD of a particular neuron indicates that the processing capability of the neuron is insufficient and that the load should be shared with

another new neuron. Eq. (7.1) is approximated by the following operational measure:

$$WD_j[n] = \gamma_W WD_j[n-1] + (1-\gamma_W) Met(\vec{W}_j[m], \vec{W}_j[m-1]), 0 < \gamma_W < 1. \quad (7.4)$$

A neuron j should generate another neuron if

$$\varepsilon_j \overset{\Delta}{=} \frac{\partial \varepsilon}{\partial WD_j} WD_j > \theta_G, \quad (7.5)$$

where ε is the overall system error, ε_j is the contribution of neuron j to the overall system error, WD_j is the WD of neuron j, and θ_G is a certain threshold value; ε_j determines the neuron generation.

7.2.2.2 Neuron Annihilation

To achieve an overall good system performance, we must remove unnecessary elements from the network. Due to the competitive mechanisms between neurons and the selection rule that makes the neurons keep their capability of finding their correct positions in the network and removes misplaced neurons, the network gradually forms an optimal structure in the development process. To be more specific, we have the following two observations:

- If a neuron does not form the appropriate interconnections with other neurons, it will die in the early development process.
- The neurons fight with each other because each neuron tries to stop other neurons from taking exactly the same functional role in the network.

The following criteria for the neuron annihilation process were derived from these observations. We kill the corresponding neuron if (A) the neuron is not a functional element of the network or (B) the neuron is a redundant element of the network. Criterion (A) can be identified by monitoring the output activity of the neuron. As a measure for this criterion, we use the variance of the output activity of neuron j:

$$VA_j = (\overline{O}_j - O_j)^2, \quad (7.6)$$

where O_j is the output activation level for neuron j and $\overline{O_j}$ is the average of O_j over all training data. We defined the operational measure VA_j as

$$VA_j[n] = \gamma_V VA_j[n-1] + (1-\gamma_V)(O_j - Act_j[n])^2, 0 < \gamma_V < 1, \quad (7.7)$$

where $VA_j[n]$ is an estimate of VA_j at iteration n and $Act_j[n]$ is the operational measure of the average output activity $\overline{O_j}$ at iteration n for neuron j:

$$Act_j[n] = \gamma_a Act_j[n-1] + (1-\gamma_a) O_j[n], 0 < \gamma_a < 1. \quad (7.8)$$

VA_j is very closely related to the information content of the output activities of neuron j.

If VA_j is zero for neuron j, this neuron does not generate any additional information. In other words, this neuron does not perform any signal-processing functions. Therefore, we eliminate the neuron whose VA_j is smaller than a threshold value.

Criterion (B) can be identified by monitoring the dependence between the output values of neurons in the network. If two neurons are totally dependent, that is, given the output of one neuron the output of the other can be decided with probability one, one of the neurons can be annihilated without affecting network performance.

7.2.3 Application to a Medical Database

To verify the effectiveness of the KBANN with structure-level adaptation of the NN, we develop a model of the occurrence of hypertension. In this chapter, the prior knowledge is given by medical experts, who decided 16 input terms and 1 output term. There are two kinds of intermediate assumptions between the input terms and the output term. Among the input terms, 10 terms are categorized as biochemical tests related to the measurement of blood pressure for past five years; the remaining terms are "gender," "age," "obesity index," "γ-GTP," and "volume of consumed alcohol." The output term represents whether the patient developed hypertension during the five years. Such information is transferred into a network using the concept illustrated in Table 7.3. Figure 7.5 shows the network structure for the model of the occurrence of hypertension.

Table 7.3. Example records for the occurrence of hypertension.

No.	Occur	Gender	Age	Obesity index	γ-GTP	Consumed alcohol
1	+	m	48	25.86	54	13
2	+	f	45	25.56	13	0
3	-	f	47	21.05	10	0
4	-	m	43	26.61	17	0
5	+	m	49	28.73	40	0

No.	Systolic blood pressure (one per year)					Diastolic blood pressure (one per year)				
	1st	2nd	3rd	4th	5th	1st	2nd	3rd	4th	5th
1	120	122	137	116	153	76	73	83	80	93
2	132	141	120	151	139	84	74	82	85	79
3	120	113	136	111	126	70	58	78	64	69
4	120	136	120	130	136	66	80	80	80	83
5	138	140	134	138	140	78	80	88	88	90

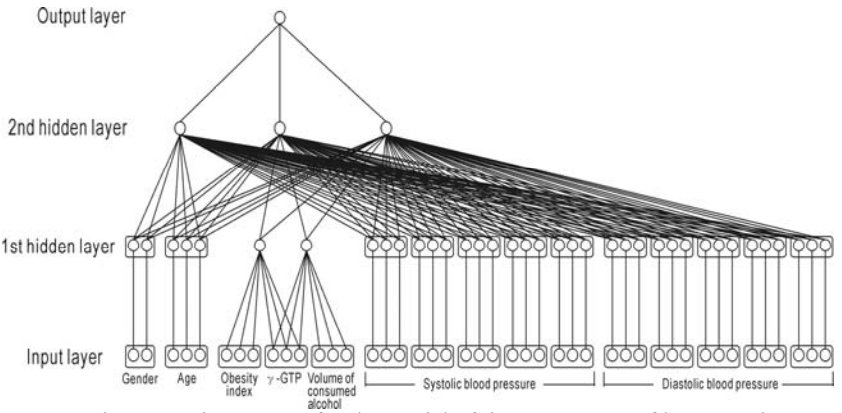


Fig.7.5. The network structure for the model of the occurrence of hypertension.

Using BP learning, the network was trained with a real medical database containing 1024 patient records. In this investigation, we used a training set consisting of 100 occurrence data and 100 no-occurrence data selected by random sampling. Table 7.3 shows a few examples from the training set corresponding to the occurrence of hypertension.

Medical information, such as laboratory tests, lifestyles, and chief complaint, is often ambiguous, and it is difficult to distinguish between normal and pathological values by a crisp threshold. In the database of the occurrence of hypertension, each term was given two cutoff values, as shown in Table 7.4; these provide gray zones between normal and pathological values.

The developed system had correctly classified 95.0% of all patient records. We detected a neuron with large *WD* in the 2nd hidden layer related to blood pressure. According to the neuron generation algorithm, this neuron was split into two and the attributes of the old neuron were inherited by the new neurons as shown in Fig. 7.6. After we trained the network again using a new network structure, the newly developed system correctly diagnosed 96.7% of patient records. In this work, the neuron annihilation process was not applied, as there was no neuron to be annihilated.

As a result, a hidden neuron was added to the initial network structure. The initial network structure was constructed to represent the knowledge structure of

Table 7.4. Cutoff values for the occurrence of hypertension.

	Value 1	Value 2
Age	40	50
Obesity index	20	24
γ-GTP	60	100
Volume of consumed alcohol	7	15
Systolic blood pressure	85	90
Diastolic blood pressure	130	140

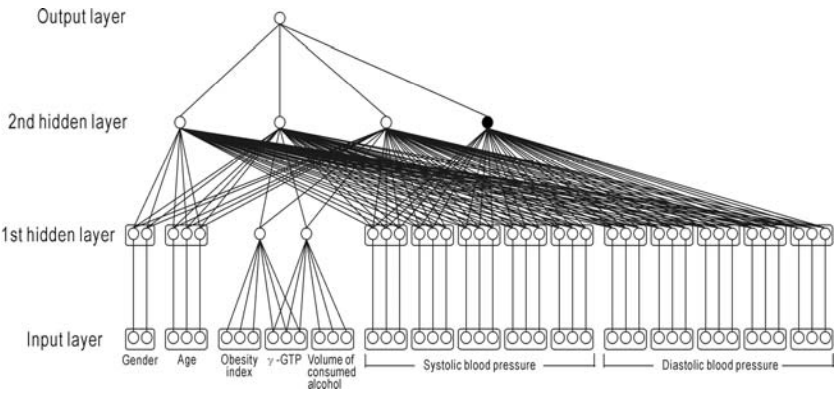


Fig.7.6. Modified network structure after structure-level adaptation.

medical experts. However, it is difficult to translate all of their knowledge into rule sets and prepare a perfect network structure. The newly added neuron may compensate such a deficit in rule sets.

In the next section, we extract fuzzy rules from the network to give a clear explanation of the relation between input and output signals.

7.2.4 Extracting Rules from KBANN

Extracting explicit rules from neural networks has attracted a lot of attention because it has potential for applications to knowledge acquisition from databases. However, distributed representation of knowledge in the hidden layers makes it difficult to extract explicit rules from trained networks.

In the case of KBANN, the initial network structure is transformed from *If-Then* rules given by experts. Extracting rules from KBANN may be easier than from usual neural networks. After the structure-level adaptation of a NN, the initial network structure and the trained one differ only by the number of neurons in the hidden layers. We can acquire new knowledge by investigating the neurons that are newly added to the hidden layers.

First, we should investigate the weights connected to the new neurons. If a connection weight value is smaller than a predetermined threshold value, the connection to the hidden neuron does not represent new knowledge. On the contrary, if the connection weight value is larger than the predetermined threshold value, an *If-Then* rule can be extracted as new knowledge from the strength of its weight. The connections between the input neurons and a hidden neuron indicate the antecedent part of an *If-Then* rule. The connection between a hidden neuron and an output neuron indicates the consequent part of an *If-Then* rule. The strength of its weight represents the agreement between the antecedent part and the consequent part represented by the fuzzy membership functions. The fuzzy membership function uses the fuzzy linguistic values {“very true,” “true,” “rather

true," "unknown," "rather false," "false," "very false"} to represent the strength of the connection weight to hidden neurons. The network is trained to make the connection weights lie in [0,1]. Then the extracted fuzzy rule is represented as follows:

$$\textit{If } x \textit{ is } m \textit{ true} \quad \textit{Then } y \textit{ is } n \textit{ true}, \tag{7.9}$$

where m and n represent the fuzzy linguistic values. In this chapter, we use the transformation rules shown in Table 7.5, which convert the strength of the connection weight to the fuzzy linguistic value.

By applying this method to the modified network, we extract fuzzy rules for the model of the occurrence of hypertension. Table 7.6 shows the fuzzy rules for a new neuron in the 2nd hidden layer. These rules are not inconsistent with medical experts' consensus.

Table 7.5. Truth scales and their corresponding numerical values.

Linguistic value	Numerical value
Very true	$0.925 < \mu \leq 1.000$
True	$0.775 < \mu \leq 0.925$
Rather true	$0.600 < \mu \leq 0775$
Unknown	$0.400 < \mu \leq 0.600$
Rather false	$0.225 < \mu \leq 0.400$
False	$0.075 < \mu \leq 0.225$
Very false	$0.000 \leq \mu \leq 0.075$

Table 7.6. The extracted fuzzy rules from a new neuron.

Antecedent part	Consequent part
1st SBP is rather true and 2nd SBP is true and 3rd SBP is true and 4th SBP is true and 5th SBP is true and 1st DBP is true and 2nd DBP is true	Predisposition is false
Gender is female and age(Old) is rather false and obesity index is rather false and consumed alcohol is rather false and 3rd DBP is rather true and 4th DBP is rather true and 5th DBP is rather true	Predisposition is true

7.3 Rule Extraction by ADG

Cooperative problem solving by multiagent systems has attracted increasing attention in recent years. A number of attempts that generate cooperative behavior by means of genetic programming (GP) have been made in the domain of multirobot control, RoboCup soccer agents, and so on. To deal with tasks requiring team solutions, the sharing of roles among agents is needed. Hara et al. have proposed automatically defined groups (ADG) with the aim of realizing effective cooperative behavior among heterogeneous agents [5], [6]. This method can optimize both group structure of agents and action rules of each group.

In this section, we describe a proposed method for rule extraction using ADG and its application to a diagnostic system for hepatobiliary disorders.

7.3.1 Cooperative Problem Solving by Multiple Agents

When we generate action control rules of multiagent GP, there are two conventional models, the homogeneous model and the heterogeneous model. When all agents in the environment take actions under identical rules, the team is called a homogeneous team. In GP, each agent refers to the same tree, as shown in Fig. 7.7. All agents decide their movements according to the same rules derived from the GP tree. However, because each agent is situated in a different environment, it is possible that each agent takes different actions according to the conditions and solves the problem by cooperating with each other.

When different agents in the environment take actions under different rules, this team is called a heterogeneous team. In GP, an individual maintains multiple trees, each of which is referred to by the corresponding agent, as illustrated in Fig. 7.7. In the heterogeneous model, the various breeding strategies (restricted breeding, free breeding, etc.) have been proposed [11], [12]. Free breeding allows any member of a team to freely breed with any other member of another team. In restricted breeding, crossover operations are restricted to corresponding branch pairs. For instance, restricted breeding allows team member 1 to breed only with another team member 1, and team member 2 to breed only with another team member 2. Generally, restricted breeding works better than free breeding because the restriction promotes preservation of diversity and specialization of each agent by dividing team members into separate breeding pools.

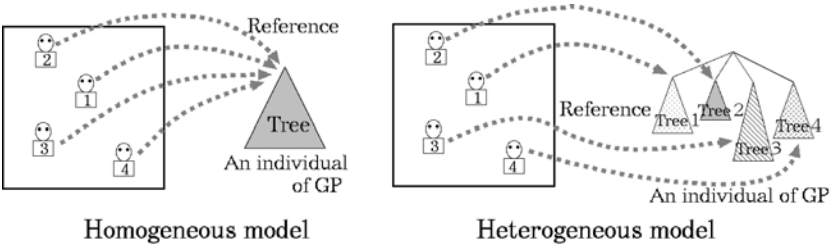


Fig.7.7. Conventional models for multi-agent control.

To solve a complex task requiring teamwork, sharing of roles among agents is needed. Generally, it is shown that the performance of heterogeneous agents is better than that of homogeneous agents, because each agent in a heterogeneous model is specialized according to its role.

This multiagent approach is effective in solving problems that seem to be unrelated to the concept of agents. Soule applied a multiagent approach to even-parity problems and linear regression problems and showed that the performance of this approach is better than that of conventional solutions [13], [14]. In these experiments, a solution is obtained by collecting each agent's output.

In this research, we apply such an idea and search for various solutions by heterogeneous agents, to the domain of knowledge acquisition from data. We handle the data containing multiple rules and aim to do both clustering of the data and rule extraction from each cluster. We use a multiagent approach to solve this problem. That is, the data are divided among agents. This corresponds to clustering of data. And each agent generates an approximate function for the assigned data. This corresponds to rule extraction in each cluster. As a result, all rules are extracted by multiagent cooperation.

To use this approach, however, we need to know the number of rules hidden in data and how to allot data to each agent. Moreover, as we prepare abundant agents, the number of trees in an individual increases. Therefore, search performance becomes poor. The proposed method using ADG can address these problems.

7.3.2 Automatically Defined Groups

In ADG, agents can take different programs, which are needed to solve the problem. Moreover, by grouping multiple agents that have similar roles, we can monitor the increase in the search space. However, we do not know the optimal number of groups of grouping of agents beforehand, ADG can address these issues by evolution. The algorithms for ADG are as follows.

A team that consists of all agents is regarded as one GP individual. One GP individual maintains multiple trees, each of which functions as a specialized program for a distinct group. We define a group as the set of agents referring to the same tree for the determination of their actions. All agents belonging to the same group use the same program.

Generating an initial population, agents in each GP individual are divided into random groups. Basically, crossover operations are restricted to corresponding tree pairs. For example, a tree referred to by an agent 1 in a team breeds with a tree referred to by an agent 1 in another team. However, we consider the sets of agents that refer to the trees used for the crossover. The group structure is optimized by dividing or unifying the groups according to the relation of the sets. Individuals search solutions as their group structure approaches the optimal one gradually.

The concrete processes are as follows: We assign an agent arbitrarily to two parental individuals. A tree referred to by the agent in each individual is used for

crossover. We use T and T' as expressions of these trees, respectively. In each parental individual, we define a set $A(T)$ as the set of agents that refers to the selected tree T. When we perform a crossover operation on trees T and T', the following three cases may arise:

- Type a. If the relation between the sets is $A(T) = A(T')$, the structure of each individual is unchanged.
- Type b. If the relation between the sets is $A(T) \supset A(T')$, the division of groups takes place in the individual with T, so that the only tree referred to by the agents in $A(T) \cap A(T')$ can be used for crossover. The individual that maintains T' is unchanged.
- Type c. If the relation between the sets is $A(T) \not\subset A(T')$ and $A(T') \not\subset A(T)$, the unification of groups takes place in both individuals so that the agents in $A(T) \cup A(T')$ can refer to an identical tree.

We expect that this method will make the search efficient and an adequate group structure will be acquired. At the same time, the acquired group structure becomes a clue for understanding the cooperative behavior and the necessary division of labor.

7.3.3 Extraction of Rules by ADG

Each agent group represents experts that have a tree structural program as the approximate formula for describing the data. For the training data, each group calculates the predicted value from input x_i. The results acquired from respective groups become candidates for the output o_i. The value closest to the answer y_i is adopted as the output o_i from the set of candidates.

Suppose that the *k*th agent belongs to a certain group, *g*. We define the load of the agent w_k as follows:

$$w_k = (\text{adopted frequency of } g)/(\text{Number of agents that belong to } g). \quad (7.10)$$

In the training data, each group calculates the predicted value from input data. The calculation results acquired from respective groups become candidates for the output. The value closest to the answer is adopted as the output. If there are multiple groups, which have the closest value to the answer, the group with more agents is adopted. Adopted frequencies of respective groups are defined as the number of adoptions for all data. In experiments for hepatobiliary disorder, the adopted frequency of each group is counted when the rule successfully returns true for each data of the target disorder.

We calculate the variance of the load in all agents, V_a. From the point of view of minimization of prediction error and load balancing, the fitness f is defined as in Eq. (7.11). We minimize f by evolution:

$$f = \sum (y_i - o_i)^2 + \alpha V_a. \quad (7.11)$$

In addition, to inhibit redundant division of groups, a penalty, $\gamma^{G-1}\ (\gamma > 1)$, is multiplied to *f*, where *G* is the number of groups. This method has the following features:

- In conventional methods, most of the research has focused on problems of only clustering or only rule extraction from the data that have already been classified. We use data containing multiple classes, and clustering of data and rule extraction in each class are performed simultaneously.
- The number of classes in the training set can be acquired. Moreover, the ratio of the number of agents in each group corresponds to the ratio of the appearance of each class. Therefore, we can understand the probability of appearance of each class. We can also regard the group with few agents as noise.
- For test data, we can obtain candidate answers and each candidate's reliability. Recognizing that the data contain multiple classes might trigger a discovery of a useful attribute for clustering. This enables us to perform accurate prediction.

7.3.4 Extracting Rules from a Medical Database

We used hepatobiliary disorder data as our experimental data. The data consist of the results of biochemical tests for four hepatobiliary disorders and the gender of the patient. Table 7.7 shows some example values of the biochemical tests conducted for each disorder. The tests are: GOT (glutamic oxaloacetic transaminase), GPT (glutamic pyruvic transaminase), LDH (lactate dehydrase), GGT (gamma glutamyl transpeptidase), BUN (blood urea nitrogen), MCV (mean corpuscular volume of red blood cell), MCH (mean corpuscular hemoglobin), Tbil(total bilirubin), and CRT(creatinine). The disorders are "alcoholic liver damage," "primary hepatoma," "liver cirrhosis," and "cholelithiasis." We have 536 patient records with some incorrect diagnostic data. The training data set consists of 322 randomly chosen records; the remaining records are used for testing.

As discussed earlier, medical information, such as the results of biochemical tests and chief complaint, is often ambiguous. We cannot clearly distinguish the difference between normal and pathological values. Biochemical test values cannot be precisely evaluated by using crisp sets. So we set up three cutoff values in Table 7.8. Therefore, each biochemical item has four levels. The functions and terminals in GP are as follows. The functional symbols are {AND, =, >,<}. Each function has two arguments. The terminal symbols are night biochemical test items and discrete values (0,1,2,3).

Table 7.7. Example of biochemical tests for four hepatobiliary disorders: (a) Alcoholic liver damage, (b) primary hepatoma, (c) liver cirrhosis, and (d) cholelithiasis.

	Gender	GOT	GPT	LDH	GGT	BUN	MCV	MCH	Tbil	CRT
a)	Male	108	114	344	176	114.8	99.7	33.1	0.6	0.9
b)	Male	354	104	1047	265	21.4	95.9	33.5	4.8	1.0
c)	Male	38	17	489	23	19.2	89.8	30.4	0.7	1.0
d)	Male	21	11	318	40	12.1	88.6	29.4	0.7	0.6

Table 7.8. Cutoff values.

	GOT	GPT	LDH	GGT	BUN	MCV	MCH	TbiL	CRT
1	40	40	350	60	13	85	30	1.0	0.8
2	100	100	500	100	18	95	32.5	2.0	1.2
3	200	200	700	300	25	105	35	3.0	1.5

The diagnoses are largely dependent on the doctor's experience. Therefore, the diagnostic rule is not necessarily represented by a single rule. We applied ADG to the diagnosis of hepatobiliary disorders. The number of groups in ADG corresponds to the number of extracted rules.

Rule 1(8 agents): $(GGT > 1) \wedge (BUN = 0) \wedge (MCV > 1) \wedge (MCH > 1)$

Rule 2(8 agents): $(GOT = 0) \wedge (GPT = 0) \wedge (GGT > 0) \wedge (MCH > 0) \wedge (CRTNN > 0)$

Rule 3(8 agents): $(LDH = 0) \wedge (GGT > 0) \wedge (MCV > 1) \wedge (TBIL = 0) \wedge (CRTNN > 0)$

Rule 4(5 agents): $(LDH = 0) \wedge (MCH > 0) \wedge (CRTNN = 0)$

Rule 5(4 agents): $(LDH = 0) \wedge (MCH = 3)$

Rule 6(4 agents): $(GGT = 3) \wedge (BUN < 2) \wedge (MCV = 0)$

Rule 7(4 agents): $(GPT = 0) \wedge (LDH = 0) \wedge (MCH = 0) \wedge (CRTNN = 2)$

Fig.7.8. The acquired rules by ADG.

We applied ADG to the diagnosis of alcoholic liver damage. As a result, 50 agents in the best individual are divided into 12 groups. Figure 7.8 shows a part of the acquired rules that corresponds to tree structural programs in the best individual. Rules are arranged according to the number of agents that support each rule. A rule with more agents means a frequently adopted rule. A rule with fewer agents means a rule for exceptional data.

In this section, we treat data containing multiple rules. We propose a new method using ADG to extract multiple rules. As a result, we extracted some effective rules for medical diagnosis.

7.4 Immune Multiagent Neural Networks

General NN learning aims to learn only good training cases, and it can realize high classification. However, we meet real databases, such as medical databases, where the records have noise or some contradictory cases. If we train the network with noise or some contradictory cases, the classification capability of the network will be reduced. Jacobs et al. proposed module nets, which can handle a subset of the complete set of training cases. Reflective NNs, proposed by Ichimura, are a kind of module nets [15].

The architecture of the proposed reflective NN is based on the network module concept shown in Fig. 7.9. There are two kinds of network modules: an

allocation module to distribute a training case and some classification modules to classify a set of training cases. Each classification module consists of a worker neural network (worker NN) and a monitor neural network (monitor NN). The monitor NN estimates how conformable the worker NN is to a given training case. The training cases are distributed over different classification modules iteratively according to the learning condition of a training case, until the sum of the squared errors reaches an expected value. We consider that each classification module competes with the others in the classification of training cases. Reflective NN has an outstanding classification capability, even if there are missing data or information, as is common in medical databases.

However, the optimal number of classification modules in reflective NNs is not determined according to the probability distribution function in the space of training cases, even if we have approached finding the optimal structure with structure level adaptation of the neural network [7]. To solve such a problem, we find some relation between the number of modules and the number of subsets of training cases by our proposed classification method of immune multiagent neural networks (IMANN) [1]. The IMANN has macrophage agents, T-cell agents, and B-cell agents. The macrophage and T-cell employ the planar lattice neural networks (PLNN) with the neuron generation/annihilation algorithm [2]. This network structure consists of hidden neurons in the lattice. The network can work in a manner similar to self-organized map (SOM) [3]. B-cell employs Darwinian neural networks (DNN) [4], which have a structural learning algorithm based on Darwin's theory of evolution.

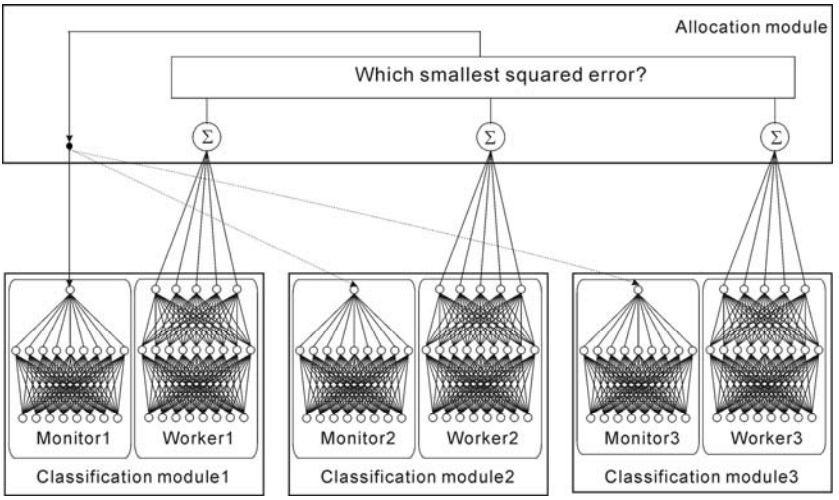


Fig.7.9. Reflective NNs.

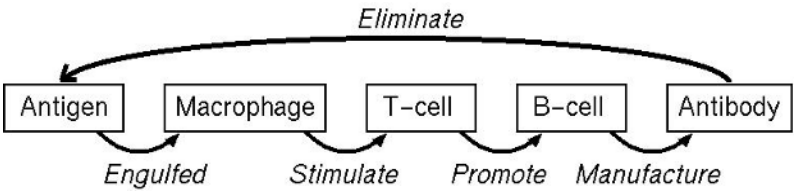


Fig.7.10. A model of the biological immune system.

7.4.1 Biological Immune System

The living body maintains normal condition by its biological immune system, where various immune cells perform their own functions and cooperate with each other. The biological immune system mainly works to protect a body from many antigens. Immune cells learn to recognize relevant patterns, they remember the patterns that have been seen previously, they fight with antigens using their patterns, and they promote robustness for unknown patterns by using their diversity. Figure 7.10 shows the relationship of immune cells.

The basic idea of our proposed IMANN is a cooperation and competition mechanism in a biological immune system. The macrophage can recognize unknown patterns that invade its living body at first. A T-cell receives the stimulation by macrophages and works to promote the B-cell's activities. And then B-cells manufacture their specified antibody. The IMANN is developed to imitate the characteristic functions of a biological immune system.

7.4.2 Planar Lattice Neural Networks

PLNN is a type of three-layered neural network, but the neurons in the hidden layer are arranged on a lattice. The network can work similar to SOM, that is, the patterns between inputs and outputs are classified into some groups in the lattice. Moreover, we can expect to extract some *If-Then* rules from the trained network.

Figure 7.11 shows an overview of PLNNs [2]. The neural network is a kind of layered neural network, consisting of an input layer, an interconnected hidden layer, and an output layer. The interconnected hidden neurons adjust the connection weights between input neurons and hidden neurons according to the relation of input-output patterns and the neighborhood N_i of the hidden neuron *i*. N_i is a set of neighboring neurons around i on the lattice. If S is the set of neurons and $N = \{N_i \mid i \in S\}$ is the set of neuron neighborhoods, then the pair $\{S, N\}$ forms a graph in the usual sense. The neighborhood system shown in Fig. 7.12 belongs to the class of homogeneous neighborhood system (HNS) defined as follows.

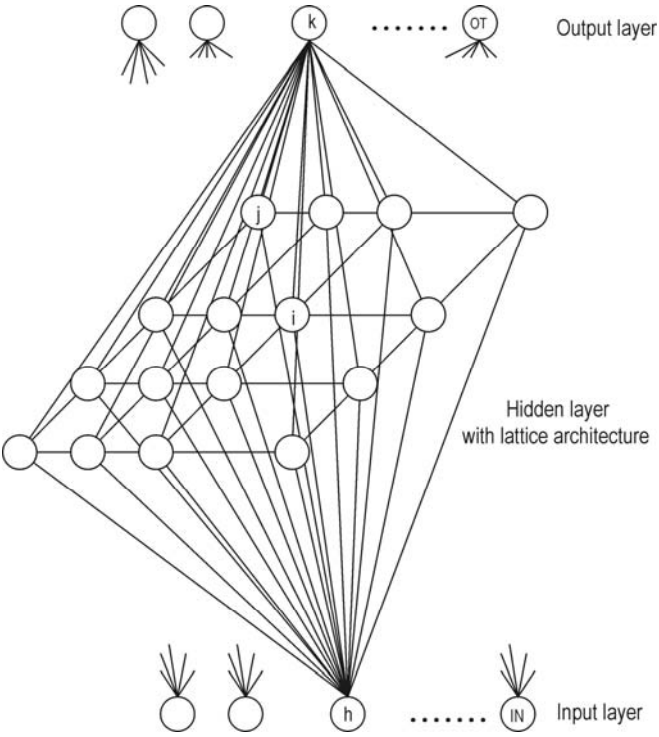


Fig.7.11. Planar lattice neural networks.

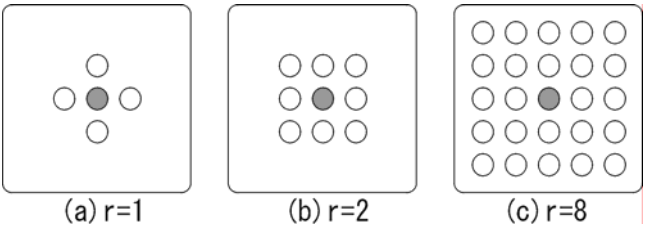


Fig.7.12. Homogeneous neighborhood system.

Definition 7.1. A homogeneous neighborhood system is of the form:

$$N = \{N_i \mid i \in S\}; N_i = \left\{ j \middle| \left\| \ell_j - \ell_i \right\|^2 \le r, i \in S \right\}, \tag{7.12}$$

where the ℓ_i and ℓ_j are the lattice positions for neuron *i* and *j*, respectively.

If the network has IN input neurons and OUT output neurons, there are $IN \times OUT$ hidden neurons. There are two function levels of variables (z_i, y_i) for

each neuron in the lattice; z_i is the presigmoidal activation level and y_i is the postsigmoidal activation level. Adjustment of the presigmoidal activation level is

$$z_i[t] = a_1 F\left(Met\left(x[t], w_i^{IH}\right)\right) + a_2 \sum_{j \in N_i} y_j[t-1]\mu_{ij}, \tag{7.13}$$

where $z_i[t]$ is the presigmoidal activation level for neuron *i* at time index *t*, the input vector of the network $\mathbf{x} = (x_1, x_2, \cdots x_h, \cdots, x_{IN})$, the input weight vector $\mathbf{w}_i^{IH} = \left(w_{i1}^{IH}, w_{i2}^{IH}, \cdots, w_{ih}^{IH}, \cdots, w_{iIN}^{IH}\right)$, and w_{ih}^{IH} is the weight associating neuron *i* with input signal x_h ($1 \le h \le IN$).

$F(\ )$ is a positive monotonic decreasing function, and $Met(\ )$is a metric that measures the distance between two vectors in a metric space. N_i is the neighborhood set of neuron *i*, and y_j is the output activity of neuron *j*. The lateral interconnection weights between neurons *i* and *j*, $\eta(\ )$, is the weight associating neuron i with neuron j; it depends only on the relative position in the lattice, $\mu_{ij} = \eta(\ell_i - \ell_j)$. $\eta(\ )$ is the spatial impulse response function of the network. a_1 and a_2 in Eq. (7.13) are two constants that are weights to inputs of the network and outputs from other neurons in the lattice, respectively. In this study, we used the Euclidean metric for $Met(\ )$. Then $F(\ )$ takes the following form:

$$F\left(Met\left(\mathbf{x}, \mathbf{w}_i^{IH}\right)\right) = f\left(\sum_{h=1}^{IN} g\left(w_{ih}^{IH}, x_h[t]\right)\right), \tag{7.14}$$

$$g\left(w_{ih}^{IH}, x_h[t]\right) = \left(w_{ih}^{IH} - x_h[t]\right)^2, \tag{7.15}$$

$$f(u) = e^{-\lambda u}. \tag{7.16}$$

The output activity of neuron z_i is

$$y_j[t] = \sigma_i(z_i[t]), \tag{7.17}$$

where $\sigma_i(\ )$ is the sigmoid function.

The weight adjustment follows the Kohonen learning algorithm [3]:

```
for (each time index t) do
    fetch source signal x[t]
    select i : ||x[t] − w_i[t−1]|| ≤ ||x[t] − w_k[t−1]||, ∀k ∈ S
    for (∀j ∈ N_i) do
        w_j[t] = w_j[t−1] + αφ[t−1](||ℓ_j − ℓ_i||)(x[t] − w_j[t−1])
    end
end
```

$x[t]$ is the input signal of the network at time index *t*. $\mathbf{w_j}[t]$ is the input weight

vector of neuron *j*. The learning rate α is a constant used to control the rate of convergence, and $\varphi[t](d)$ is a time-varying function used to control the influence region of a neuron to other neurons in its neighborhood. $\varphi[t](d)$ is

$$\varphi[t](d) = \begin{cases} 1 & \text{if } d \le R[t] \\ 0 & \text{otherwise} \end{cases} \tag{7.18}$$

where $R[t]$ is a positive definite monotonic decreasing function and the minimum value for $R[t]$ is 1 for the network to retain organizing capability.

The weights will be organized as follows: The neighboring neurons on the network tend to be similar to input weight vectors, and the weights represent neighboring regions in the input pattern space. In particular, the asymptotic values of the weight vectors will tend to be a weighted center of their influence regions. The influence region for neuron i is the region Ω_i in the input pattern space. During the training phase, whenever an input pattern falls in Ω_i, the $\mathbf{w}_i$ will be modified. That is, $\Omega_i = \bigcup_{j \in N_i} \mathbf{V}_j$, where $\mathbf{V}_j$ is the Voronoi region for neuron *j*.

We define the asymptotic values of the weight vector,

$$Asy_i[t](x) = \phi[t](\|\ell_i - C(x)\|), \tag{7.19}$$

where $C(x) = \ell_j$ is the lattice position for neuron j that has the input weight vector closest to input x. Then we obtained the following equation:

$$\hat{\mathbf{w}}_i = \lim_{t \to \infty} \mathbf{w}_i[t] = \frac{\int_{\Omega_i} p(x) Asy_i[t](x) x dx}{\int_{\Omega_i} p(x) Asy_i[t](x) dx}, \tag{7.20}$$

where $p(x)$ is a probability distribution function [25].

We follow the backpropagation algorithm to adjust the weight between hidden neurons and output neurons to minimize ε in Eq. (7.21):

$$\varepsilon = \frac{1}{2} \sum_{k=1}^{OT} (\hat{o}_k(x) - o_k(x))^2 \tag{7.21}$$

$$\Delta w_{ki}^{HO} = -\eta \frac{\partial \varepsilon}{\partial w_{ki}^{HO}} \tag{7.22}$$

where η is the learning rate.

The neurons in the lattice are added or eliminated by the generation/annihilation algorithm according to monitoring of the variance of weight vector described in Section 7.2.2.

For clear comprehension, we define $WD_{\max 1}$ as a neuron with the largest WD in the neighborhood N_i and $WD_{\max 2}$ as a neuron with the second largest *WD*. A new neuron is added in the middle of the two neurons $WD_{\max 1}$ and $WD_{\max 2}$. The new neuron should be inserted into the lattice, taking into account the arrangement of other neurons.

A typical neuron-generation situation is shown in Fig. 7.13. Figure 7.13(a) shows that the $WD_{\max 1}$ neuron is parallel to the $WD_{\max 2}$ neuron. In this case,

the new neuron is added as shown in the right panel of Fig. 7.13(a). Figure 7.13(b) shows that the $WD_{\max 1}$ neuron and the $WD_{\max 2}$ neuron are on a diagonal line. In this case, the new neuron is added as shown in the right panel of Fig. 7.13(b). On the other hand, if the neuron annihilation condition is satisfied, the specified neuron is eliminated as shown in Fig. 7.14.

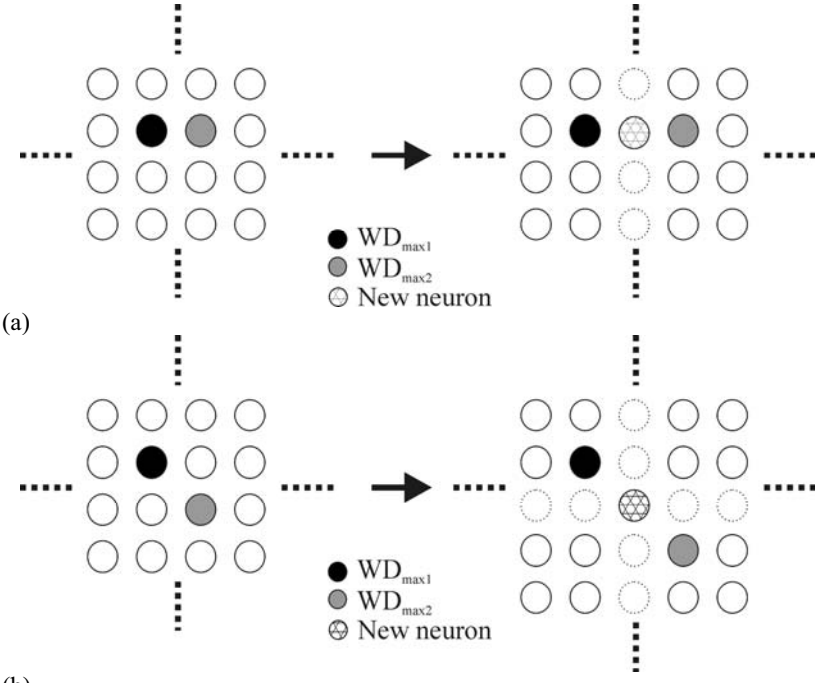


Fig.7.13. Neuron generation in PLNN (a) neuron generation case 1 and (b) neuron generation case 2.

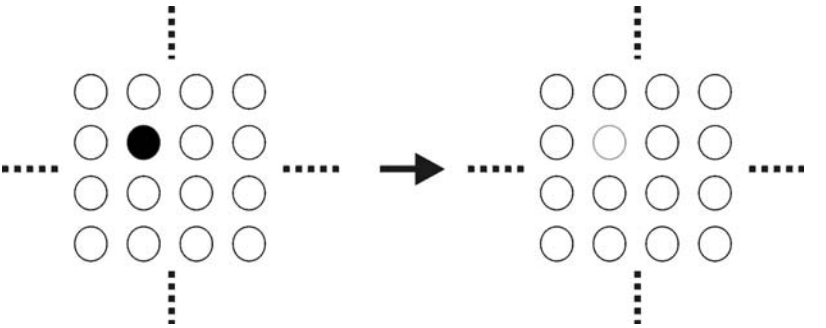


Fig.7.14. Neuron annihilation in PLNN.

7.4.3 Darwin Neural Network

In this section, we explain a structure adaptive learning algorithm for neural networks based on the theory of evolution [4]. Our proposed method imitates the process that living things adapt their structures according to the environment by evolution and learning. In this method, if the teaching data change during learning under dynamic environments, the learning does not restart from the initial state. This method is useful for adaptive learning, which can take into account inheritance of the network structure, the connection weight vectors, and the learning parameters.

From the theory of evolution, Sasaki and Tokoro compared two types of hereditary mechanisms: Lamarckian and Darwinian [16], [17]. There are two phases in each mechanism: the evolution phase over generations and the learning phase in the individual's lifetime. In [16] and [17], they regarded neural network's structure as the individual in population. BP learning is employed as the learning method and GA search is employed as the evolution method; the two algorithms work together to adapt their parameters under their environments. The Lamarckian hereditary mechanism in Fig. 7.15 inherits the trained connection weights by BP learning to the next generation. The Darwinian hereditary mechanism in Fig. 7.16 inherits only chromosomes from their parents. They reported that the performance of Darwinian is better than that of Lamarckian under their assumptions.

The evolution in the real world includes the change of length of chromosome, but their approach does not account for it. Also, some parameters in BP learning and GA search were determined by the designer. To improve this, we describe our proposed method in the next section.

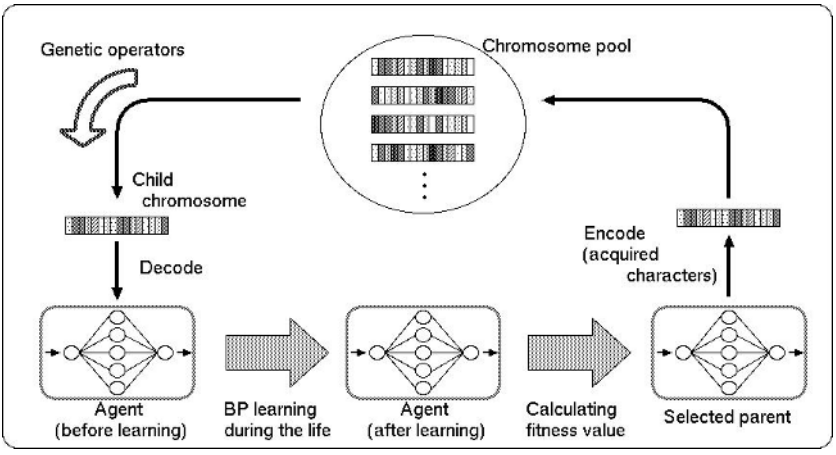


Fig.7.15. Hereditary mechanism of Lamarckian type.

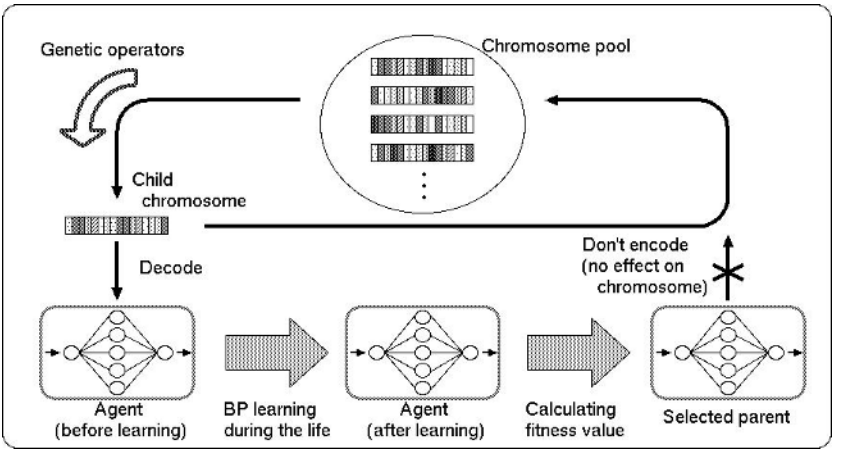


Fig.7.16. Hereditary mechanism of Darwinian type.

7.4.3.1 Adaptive Evolutionally Learning Method

In feedforward neural networks, when the input pattern p is set to input neurons, the output activity in the mth layer is given by the following equations:

$$o_{ip}^{m} = s\left(net_{ip}\right) \tag{7.23}$$

$$net_{ip} = \sum_{j} w_{ji} o_{jp}^{m-1}, \tag{7.24}$$

where w_{ji} is a connection weight from the jth neuron in the $m-1$ layer to the ith neuron in the mth layer and $s(x)$ is a sigmoid function as follows:

$$s(x) = \frac{1}{1+\exp(-\varepsilon x)}, \tag{7.25}$$

where ε is a constant.

The error is estimated by the sum of square error as follows:

$$E_p(\mathbf{W}) = \frac{1}{2}\sum_{k=1}^{K}\left(\bar{o}_{kp} - o_{kp}\right)^2 \tag{7.26}$$

$$E(\mathbf{W}) = \sum_{p} E_p(\mathbf{W}), \tag{7.27}$$

where $\bar{o}_{kp}$ is a teaching signal, o_{kp} is an output activity, K is the number of output neurons, and $\mathbf{W}$ denotes the vector of all connection weights.

The connection weight is adjusted by the following equation:

$$\Delta w_{ji}(t+1) = -\eta \frac{\partial E(\mathbf{W})}{\partial w_{ji}} o_{ip}^{m} + \alpha \Delta w_{ji}(t), \tag{7.28}$$

where t is the time index, α is the coefficient of momentum term, and η is the learning rate.

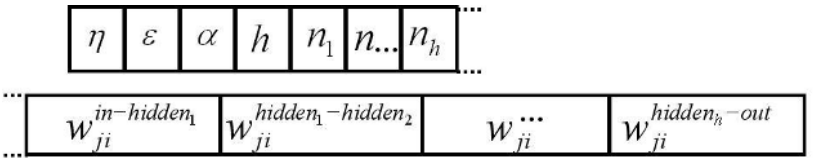


Fig.7.17. Genotype of neural networks.

7.4.3.2 Finding Network Structures by GA

In this section, we applied the method described in [18] to search for an optimal network structure by GA. We explain the genotype and genetic operators.

Genotype Representation

Figure 7.17 shows the chromosome with some learning parameters η, ε, α, and n in Table 7.9 and connection weights w_{ji} $(-1.0 \le w_{ji} \le 1.0)$. Here n is the number of hidden neurons.

Genetic Operators

GA search involves three kinds of genetic operators: selection, crossover, and mutation. Here the selection is implemented as a combination of elite strategy and roulette selection. The ratio of the chromosomes copied to the next generation by elite strategy is $P_e = 0.1$. That is, 10% of the chromosomes are selected using elitist strategy, and the remaining individuals are selected using roulette wheel selection. Successively two individuals are selected to make a crossover operation according to their fitness values. Crossover operation generates new offspring from the two selected individuals. Each gene representing a network is shown in Fig. 7.17. In this investigation, we employ the uniform crossover. For the mutation operation, [18] defined two types of mutation rate: local mutation P_{ml} and global mutation P_{mg}. The local mutation gives a small perturbation as shown in Table 7.10 to search in the neighborhood of local minima. On the other hand, the global mutation expands the search space by the parameters in Table 7.11.

Table 7.9. Learning parameters of a neural network.

η	$0 < \eta \le 1.0 \quad \eta \in \mathbf{R}$
ε	$0 < \varepsilon \le 1.0 \quad \varepsilon \in \mathbf{R}$
α	$0 < \alpha \le 1.0 \quad \alpha \in \mathbf{R}$
n	$2 \le n \le 20 \quad n \in \mathbf{N}$

Table 7.10. Perturbation by Local mutation around local minima.

$\{\eta, \varepsilon, \alpha\} \leftarrow \{\eta, \varepsilon, \alpha\} + r_1$	$-0.1 \le r_1 \le 0.1,\ r_1 \in R$
$n_k \leftarrow n_k + r_2$	$-5 \le r_2 \le 5,\ r_2 \in Z$
$h \leftarrow h + r_3$	$-1 \le r_3 \le 1,\ r_3 \in Z$
$w_{ji}^{s(s+1)} \leftarrow w_{ji}^{s(s+1)} + r_4$	$-1.0 \le r_4 \le 1.0,\ r_4 \in R$

Table 7.11. Expanding the search space by global mutation.

$\{\eta, \varepsilon, \alpha\} \leftarrow r_5$	$0.0 \le r_5 \le 1.0,\ r_5 \in R$
$n_k \leftarrow r_6$	$2 \le r_6 \le 20,\ r_6 \in N$
$h \leftarrow r_7$	$1 \le r_7 \le 3,\ r_7 \in N$
$w_{ji}^{s(s+1)} \leftarrow r_8$	$-5.0 \le r_8 \le 5.0,\ r_8 \in R$

Fitness Function with AIC

GA search intends to find a better network structure to fit the environment. Therefore, we should define a fitness function, which evaluates the error function and the network structure. We adopt the information criterion AIC [19] to evaluate the network structure.

The Network Evaluation with AIC [20]

AIC evaluates the goodness of fit of given models based on the mean square error for training data and the number of parameters as follows:

$$AIC = -2(\max\_\log\_likelihood) + 2F. \tag{7.29}$$

The F is the number of free parameters.

Let $\mathbf{e}_p = \overline{\mathbf{o}}_p - \mathbf{o}_p$ as the error for input pattern p; $\mathbf{o}_p$ is the output pattern for the input pattern of training case p, and $\overline{\mathbf{o}}_p$ is an average of $\mathbf{o}_p$. $\overline{\mathbf{o}}_p$ and $\mathbf{o}_p$ are normally distributed $N(0, \sigma^2\mathbf{I})$ and independent of each other. The likelihood of error for the training data is given by

$$L = \prod_{p=1}^{P} \left(2\pi\sigma^2\right)^{-\frac{K}{2}} \exp\left(-\frac{1}{2\sigma^2}\mathbf{e}_p^T\mathbf{e}_p\right) \tag{7.30}$$

The logarithm of Eq. (7.30) gives the following:

$$\log(L) = l = -\frac{KP}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\sum_{p=1}^{P}\mathbf{e}_p^T\mathbf{e}_p$$

$$= -\frac{KP}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}E(\mathbf{W}). \tag{7.31}$$

$E(\mathbf{W})$ can be minimized by BP learning based on the steepest gradient descent. As a result it enables us to obtain the maximum likelihood in Eq. (7.31).

Suppose the neural network has three layers: M input neurons, H hidden neurons, and K output neurons. This network has $H(M+K)$ connection weights and

H+K threshold values; we can set (H(M+K)+H+K) as F in Eq. (7.29). Therefore, Eq. (7.29) becomes Eq. (7.32).

$$AIC = -2(l) + 2(H(M + K) + H + K). \tag{7.32}$$

Fitness Function with AIC

To include not only error estimation but also the goodness of the network structure, we define the fitness function with AIC as follows:

$$Fitness(u) = \frac{\lambda_{\max} - AIC_u}{\lambda_{\max} - \lambda_{\min}}, \tag{7.33}$$

where $\lambda_{\max}$ is the maximum value of AIC and $\lambda_{\min}$ is the minimum value of AIC. The u is the index of an individual in the population.

7.4.4 Immune Cells by PLNNs

Macrophage employs the PLNN to classify the training cases. The hidden neurons are generated/annihilated during the learning phase, and consequently the remaining neurons are assigned to the corresponding subset of training cases, respectively. T-cell employs neural network learning to assign a training case into one of the B-cells. In this chapter, T-cell employs the lower part of PLNN and the network enforces learning reverse signals from output neurons as shown in Fig. 7.11. Because T-cell also recognizes input signals, T-cell trains the lower part of PLNN simultaneously. The teaching signals in the network consist of binary strings: 1 (on) and 0 (off). In biological immune systems, B-cells receive stimulation from T-cells. In our model, B-cells employ a simple DNN learning method [1] to train a network for the subset of training cases assigned by T-cell, as shown in Fig. 7.18. Although B-cells work to learn a subset of training cases independently, B-cells cooperate with each other in a classification task, as shown in Fig. 7.19.

Figure 7.20 shows the reasoning process of the trained IMANN. After training PLNN, an arbitrary input is given to T-cell NN. T-cell NN classifies into a group and stimulates the corresponding B-cell NN. The B-cell NN calculates output activities as a total output of IMANN.

7.4.5 ICU Database

To demonstrate the effectiveness of our scheme, we use the intensive care unit (ICU) database [23], [24]. The variables incorporated into the models were gender, age (<44, 45–64, 65–74, 75+), severity of illness (0–9, 10–19, 20–29, 30–39, 40–49, 50–59, 60+; predictive hospital mortality rate derived from APACHE II score [21]), operation (none, elective, urgent), ventilator (yes, no), urinary catheter (yes, no), central venous catheter (yes, no), and infection (none, drug-susceptive, drug-resistant). The signal of an output neuron was “0” or “1” representing “dead” or “alive.” Table 7.12 shows the variables in the ICU database. Figure 7.21 shows relationships between the variables in the ICU database.

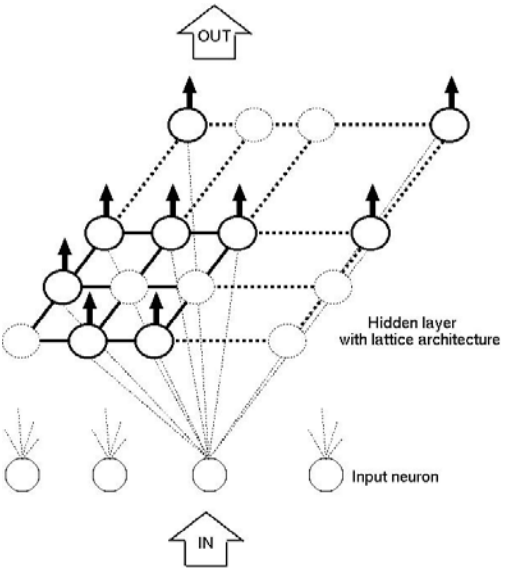


Fig.7.18. T-cell neural network.

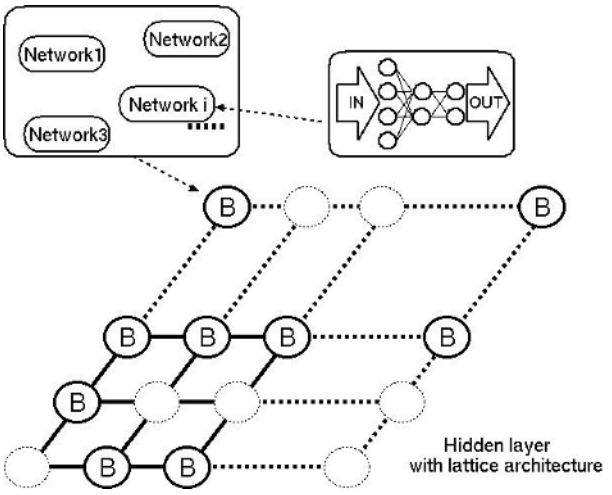


Fig.7.19. B-Cell neural network.

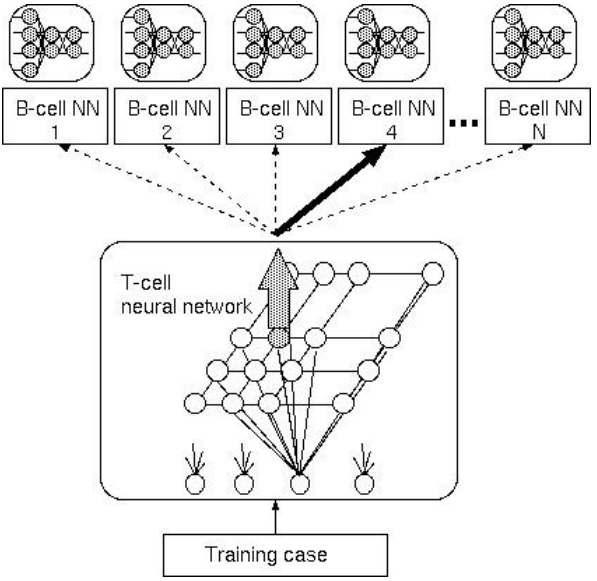


Fig.7.20. Reasoning process of the trained IMANN.

Table 7.12. Variables in the ICU database.

Gender		0:men, 1:women
Age(Categorized)		0:0 to 44, 1:45 to 54, 2:55 to 64, 3:65 to 74, 4:over 75
Predictive mortality rate (Categorized)		0:1 to 9, 1:10 to 19, 2:20 to 29, 3:30 to 39, 4:40 to 49, 5:50 to 59, 6:over 60
Operation	Elective operation	0:None, 1:Done
	Urgent operation	0:None, 1:Done
ICU-acquired infection	Drug-sensitive	0:None, 1:Done
	Drug-resistant	0:None, 1:Done
Respirator		0:None, 1:Done
Urinary catheter		0:None, 1:Done
Central venous catheter		0:None, 1:Done

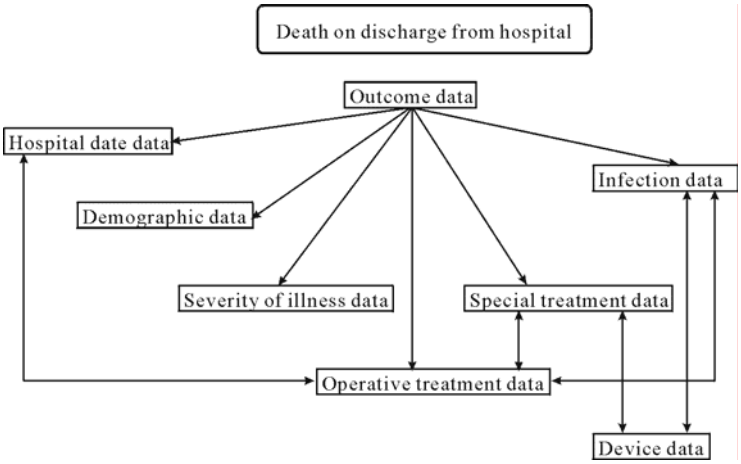


Fig.7.21. Relationships between variables in the ICU database.

We applied IMANNs to classify the ICU database. The PLNN has 15×15 squared neurons in a hidden layer after the neuron generation/annihilation, as shown in Fig. 7.22. A total of 61 neurons were classified into 20 subgroups in the lattice of T-cell NN. T-cell NN learns the relation between an input pattern and its allocated categories using two divided PLNN. B-cell NN trains the neural network for the 20 subgroups of training cases, respectively. The total correct rate on the test data was 91.7% (629/686). The correct rates for the 20 subgroups were 90.9%, 94.7%, 84.6%, 97.7%, 88.5%, 96.9%, 100.0%, 17.6%, 94.4%, 92.9%, 100.0%, 77.3%, 98.7%, 76.0%, 97.0%, 84.6%, 100.0%, 41.7%, 95.0%, and 90.0%, respectively. We found that the correct rate for the eighth subgroup was very low. Although the B-cell NN in this group were trained to output "alive," most of the assigned test cases in this group by the T-cell NN were "dead." To improve this low accuracy, the macrophage NN was trained until the squared error was very small.

We applied a search algorithm to the subspaces divided by PLNN (T-cell NN) and extracted *If-Then* rules from the trained IMANN without using an expert's explicit knowledge. After giving all possible patterns of input vector, the search algorithm may be able to extract not only the explicit knowledge, but also new unknown knowledge from the network as shown in Fig. 7.23 [22].

IMANN makes rough classification of training cases and checks whether the classification result is correct. T-cell NN determines one subgroup with the maximum output activity in the lattice of T-cell NN. This subgroup is considered to have typical characteristics of input-output patterns. Each subgroup is labeled according to the characteristics. For example, we may find that the characteristics of the output signal categorize the subgroup.

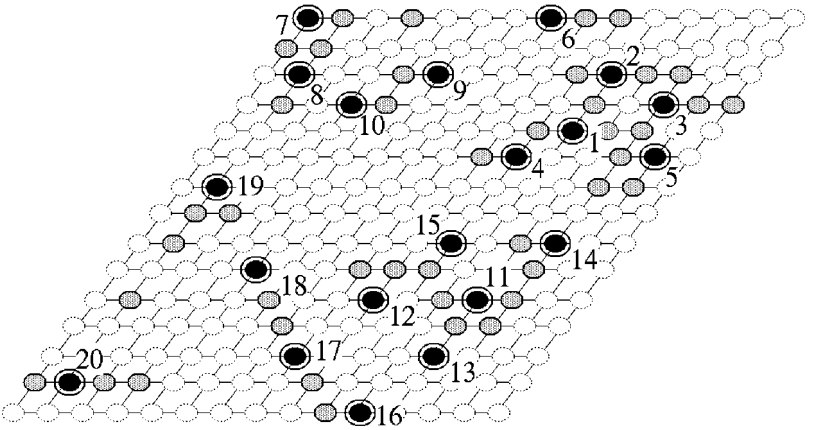


Fig.7.22. Neuron arrangements in a hidden layer.

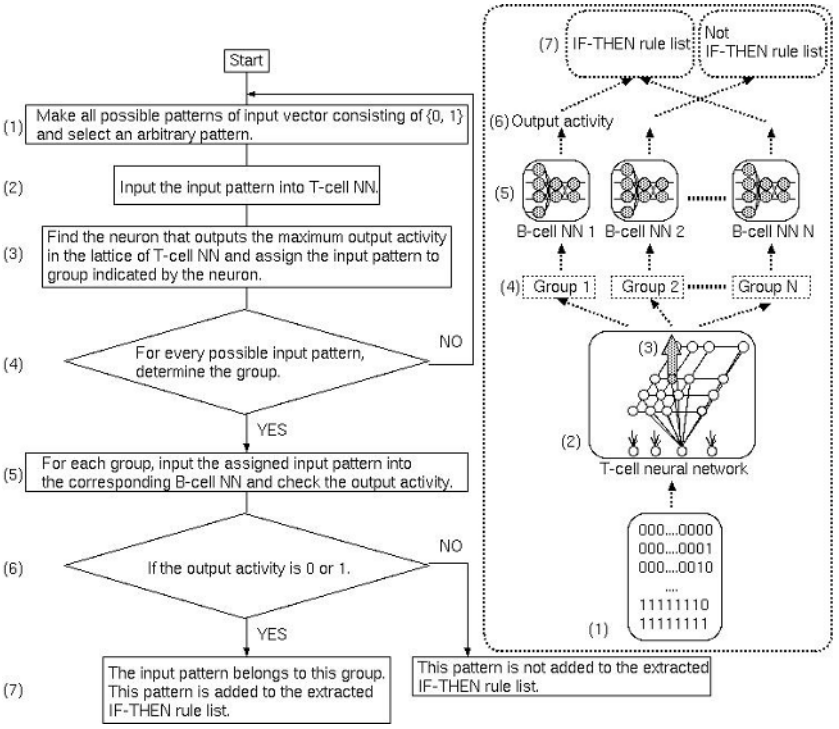


Fig.7.23. Flow of extracting *If-Then* rules from the trained IMANN.

We extracted *If-Then* rules from the trained IMANN of the ICU database. The output signal in the ICU database was "dead" or "alive" so that each subgroup in Fig. 7.23 outputted "dead" or "alive." The *If-Then* rules extracted from each subgroup were associated with the corresponding output signal. For example, the seventh subgroup outputted "dead." The *If-Then* rules extracted from the seventh subgroup by the search algorithm in Fig. 7.23 are:

- If "Ventilator" is true, then "dead" is very true.
- If "Urgent operation" is rather true, then "dead" is very true.
- If "Urgent operation" is rather true and "Ventilator" is true, then "dead" is very true.

7.5 Conclusion and Discussion

In this chapter, we described three methods of extracting *If-Then* rules from neural networks: KBANN with SLA, ADG, and IMANN. KBANN represents the knowledge structure of experts in a network structure. ADG combines multiagent system and GP techniques and extracts *If-Then* rules by generating cooperative behaviors of agents. IMANN combines the ideas of multiagent system and neural networks and makes two-stage classification by PLNN and DNN. All of the three methods were applied to medical databases, and they extracted *If-Then* rules from them successfully.

However, we may face the following dilemma. After giving all possible patterns of input vectors, the search algorithm may be able to extract not only explicit knowledge, but also new unknown knowledge from the network. However, the extracted rules may be contaminated by some meaningless rules. On the other hand, if we use an expert's explicit knowledge to prevent such contamination, we will acquire only ordinary knowledge from databases.

To develop effective data-mining methods for medical databases, we should begin by finding explicit knowledge from the network. After that, we should try to develop new techniques for extracting new unknown knowledge from databases. Our proposed methods will help develop effective data-mining methods for medical databases.

Acknowledgment

This research was performed with partial support of a Hiroshima City University Grant for Special Academic Research (General Studies) and supported by a Grant-in-Aid for Scientific Research (Grant-in-Aid for Scientific Research (B) 13470099, Grant-in-Aid for Exploratory Research 14657106, and Grant-in-Aid for Young Scientists 15790306) from the Japanese Ministry of Education, Culture, Sports, Science, and Technology and the Japanese Society for the Promotion of Science.

References

[1] Ichimura, T., Oeda, S., and Yoshida, K., A classification method of medical database by immune multi agent neural networks with planar lattice architecture, *Proc. 7th International Conference on Knowledge-Based Intelligent Engineering Systems & Allied Technologies* (KES2003), 2, pp. 380–7, 2003.

[2] Ichimura, T., Oeda, S., Yamashita, T., and Tazaki, E., A learning method of neural network with lattice architecture. *Journal of Japan Society for Fuzzy Theory*, 14:1, pp. 28–42, 2002.

[3] Kohonen, T., *Self-Organizing Maps*, Springer Series in Information Sciences, New York, 30, 1995.

[4] Oeda, S., Ichimura, T., Terauchi, M., Takahama, T., and Isomichi, Y., A synthesis of structural adaptive learning algorithm in neural network based on the theory of evolution, *Trans. IPS 2002*, 43:8, pp. 2728–38 (Japanese), 2002.

[5] Hara, A., and Nagao, T., Construction and analysis of stock market model using ADG; automatically defined groups. *International Journal of Computational Intelligence and Applications* (IJCIA), 2:4, pp. 433–46, 2002.

[6] Hara, A., Ichimura, T., Takahama, T., and Isomichi, Y., Extraction of rules by heterogeneous agents using automatically defined groups. *Proc. 7th Conference on Knowledge-Based Intelligent Information and Engineering Systems* (KES'2003), 2, pp. 1405–11, 2003.

[7] Ichimura, T., and Tazaki, E., Learning and extracting method for fuzzy rules using neural networks with modified structure level adaptation. *Proc. 2nd European Congress on Fuzzy and Intelligent Technologies* (EUFIT'94), 3, pp. 1237–41, 1994.

[8] Ichimura, T., Ooba, K., Tazaki, E., Takahashi, H., and Yoshida, K., Knowledge based approach to structure level adaptation of neural networks –Modeling of occurrence of hypertension, *IEEE Intl. Conf. on Systems, Man and Cybernetics* (SMC'97), 1, pp. 548–53, 1997.

[9] Towell, G. G., and Shavlik, J. W., Extracting refined rules from knowledge-based neural networks, *Machine Learning*, 13, pp. 71–101, 1993.

[10] Ichimura, T., *Studies on Learning and Reasoning Methods in Neural Network*, Ph.D. thesis, Toin University of Yokohama, 1997.

[11] Iba, H., Emergent cooperation for multiple agents using genetic programming, *Parallel Problem Solving from Nature IV*. Proc. International Conference on Evolutionary Computation, pp. 32–41, 1996.

[12] Iba, H., Multiple-agent learning for a robot navigation task by genetic programming, *Genetic Programming 1997: Proc. 2nd Annual Conference*, pp. 195–200, 1997.

[13] Soule, T., Voting teams: A cooperative approach to non-typical problems using genetic programming, *Proc. Genetic and Evolutionary Computation Conference*, pp. 916–22, 1999.

[14] Soule, T., Heterogeneity and specialization in evolving teams, *Proc. Genetic and Evolutionary Computation Conference*, pp. 778–85, 2000.

[15] Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E., Adaptive

mixture of local experts, *Neural Computation*, 3, pp. 79–87, 1991.
[16] Sasaki, T., and Tokoro, M., Adaptation toward changing environments: Why Darwinian in nature? *Proc. 4th European Conference on Artificial Life*, (ECAL-97), pp. 145–53, 1997.
[17] Sasaki, T., and Tokoro, M., Adaptation under changing environments with various rates of inheritance of acquired characters: Comparison between Darwinian and Lamarckian evolution, *Proc. 2nd Asia-Pacific Conf. on Simulated Evolution and Learning*, (SEAL-98), pp. 145–53, 1998.
[18] Takahashi, H., and Nakajima, M., Evolutional design and training algorithm for feedforward neural networks, *IEICE Trans. Information & Systems*, E82-D:10, pp. 1384–92, 1999.
[19] Akaike, H., A new look at the statistical model identification, *IEEE Trans. on Automatic Control*, AC-19:6, pp. 716–23, 1974.
[20] Kurita, T., and Motomura, Y. Feed-forward neural networks and their related topics, *Japanese Society of Applied Statistics*, 22:3, pp. 99–115, 1993.
[21] Knaus, W. A., Draper, E. A., Wagner, D. P., and Zimmerman, J. E., APACHE II: A severity of disease classification system, *Crit Care Med*, 13, pp. 818–29, 1985.
[22] Oeda, S., *A Proposal of Immune Multi-Agent Neural Networks*, Ph.D, thesis. Tokyo Metropolitan Institute of Technology, 2003.
[23] *The 2003 Report of Health and Labour Sciences Research Grant* (Research on Emergent or Revival Infections), Research on network of drug-resistant infection surveillance, 2003.
[24] Suka, M., Yoshida, K., and Takezawa, J., Impact of intensive care unit-acquired infection on hospital mortality in Japan: A multicenter cohort study, *Environ Health Prev Med*, 9, pp. 53–7, 2004.
[25] Ichimura, T., *Studies on Learning and Reasoning Methods in Neural Networks*, Ph.D. thesis, Toin University of Yokohama, 1997.

8. Satellite Image Classification Using Cascaded Architecture of Neural Fuzzy Network

Chin-Teng Lin, Her-Chang Pu, and Yin-Cheung Lee

Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu, Taiwan, R.O.C.

Because satellite images usually contain many complex factors and mix-up samples, a high recognition rate is not easy to attain. Especially for a nonhomogeneous region, the gray values of its satellite image vary greatly, and thus the direct use of gray values cannot do the categorization task correctly. Classification of terrain cover using polarimetric radar is an area of considerable current interest and research. Without the benefit of satellite, we cannot analyze the information of the distribution of soils and cities for a land development, as well as the variation of clouds and volcano for weather forecasting and for precaution, respectively. This chapter discusses a hybrid neural fuzzy network, combining unsupervised and supervised learning, for designing classifier systems. Based on systematic feature analysis, which is crucial for data mining and knowledge extraction, the proposed scheme signifies a novel algebraic system identification method, which can be used for knowledge extraction in general, and for satellite image analysis in particular. The goal of this chapter is to develop a cascaded architecture of a neural fuzzy network with feature mapping (CNFM) to help the classification of satellite images.

8.1 Introduction

This chapter discusses a hybrid neural fuzzy network combining unsupervised and supervised learning for designing classifier systems. Based on systematic feature analysis, which is crucial for data mining and knowledge extraction [1], [2], [3], [4], the proposed scheme signifies a novel algebraic system-identification method, which can be used for knowledge extraction in general and for satellite image analysis in particular.

Classification of terrain cover using polarimetric radar is an area of considerable current interest and research. Without the benefit of satellite, we cannot analyze the information of the distribution of soils and cities for land development, as well as the variation of clouds and volcano for weather forecasting and precaution, respectively. However, one cannot talk about these applications without mentioning the classification. Our motivation is to build up a system that can assist us in analyzing and classifying the information from satellite images automatically.

Early investigations for satellite-image classification have employed autocorrelation functions [5], power spectra, relative frequencies of various gray levels on the unnormalized image [6], and the second-order gray-level statistics method [7] to obtain texture features. These applications should be extended to proceed the

classification with arbitrary patterns, not just targets of selected blocks with different textures. Others have applied the Bayesian classifier [8] and Markov random field [9], [10] to obtain relative frequencies of individual and neighbors among a pixel. With these structures it is hard to obtain the required results when samples are insufficient. Moreover, the consumption of time to do the classification should also be one of the considerations. Some people use neural networks [11], [12], [13] to proceed the classification. Their results show that neural networks are likely to get a satisfying result and this is why the neural network is used as our reference. Because the satellite images usually contain many complex factors and mix-up samples, a high recognition rate is not easy to attain. Especially for a nonhomogeneous region, the gray values of its satellite image vary greatly, and thus the direct use of gray-level statistics fails to do the categorization task satisfactorily. The goal of this chapter is to develop a cascaded architecture of a neural fuzzy network with feature mapping (CNFM) to help the clustering of satellite images.

In this chapter, the dimension of inputs is first reduced by Kohonen's self-organizing feature map (SOM). It is an unsupervised neural network whose inputs of each channel are composed as follows. First, gray values are selected as the reference values. Second, statistical features computed from co-occurrence matrices [7] such as contrast, inverse difference moment, angular second moment, and entropy are used. Third, energies and entropies from wavelet decomposition [16], [17] are served as spectral features. No matter how many features and how many channels we use, each group of features in high dimension can be transformed into 2D coordinates by Kohonen's SOM. In addition to the benefit of reduction in dimension, it can remove some noisy areas and avoid the training process being overoriented to the training patterns. After the inputs are transformed by Kohonen's SOM, further classification will be performed by a neural fuzzy-network (called SONFIN [14]). It is a supervised neural network that can classify desired outputs delicately. This cascaded architecture, named CNFM, is a general and powerful structure that can give very promising results in terms of accuracy and performance. Experimental results show that the CNFM can reach an accuracy of 96.5% with respect to all feature domains.

Figure 8.1 shows the system architecture of CNFM. There are three types of input, which are spatial features of gray values, statistical features from an occurrence matrix, and spectral features from wavelet decomposition with N channels. Suppose there are M features in total. If we do not reduce our dimension of inputs, our network inputs will be of $(M * N)$ dimension. It must be noted that if the number of features increases, the input space of the multichannel satellite-image classification problem grows. However, our system can solve this problem gracefully; the input dimension is first reduced by Kohonen's SOM, and further classification is performed by a neural fuzzy network (SONFIN). Figure 8.2 shows the details of our CNFM. Given a center pixel, we select different neighborhood sizes for different feature domains. We use co-occurrence matrix and wavelet decomposition to extract the required features. After that, we pass the features, such as gray values, angular second moment (ASM), inverse difference moment (IDM), contrast (CON), entropies (ENT) and energies to Kohonen's SOMs. Finally, we use a neural fuzzy network, SONFIN, to train the outputs, in 2D coordinate format, of

Kohonen's SOMs. In the next section, the acquisition, implementation, and modification of the three groups of inputs are given. In Section 8.3, the integration of CNFM by cascading Kohonen's SOM and SONFIN is modeled. Experimental results are shown in Section 8.4. Finally, some conclusions are reached in Section 8.5.

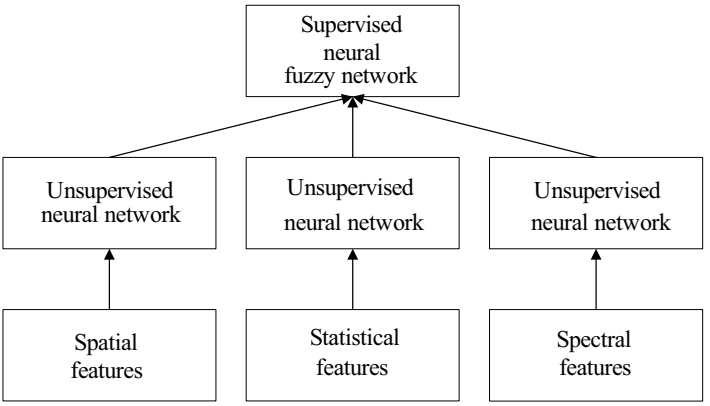


Fig. 8.1. System architecture of CNFM.

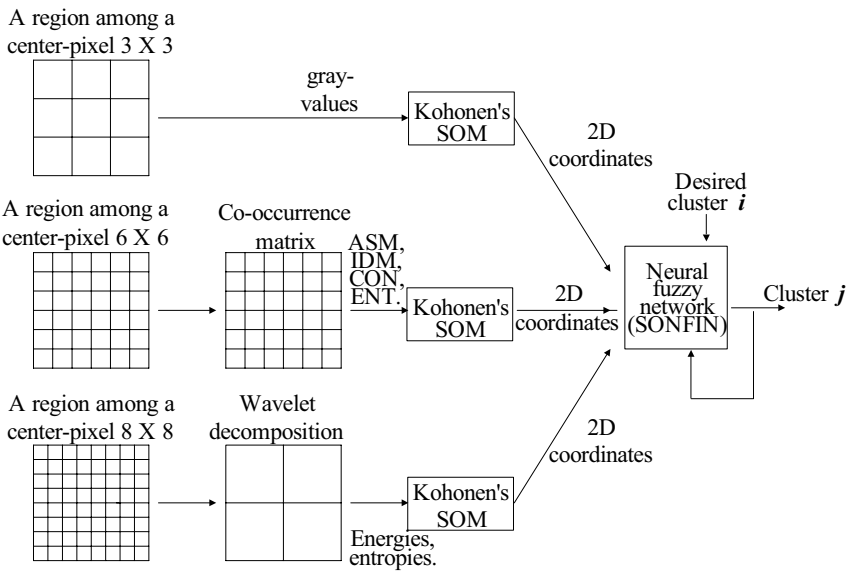


Fig. 8.2. A detailed view of CNFM.

8.2 Input Acquisition

The objective of this section is to clarify how the input features are chosen and what they are actually for. These input features are gray values, statistical features, and spectral features. The varieties of gray values among channels are characteristics that can be used to classify the ground cover type from data of multispectral spaces into desired clusters. However, due to the complexities, mix-up samples, and textural problems, we have to use statistical features from the co-occurrence matrix and spectral features from wavelet decomposition to improve classification accuracy. The following sections show how our input acquisition is implemented and modified to get a better performance.

8.2.1 Features from the Spatial Domain by Multispectral Data

Taking advantage of multispectral data obtained from different sensors, we can classify the data as required. Here we choose the gray values as our spatial features. Because the variation of gray values among channels can serve as discriminating features, they can be used to classify the ground cover type in these multispectral spaces into desired clusters. In details, the groups or clusters of pixel points are referred to as information classes because they are the actual classes of data that a computer can recognize.

8.2.2 Features from the Statistical Domain

8.2.2.1 Co-occurrence Matrix

Gray-level co-occurrence matrices constitute one of the basic approaches to the statistical analysis of texture. By computing a set of gray-tone spatial-dependence probability-distribution matrices for a given image block and extracting a set of textural features from each of these matrices, the basic model attempts to take the variation as a function of the direction of spatial distance.

Here is the second-order histogram: $S = f(i, j, d, \theta)$, where i, j are gray values of pixels distance d apart and θ is the angle (usually every 45°) of the line joining the centers of these pixels with the horizontal axis. These matrices are symmetric: $P(\theta) = P(\theta + \pi)$, that is, $P(i, j, d, \theta) = P(j, i, d, \theta)$. From this matrix, a number of features can be computed. With G gray levels in the image, the dimension of the matrix will be $G \times G$. The (i, j)th element, P_{ij}, of this matrix is defined by

$$P_{ij} = \frac{f_{ij}^{d,\vartheta}}{\sum_{i}^{N}\sum_{j}^{N} f_{ij}^{d,\vartheta}}, \tag{8.1}$$

where $f_{ij}^{d,\theta}$ is the frequency of occurrence of gray levels i and j, separated by a distance d and direction $\theta = 0^\circ, 45^\circ, 90^\circ, 135^\circ$. The summation is over the total number of pixel pairs N, given d, in the window.

We shall compute the following texture features from the co-occurrence matrix: angular second moment (ASM), contrast (CON), inverse difference moment (IDM), and entropy (ENT). These features are among the most commonly used co-occurrence features in Section 8.2.2.4.

8.2.2.2 Improvement in Performance by Symmetric Linked List (SLL)

Because the number of operations required to compute any of the aforementioned features is proportional to the number of resolution cells (gray values being used) in the image block, co-occurrence matrices are time-consuming to compute and are memory-intensive as well. For a typical gray-valued image, each co-occurrence matrix computed from the image is a $G \times G$ matrix (G = 256). However, assuming that the window size is N and the gray level is G, then at most there are $N(N-1)\times 2$ matrix entries. So the redundancy computation is a factor of $G \times G - N(N-1)\times 2$. To overcome this problem, the equal probability quantizing (EPQ) algorithm to reduce G is preferable. Although quantization can remove the contrast sensitivity of textures, it will also remove the first-order differences in the images. Also, it deeply destroys the relation of multisources, especially input sources from multichannels. Furthermore, selection of the gray level is required. The example in Fig. 8.3 shows three selected blocks with the highest levels of gray values, so they will remain and the other gray values will be removed to reduce the dimension of co-occurrence matrix.

To take advantage of the characteristic of overlapped windows and symmetric property, we can construct a symmetric linked list (SLL). Each node in SLL consists of data i, data j, and a counter. Data i and j are indexes of the co-occurrence matrix, and counter is used to count the frequency of occurrence of gray levels i and j. To increase the efficiency, each entry in SLL should be sorted. Because the co-occurrence matrix is symmetric, the memory store can be reduced at least by half. Furthermore, as we can see, updating performance will be improved.

The update procedure is as follows. First, we construct a basic co-occurrence matrix at the top-left corner of the image, and we compute the required features from this matrix. Then we subtract elements from the last column. Second, we move the window one column to the right. At the same time, we add elements from the new inserted column and compute the required features. Repeat this process column by column. When the window reaches the end of the row, we slide

down the next row. Then the window will move in a zigzag pattern until the entire image has been covered. As an illustrative example (Fig. 8.4), the first block in the top-left corner is the first co-occurrence matrix. Then the label S on this block is the last column to be subtracted while the label A on the next matrix is to be added. We repeat this process column by column and row by row in a zigzag pattern.

Because the data are coded in zigzag format, they should be regularized after the process finished. Although it is not easy to design the SLL, it is the fastest method to calculate the co-occurrence matrix with full color range. Moreover, this update method can be applied to other applications.

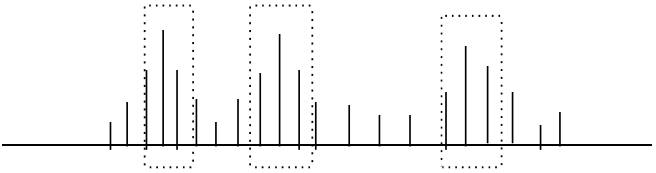


Fig. 8.3. The equal probability quantizing (EPQ) algorithm.

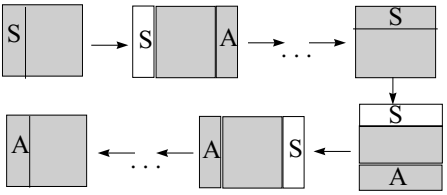


Fig. 8.4. The symmetric linked list (SLL) algorithm.

8.2.2.3 Global Texture Features from Second-Order Histogram Statistics

From the co-occurrence matrix, we can define the features as follows:

- Entropy (ENT). The entropy computed from the second-order histogram provides a standard measurement of homogeneity and is defined as

$$Entropy = -\sum_i \sum_j p(i, j) \log(p(i, j)). \tag{8.2}$$

  Higher values of homogeneity will indicate fewer structural variations whereas lower values will be interpreted as a higher probability of textural region.
- Contrast (CON). The contrast feature is a difference moment of the P matrix and a standard measurement of the number of local variations presented in an image. The contrast feature on the second-order histogram is defined as

$$Contrast = \sum_i \sum_j (i,j)^2 \, p(i,j). \tag{8.3}$$

The higher values of the contrast are, the sharper the structural variations in the image.

- Angular Second Moment (ASM). The angular second moment gives a strong measurement of uniformity and can be defined as

$$Angular\ Second\ Monent = \sum_i \sum_j \{p(i,j)\}^2. \tag{8.4}$$

Higher nonuniformity values provide evidence of higher structural variations.

- Inverse Difference Moment (IDM). The inverse difference moment is a measure of local homogeneity and is defined as

$$Inverse\ Difference\ Monent = \sum_i \sum_j \frac{1}{1+(i-j)^2 / G^2} \, p(i,j). \tag{8.5}$$

Features such as correlation cannot be used. If the variance becomes zero, the correlation will go to infinity.

8.2.3 The Complete Procedure of Wavelet Decomposition

Wavelet decomposition [17] is a mathematical framework for constructing an orthonormal basis for the space of all finite energy signals. It can decompose input signals into multiscale details, describing their power at each scale and position. It can discriminate the locate properties corresponding to smooth and textured areas. We will introduce the framework, implementation, and complete procedure of wavelet decomposition in this section.

At each level, the coefficients *{h(n)}* and *{g(n)}* are squared, then summed, and the square root is taken to generate a single feature for each approximation and detail of that level. These sets of four numbers, i.e., the scaling function and the wavelets for each level, are then used for classification. First, we perform one step of horizontal pairwise averaging and differing on the pixel value in each row of the image. Next, we apply vertical pairwise averaging and differing to each column of the result. To complete the transformation, we repeat this process recursively only on the quadrant containing averages in both directions.

The complete procedure is shown in Fig. 8.5. First we select a point from the top-left of the image and obtain a block from the gray values of its neighbors. Second, we perform the wavelet decomposition row by row, including cases of low pass and high pass. Finally, we apply the same process column by column. This is the 2D wavelet decomposition. For the wavelet decomposition, we first extend the size of the selected block to be a square block. Then we perform the convolution and then downsampling the size of the block into half. Then we transpose the image for the next wavelet decomposition. As an example shown in Fig. 8.6, the top-left and bottom-right images are the result of wavelet decomposition from

tion from both low pass and high pass in columns and in rows, respectively. The top-right image is from low pass in columns and high pass in rows. Conversely, the bottom- left image is from high pass in columns and low pass in rows.

The complete discrete wavelet decomposition is obtained by cascading the outputs of the low-pass filter *{h(n)}* into the same filter bank as shown in the upper part of Fig. 8.7. The outputs of each filter are critically sampled. The decomposition results in a fine-to-coarse representation of the input signal. The scaling coefficients at a given scale are a low-pass-filtered and contracted version of the scaling coefficients at the previous scale. The wavelet coefficients at a given scale represent the different detail information needed to reconstruct the signal at the previous finer scale. The scaling function $\phi_i(x)$ is obtained from *{h(n)}* only while wavelets are obtained from *{h(n)}* and *{g(n)}*, where $\psi_{k,i}(x)$ is the wavelet with input *i* and the output at level *k*.

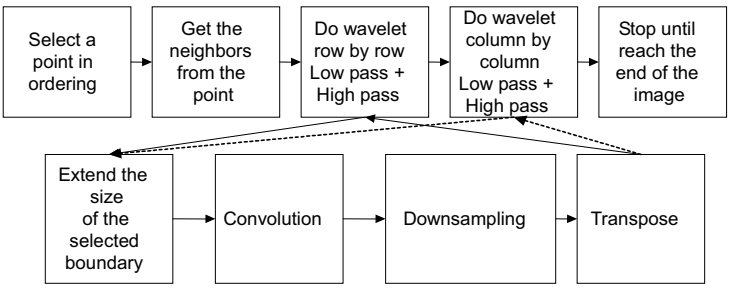


Fig. 8.5. Procedure of wavelet decomposition.

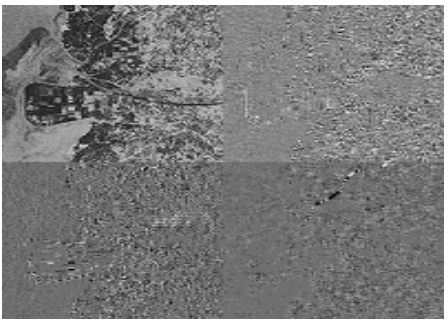


Fig. 8.6. Result of wavelet decomposition of a channel input.

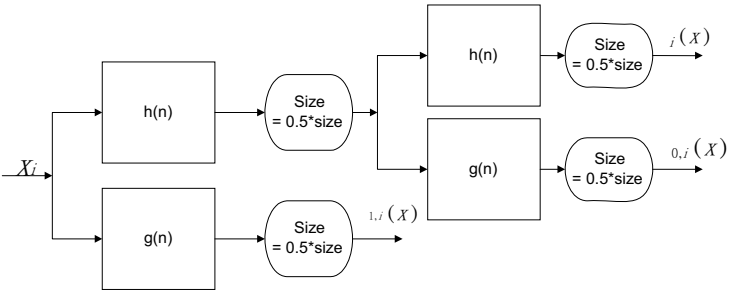


Fig. 8.7. Procedure of recursive wavelet decomposition.

8.3 A Cascaded Architecture of a Neural Fuzzy Network with Feature Mapping (CNFM)

After we obtain the features from the spatial, statistical, and spectral domains, we shall proceed to train the CNFM. It is composed of two cascaded neural networks. The former is the unsupervised Kohonen's SOM, and the latter is the supervised neural fuzzy network (called SONFIN). This complementary architecture can compensate for the problems of inaccuracy and long training time of unsupervised and supervised neural networks, respectively. In more detail, the most important contribution of this architecture is to improve the method of input selection. Compared to the conventional trial-and-error method, the input dimension of our system is first reduced by Kohonen's SOM, and then each group of features from a channel is transformed into 2D coordinates. Next, we use SONFIN to overcome the inaccuracy due to Kohonen's SOM. Hence, no matter how many features and how many channels we use, the problems of big input space and long training time can be removed by the proposed mechanism.

8.3.1 Reduction of Input Dimension by Unsupervised Network

This subsection introduces Kohonen's self-organizing map (SOM), which can map high-dimensional inputs into a 2D map and filter out some noisy information. We shall apply conscience to Kohonen's SOM to normally distribute the input clusters. With the benefits of Kohonen's SOM, the system can easily be adapted to the changes (increase) in both features and channels. Also, it can filter out some noisy information. The most important merit is that we can avoid trial and error to remove redundancy, such as by genetic algorithm, KL expansion, and correlation $(x_i^T y) / [(x_i^T x)(y_i^T y)]$.

8.3.1.1 Kohonen's Self-Organizing Map (SOM)

The self-organizing neural network, also called the topology-preserving map, assumes a topological structure among the cluster units. This property is observed in the brain but is not found in artificial neural networks. There are *m* cluster units, arranged in a 1D or 2D array; the input signals are *n*-tuples.

The weight vector for a cluster unit serves as an exemplar of the input patterns associated with that cluster. During the self-organization process, the cluster unit whose weight vector matches the input pattern most closely (typically, in terms of the minimum Euclidean distance) is chosen as the winner. The winning unit and its neighboring units (in terms of the topology of the cluster units) update their weights. The problem of mistuning initial weights and slow convergence may be caused by the winner-take-all strategy or if the same learning step is used for all neighbors. This problem can be solved by applying the Mexican hat structure, which can excite the cooperative neighbors in close proximity and inhibit the competitive neighbors that are somewhat further away with a Gaussian-like function.

8.3.1.2 Applying Conscience to Kohonen's SOM

Inspection of Fig. 8.8 may suggest that the number of clusters generated in Class 1 will either be larger than or at least equal to that in Class 2. Although this seems reasonable because the number of individuals in Class 1 is larger than that in Class 2, we get an opposite result using Kohonen's normal SOM.

To make an unsupervised neural network utilize the information about the distribution of patterns, the patterns must be divided into *m* clusters according to the occurrences of clusters. Therefore, clusters will be sufficient in a high-density region, whereas the number of clusters will be reduced in a sparse region.

We apply this conscience [18] to the update of Kohonen's SOM. After selection of the winner, i.e.,

$$y_i = \begin{cases} 1 & \text{if } \left\| X(t) - W_j(t) \right\| = \min_{l=1}^{m} \left\| X(t) - W_j(t) \right\| \\ 0 & \text{otherwise} \end{cases} \tag{8.6}$$

where y_j is the winner, $X(t)$ are the current inputs, $W_j(t)$ are the weights (cluster *j*), instead of updating the weights as usual, we record the usage of each cluster, i.e.,

$$p_j^{\text{new}} = p_j^{\text{old}} + \beta(y_j - p_j^{\text{old}}), \quad j = 1,...,m, \tag{8.7}$$

where p_j is the record of usage and β is a step size.

After the statistics are known, selection of the winner is done as

$$y_i = \begin{cases} 1 & \text{if } \left\| X(t) - W_j(t) \right\| - b_j = \min_{l=1}^{m} \left\| X(t) - W_l(t) \right\| - b_l \\ 0 & \text{otherwise} \end{cases} \tag{8.8}$$

where $b_j = \gamma(\frac{1}{m} - p_j^{\text{new}})$ is an offset and γ is a constant.

As the usage p_j of cluster j increases, its offset b_j will decrease. This reduces its competitive ability. In other words, its conscience is increased and persuaded to give opportunity to other clusters.

Fig. 8.8. The problem caused by distributions with different variances.

8.3.2 Classification by Supervised Network

After the transformation of input space using Kohonen's SOM has been completed, we pass the new composed inputs into a supervised neural network for its classification. This cascaded architecture has the ability to perform the classification of satellite images even if there are very complex mixed-up samples.

The neural fuzzy network that we used for satellite image classification is called the self-constructing neural fuzzy inference network (SONFIN), which we proposed in [14]. The SONFIN is a general connectionist model of a fuzzy logic system, which can find its optimal structure and parameters automatically. There are no rules initially in the SONFIN. They are created and adapted as online learning proceeds via simultaneous structure and parameter learning, so the SONFIN can be used for normal operation any time as learning proceeds without any assignment of fuzzy rules in advance. Of course, available fuzzy rules can be put into the network to speed up its learning. A novel network construction method for solving the dilemma between the number of rules and the number of consequent terms is developed. The number of generated rules and membership functions is small, even for modeling a sophisticated system. The SONFIN can always find itself an economic network size, and the learning speed and modeling ability are appreciated compared to normal neural networks.

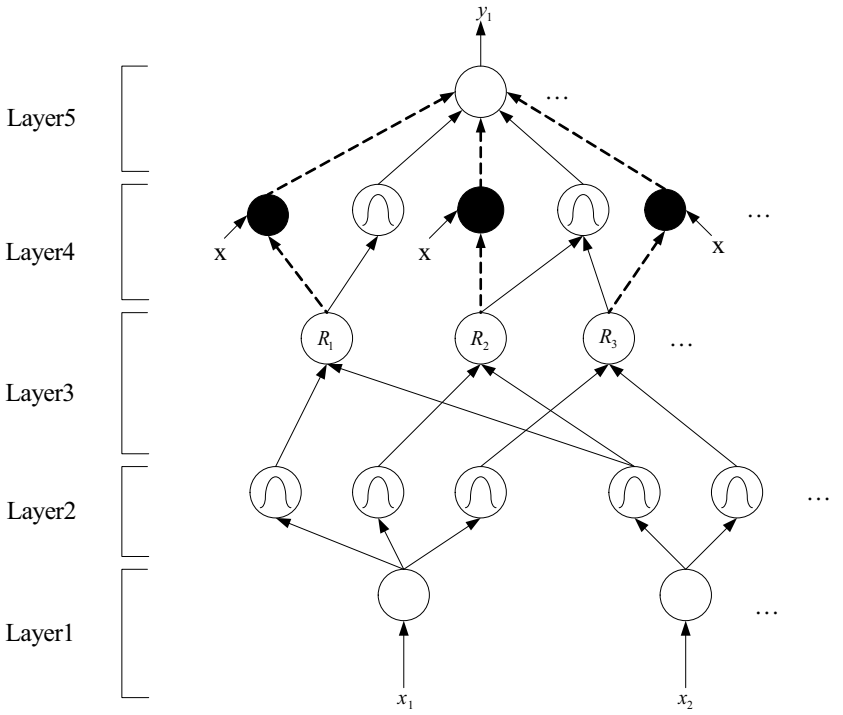


Fig. 8.9. Structure of self-constructing neural fuzzy inference network (SONFIN).

The structure of the SONFIN is shown in Fig. 8.9. The five-layered network realizes a fuzzy model of the following form

Rule i : IF x_1 is A_{i1} and … and x_n is A_{in}
THEN y is $m_{oi} + a_{ji}x_j + \ldots$,

where A_{ij} is a fuzzy set, m_{oi} is the center of a symmetric membership function on y, and a_{ji} is a consequent parameter. Unlike the traditional TSK model where all the input variables are used in the output linear equation, only the significant ones are used in the SONFIN; i.e., some a_{ij}'s in the fuzzy rules are zero. We next describe the functions of the nodes in each of the five layers of the SONFIN.

Layer 1: No computation is done in this layer. Each node in this layer, which corresponds to one input variable, only transmits input values to the next layer directly. That is,

$$a^{(1)} = u_i^{(1)} = x_i \,. \tag{8.9}$$

Layer 2: Each node in this layer corresponds to one linguistic label (small, large, etc.) of one of the input variables in Layer 1. In other words, the membership value that specifies the degree to which an input value belongs to a fuzzy set

is calculated in Layer 2. With the use of the Gaussian membership function, the operations performed in this layer are

$$a^{(2)} = e^{-\frac{(u_{ij}^{(2)} - m_{ij})^2}{\sigma_{ij}^2}}, \tag{8.10}$$

where m_{ij} and σ_{ij} are, respectively, the center (or mean) and the width (or variance) of the Gaussian membership function of the *j*th term of the *i*th input variable x_i. Unlike other clustering-based partitioning methods, where each input variable has the same number of fuzzy sets, the number of fuzzy sets of each input variable is not necessarily identical in the SONFIN.

Layer 3: A node in this layer represents one fuzzy logic rule and performs precondition matching of a rule. Here, we use the following AND operation for each Layer-3 node,

$$a^{(3)} = \prod_i u_i^{(3)}, \tag{8.11}$$

where *n* is the number of Layer-2 nodes participating in the IF part of the rule.

Layer 4: This layer is called the consequent layer. Two types of nodes are used in this layer, denoted blank and shaded circles in Fig. 8.9, respectively. The node denoted by a blank circle (blank node) is the essential node representing a fuzzy set (described by a Gaussian membership function) of the output variable. Only the center of each Gaussian membership function is delivered to the next layer for the LMOM (local mean of maximum) defuzzification operation [15], and the width is used for output clustering only. Different nodes in Layer 3 may be connected to the same blank node in Layer 4, meaning that the same consequent fuzzy set is specified for different rules. The function of the blank node is

$$a^{(4)} = \sum_j u_j^{(4)} \cdot a_{0i}, \tag{8.12}$$

where $a_{0i} = m_{0i}$, the center of a Gaussian membership function. As to the shaded node, it is generated only when necessary. Each node in Layer 3 has its own corresponding shaded node in Layer 4. One of the inputs to a shaded node is the output delivered from Layer 3, and the other possible inputs (terms) are the input variables from Layer 1. The shaded node function is

$$a^{(4)} = \sum_j a_{ji} x_j \cdot u_i^{(4)}, \tag{8.13}$$

where the summation is over all the inputs and a_{ji} is the corresponding parameter. Combining these two types of nodes in Layer 5, we obtain the whole function performed by this layer for each rule as

$$a^{(4)} = (\sum_j a_{ji} x_j + a_{0i}) u_i^{(4)}. \tag{8.14}$$

Layer 5: Each node in this layer corresponds to one output variable. The node integrates all the actions recommended by Layers 3 and 4 and acts as a defuzzifier with

$$a^{(5)} = \sum_i a_i^{(4)} \Big/ \sum_i a_i^{(3)}. \tag{8.14}$$

Two types of learning, structure and parameter learning, are used concurrently for constructing the SONFIN. The structure learning includes both the precondition and consequent structure identification of a fuzzy *if-then* rule. Here the precondition structure identification corresponds to the input space partitioning and can be formulated as a combinational optimization problem with the following two objectives: to minimize the number of rules generated and to minimize the number of fuzzy sets on the universe of discourse of each input variable. As to the consequent structure identification, the main task is to decide when to generate a new membership function for the output variable and which significant terms (input variables) should be added to the consequent part (a linear equation) when necessary. For parameter learning, based on supervised learning algorithms, the parameters of the linear equations in the consequent parts are adjusted by either LMS or RLS algorithms, and the parameters in the precondition part are adjusted by the backpropagation algorithm to minimize a given cost function. The SONFIN can be used for normal operation at any time during the learning process without repeated training on the input-output patterns when online operation is required. There are no rules (i.e., no nodes in the network except the input/output nodes) in the SONFIN initially. They are created dynamically as learning proceeds on receiving online incoming training data by performing the following learning processes simultaneously, (A) input/output space partitioning, (B) construction of fuzzy rules, (C) consequent structure identification, and (D) parameter identification. Processes A, B, and C belong to the structure learning phase, and process D belongs to the parameter learning phase. The details of these learning processes can be found in [14].

8.3.3 Cascaded Architecture of a Neural Fuzzy Network with Feature Mapping (CNFM)

This section gives the details of how to implement the cascaded architecture of the unsupervised and supervised neural networks presented in Sections 8.3.1 and 8.3.2, respectively. Compared to the conventional methods that use supervised neural networks alone and select their inputs through trial and error, our system is composed of two connective neural networks. The general architecture of CNFM is set up as follows. First, we use three Kohonen' SOMs to reduce the dimensions of three sets of inputs. These inputs are gray values, statistical properties, and features from wavelet decomposition. Instead of using all sets of selected features as the inputs of a supervised neural network, our Kohonen' SOM can transform each set of features into 2D coordinates and use these low-dimension inputs as the input of our supervised neural fuzzy network, SONFIN. Thus, no matter how many sets and features are present in each set, we can transform the inputs into this simple representation. Not only can we get a better representation, but we can obtain a graceful meaning when the 2D coordinates serve as the inputs to a supervised neural network.

To go into a little more detail, Fig. 8.10 shows the architecture of our system. The first set of inputs contains three gray values, each of which comes from one of

three channels, and represents the features of spatial domain. The second set of inputs consists of angular second-moment, contrast, inverse difference moment, and entropy that come from co-occurrence matrices. This set of inputs represents the features of statistical domain. The last set of inputs includes energy and entropy that come from wavelet decomposition and represents the features of spectral domain. After the transformation of Kohonen's SOMs, these three sets of inputs are reduced into 2D coordinates. However, we do not expand a gray value into 2D coordinates because its dimension is low enough in our test examples. If we have transformed the gray values of all channels, we may remove the information of differences among channels. Because the statistical and spectral features focus on local varieties, we can apply the transformation directly and do not need to care for the information of differences among channels. Then we reduce the dimension from 39 features to 15. As shown in Fig. 8.10, the three types of inputs are: three gray values obtained from each channel, four statistical features from each of the three channels, and eight spectral features obtained by wavelet decomposition from each of the three channels. Hence, if we do not reduce the dimension of the inputs, there will be 39 input features. If the number of features increases, a large number of inputs are to be used for classification. However, our system can solve this problem gracefully. Input dimension is first reduced by Kohonen's SOM, then further classification is performed by SONFIN.

After the features have been reduced and transformed by Kohonen's SOM, we pass them to a supervised neural fuzzy network, SONFIN. This network can perform online input space partitioning, which creates only the significant membership functions on the universe of discourse of each input variable using a fuzzy measure algorithm and the orthogonal least square (OLS) method. Our objective is to use the ability of SONFIN to obtain lower mean square error (MSE) and higher learning rate. The result will be compared to that of normally used statistical methods and backpropagation network in Section 8.4.

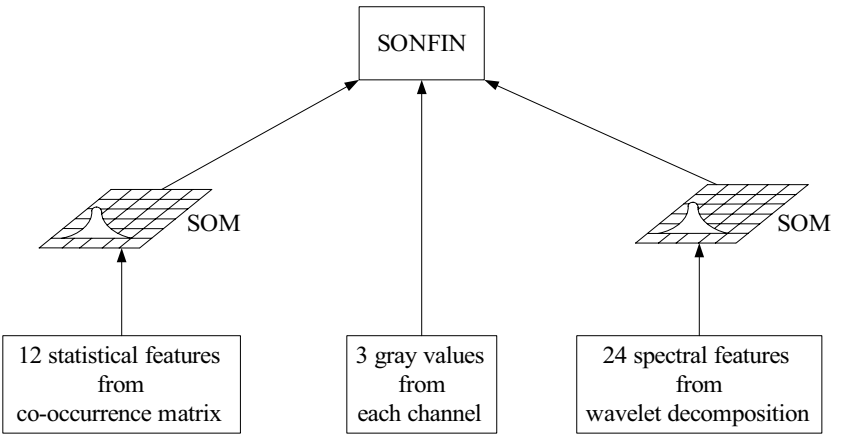


Fig. 8.10. System architecture of CNFM.

8.4 Experimental Results

Our experiment uses the SPOT spectral satellite images provided by the Earth Resource Satellite Receiving Station in Taiwan. These images contain five classes: soil, city 1, city 2, sea, and forest. Our objective is to classify these classes from the three-channel satellite images. The results show that when we try to use all inputs without any dimension reduction to perform the training of a classifier, either the SONFIN or the backpropagation (BP) network, the result is poor and the MSE is very large. We first apply our CNFM to reduce the dimension of inputs and then perform the training of the classifier.

An illustrative example of classification of a channel is shown in Fig. 8.11. It gives the details of our proposed system. First we obtain the gray value of a center-pixel. Its value is 186. Among those points, we calculate the ASM, CON, IDM, entropies, and energies from the co-occurrence matrices and wavelet transformations with a neighborhood size $N = 7$. After we have acquired the input features, we pass each group of the features into Kohonen's SOM. Each group of input features with high dimension is reduced to 2D coordinates. Finally, we use these 2D coordinates, combined with the coordinates from other channels, as the inputs to our neural fuzzy network, SONFIN. The result we obtained is the desired class; here it is class 3.

Tables 8.1, 8.2, and 8.3 show the results of the classification using the proposed CNFM. Here we compared the accuracy among different types of inputs. The results in Table 8.1 are the best, and we obtain good results because all information was used as inputs to CNFM. Table 8.2 shows the result of classification when gray level information and statistical features are used. This is the general architecture that most people use as their framework. We can see that the accuracy decreased in each class. The results in Table 8.3 are obtained using any three gray values. It can be seen that without sufficient information, good classification is hard to achieve. Figure 8.12 shows the original satellite image with ground truth and the classified image using CNFM with three types of inputs. From Fig. 8.12, we can conclude that CNFM can perform an accurate classification.

Table 8.4 shows the classification results of the *k*-nearest-neighborhood (KNN) method, BP network, and the CNFM. The inputs of the BP network and CNFM are first filtered by Kohonen's SOM, whereas KNN uses all features directly. It can be easily concluded that the KNN and BP network are insufficient to obtain accurate results. Figure 8.13 shows the MSE (mean square error) of the BP network against SONFIN in our CNFM. We can conclude that the BP network is not adequate to reduce the MSE due to the complexity of the images, and our system can tackle this problem successfully. The used BP network has two hidden layers with 30 hidden nodes in each layer.

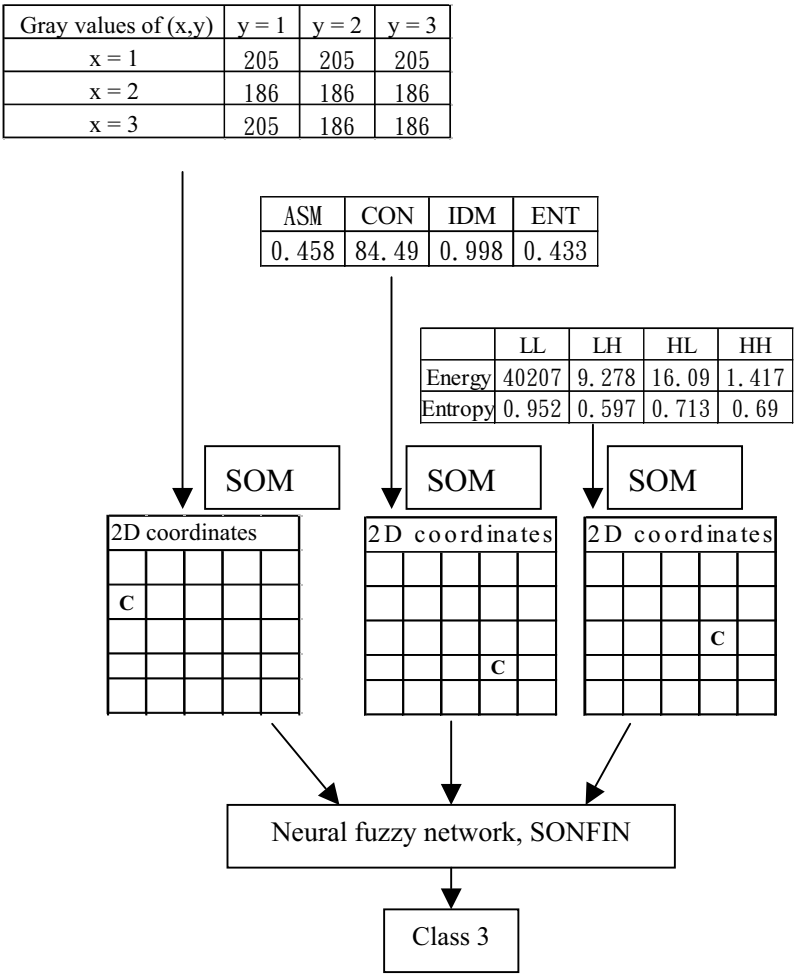


Fig. 8.11. An illustrative example of the CNFM.

Table 8.1. Classification results with all types of inputs.

	Class 1	Class 2	Class 3	Class 4	Class 5	Overall	Average
Class 1	495	0	0	4	1	99 %	96.5 %
Class 2	0	512	0	0	0	100 %	
Class 3	0	0	470	0	0	100 %	
Class 4	7	2	1	424	28	92.2 %	
Class 5	25	5	0	13	462	91.5%	

Table 8.2. Classification results with gray levels and statistical features.

	Class 1	Class 2	Class 3	Class 4	Class 5	Overall	Average
Class 1	490	0	0	7	3	98 %	94.88 %
Class 2	0	512	0	0	0	100 %	
Class 3	0	0	470	0	0	100 %	
Class 4	9	0	0	416	37	90.3 %	
Class 5	44	8	5	16	435	86.1%	

Table 8.3. Classification result with gray levels only.

	Class 1	Class 2	Class 3	Class 4	Class 5	Overall	Average
Class 1	462	0	3	23	12	92.4 %	92.48 %
Class 2	9	477	5	10	11	93.2 %	
Class 3	0	0	470	0	0	100 %	
Class 4	17	7	3	384	51	83.1 %	
Class 5	21	7	1	9	649	93.7 %	

Table 8.4. Comparison of the classification results among KNN, BP, and CNFM.

	KNN	BP	CNFM
Class 1	77%	88%	99 %
Class 2	82%	85%	100 %
Class 3	97.5%	100%	100 %
Class 4	73%	78.5%	92.2 %
Class 5	70%	77%	91.5%

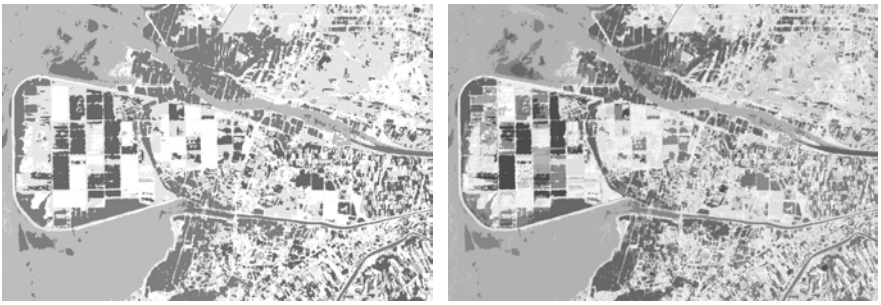


Fig. 8.12. Desired satellite image (left) and the classified result of CNFM (right).

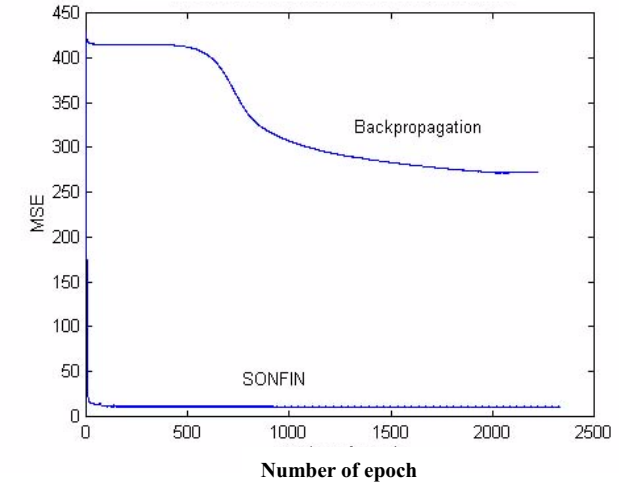


Fig. 8.13. The MSE of backpropagation network against SONFIN.

8.5 Conclusions

Data mining has become widely recognized as an important concept by researchers in many areas. The use of valuable information "mined" from data is recognized as necessary to maintain competitiveness in today's business environments. This chapter proposed a cascaded architecture of neural fuzzy network with feature mapping (CNFM) for a pragmatic approach to solving pattern recognition, classification problems, and feature analysis, which are all important for data mining and knowledge extraction. The proposed CNFM signifies a novel algebraic system identification method, which can be used for knowledge extraction in general and for satellite image analysis in particular.

Although the neural network is a useful black-box tool that can do good prediction for many problems, it still cannot handle the problem of "garbage in, garbage out." We must provide some auxiliary information to handle the texture problem and the variation of neighborhood in satellite images. However, the additional features may increase the computation complexity and mislead the training. To overcome this problem, we presented a new cascaded system that uses Kohonen's self-organizing feature map as the reduction mechanism of input dimension. After we have extracted the lower-dimensional inputs, the classification task is performed by a neural fuzzy network (SONFIN).

The important contribution of the proposed system is that it can solve the problem of scaling and selection of features gracefully. Also, we extended the mechanism of Kohonen's SOM to reduce the high input dimension rather than filtering the training sets. Our CNFM combines spatial, statistical, and spectral features into one system resulting in a classifier with improved performance. Furthermore, we improved the computation complexity of the co-occurrence matrix by the symmetric linked list (SLL) structure. Our future work is to enhance

by the symmetric linked list (SLL) structure. Our future work is to enhance our cascaded system by applying edge information. The edge information can be used as an outline of the distribution of clusters because it can provide information about the distribution of textures. We hope that by applying this information to our system, the classification accuracy can be further improved.

8.6 References

[1] Kennedy, R. L., Lee, Y., Roy, B. V., Reed, C. D., and Lippmann, R. P., *Solving data mining problems through pattern recognition*, Prentice Hall, 1998.

[2] Lippmann, R. P., Pattern classification using neural networks, *IEEE Communications Magazine*, pp. 47–54, 1989.

[3] Lu, H., Setiono, R., Liu, H., Effective data mining using neural networks, *IEEE Transactions on Knowledge and Data Engineering*, 8, pp. 957–61, Dec. 1996.

[4] Introduction to the special issue on neural networks for data mining and knowledge discovery, *IEEE Transactions on Neural Networks*, 11, May, pp.545–9, 2000.

[5] Kaizer, H., A quantification of textures on aerial photographs, Boston Univ. Res. Lab., Boston, Ma Tech. Note 121, AD 69484, 1955.

[6] Darling, E. M., and Joseph, R. D., Pattern recognition from satellite altitudes, *IEEE Trans. Syst. Sci. Cybern.*, SSC-4, pp. 38–47, Mar. 1968.

[7] Haralick, R. M., Shanmugam, K., and Dinstein, I., Texture features for image classification, *IEEE Trans. Syst. Man Cybern.*, 3, pp. 610–21, 1973.

[8] Boucher, J. M., Lena, P., and Marchand, J. F., Application of local and global unsupervised Bayesian classification algorithms to the forest, *IGARSS'93, Better Understanding of Earth Environment* (Cat. No. 93CH3294-6), 2, pp. 737–9, 1993.

[9] Rignot, E., and Chellappa, R., Segmentation of polarimetric synthetic aperture radar data, *IEEE Trans. Image Processing*, 1 (3), pp. 281–99, Jul. 1997.

[10] Solberg, H. S., Taxt, T., and Jain, A. K., A Markov random field model for classification of multisource satellite imagery, *IEEE Trans. Geosci. Remote Sensing*, 34 (1), pp. 100–13, Jan. 1996.

[11] Hara, Y., Atkins, G., Yueh, S. H., Shin, R. T., and Kong, J. A., Application of neural networks to radar image classification, *IEEE Trans. Geosci. Remote Sensing*, 32 (1), pp. 100–9, Jan. 1994.

[12] Tzeng, Y. C., Chen, K. S., Kao, W. L., and Fung, A. K., A dynamic learning neural network for remote sensing applications, *IEEE Trans. Geosci. Remote Sensing*, 32 (5), pp. 994–1102, Sept. 1994.

[13] Tzeng, Y. C., and Chen, K. S., A fuzzy neural network to SAR image classification, *IEEE Trans. Geosci. Remote Sensing*, 36 (1), pp. 301–7, Jan. 1998.

[14] Juang, C. F., and Lin, C. T., An on-line self-constructing neural fuzzy inference network for system modeling, *IEEE Trans. Fuzzy Systems*, 6 (1), Feb.

1998.
[15] Lin, C. T., and Lee, C. S. G., Neural-network-based fuzzy logic control and decision system, *IEEE Trans. Comput.*, 40 (12), pp. 1320–36, 1991.
[16] Daubechies, I., Orthonormal bases of compactly supported wavelets, *Commun. Pure Appl. Math.*, 41, pp. 909–96, Nov. 1988.
[17] Mallat, S. G., Multiresolution approximation and wavelets, *Trans. Amer. Math. Soc. 1989*, pp. 69–88, Sept. 1989.
[18] DeSieno, D., Adding a conscience to competitive learning, *IEEE ICNN (San Diego)*, 1, pp.117–24, 1988.

9. Discovery of Positive and Negative Rules from Medical Databases Based on Rough Sets

Shusaku Tsumoto

Department of Medical Informatics, Shimane University, School of Medicine, 89-1 Enya-cho Izumo City, Shimane 693-8501, Japan;
email: tsumoto@computer.org

One of the important problems in rule-induction methods is that extracted rules do not plausibly represent information on experts' decision processes. To solve this problem, the characteristics of medical reasoning are discussed. The concept of positive and negative rules is introduced. Then, for induction of positive and negative rules, two search algorithms are provided. The proposed rule-induction method is evaluated on medical databases. The experimental results show that the induced rules correctly represent experts' knowledge, and several interesting patterns are discovered.

9.1 Introduction

Rule-induction methods are classified into two categories, induction of deterministic rules and of probabilistic ones [9.4], [9.5], [9.7], [9.10]. On one hand, deterministic rules are described as *if-then* rules, which can be viewed as propositions. From the set-theoretical point of view, a set of examples supporting the conditional part of a deterministic rule, denoted by C, is a subset of a set whose examples belong to the consequence part, denoted by D. That is, the relation $C \subseteq D$ holds and deterministic rules are supported only by positive examples in a data set. On the other hand, probabilistic rules are *if-then* rules with probabilistic information [9.10]. When a classical proposition will not hold for C and D, C is not a subset of D but closely overlapped with D. That is, the relations $C \cap D \neq \phi$ and $|C \cap D|/|C| \geq \delta$ will hold in this case, where the threshold δ is the degree of closeness of overlapping sets, which will be given by domain experts. (For more information, see Section 9.3.) Thus, probabilistic rules are supported by a large number of positive examples and a few negative examples. The common feature of both deterministic and probabilistic rules is that they deduce their consequence positively if an example satisfies their conditional parts. We call the reasoning by these rules *positive reasoning*.

However, medical experts use not only positive reasoning but also *negative reasoning* for selection of candidates, which is represented as *if-then* rules whose consequences include negative terms. For example, when a patient who complains of headache does not have a throbbing pain, migraine should not be suspected with a high probability. Thus, negative reasoning

also plays an important role in cutting the search space of a differential diagnosis process [9.10]. Thus, medical reasoning includes both positive and negative reasoning, though conventional rule-induction methods do not reflect this aspect. This is one of the reasons medical experts have difficulty in interpreting induced rules, and interpreting rules for a discovery procedure does not proceed easily. Therefore, negative rules should be induced from databases in order not only to induce rules reflecting experts' decision processes, but also to induce rules that will be easier for domain experts to interpret, both of which are important to enhance the discovery process done by the cooperation of medical experts and computers.

In this chapter, first the characteristics of medical reasoning are discussed and then two kinds of rules, positive rules and negative rules, are introduced as a model of medical reasoning. Interestingly, from the set-theoretic point of view, sets of examples supporting both rules correspond to the lower and upper approximations in rough sets [9.5]. On the other hand, from the viewpoint of propositional logic, both positive and negative rules are defined as classical propositions or deterministic rules with two probabilistic measures, classification accuracy, and coverage. Second, two algorithms for induction of positive and negative rules are introduced, defined as search procedures using accuracy and coverage as evaluation indices. Finally, the proposed method is evaluated on several medical databases. The experimental results show that the induced rules correctly represent experts' knowledge. In addition, several interesting patterns are discovered.

9.2 Focusing Mechanism

One of the characteristics in medical reasoning is a focusing mechanism, which is used to select the final diagnosis from many candidates [9.10], [9.11]. For example, in differential diagnosis of headache, more than 60 diseases will be checked by present history, physical examinations, and laboratory examinations. In diagnostic procedures, a candidate is excluded if a symptom necessary to diagnose is not observed.

This style of reasoning consists of the following two processes: exclusive reasoning and inclusive reasoning. Relations of this diagnostic model with another diagnostic model are discussed in [9.12]. The diagnostic procedure proceeds as follows (Fig. 9.2): First, exclusive reasoning excludes a disease from candidates when a patient does not have a symptom that is necessary to diagnose that disease. Second, inclusive reasoning suspects a disease in the output of the exclusive process when a patient has symptoms specific to a disease. These two steps are modeled as two kinds of rules, negative rules (or exclusive rules) and positive rules; the former corresponds to exclusive reasoning, the latter to inclusive reasoning. In the next two sections, these two rules are represented as special kinds of probabilistic rules.

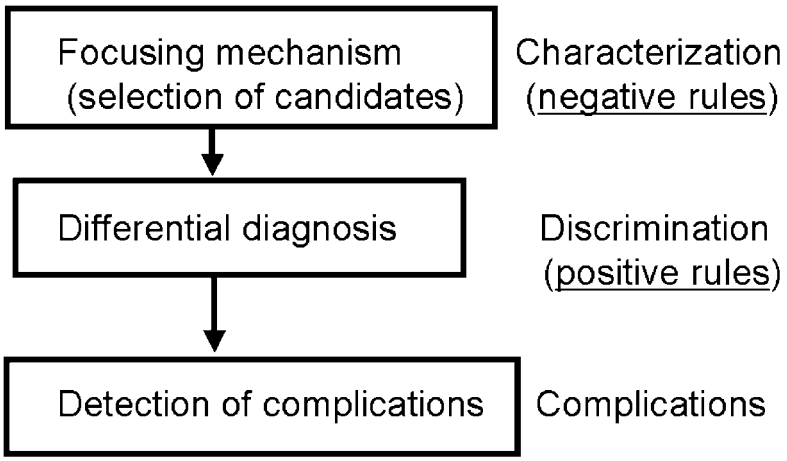


Fig. 9.1. Illustration of focusing mechanism.

9.3 Definition of Rules

9.3.1 Rough Sets

In the following sections, we use the following notation introduced by Grzymala-Busse and Skowron [9.8], based on rough set theory [9.5]. These notations are illustrated by a small data set shown in Table 9.1, which includes symptoms exhibited by six patients who complained of headache.

Let U denote a nonempty finite set called the universe and A denote a nonempty, finite set of attributes, i.e., $a : U \rightarrow V_a$ for $a \in A$, where V_a is called the domain of a, respectively. Then a decision table is defined as an information system, $A = (U, A \cup \{d\})$. For example, Table 9.1 is an information system with $U = \{1, 2, 3, 4, 5, 6\}$ and $A = \{age, location, nature, prodrome, nausea, M1\}$ and $d = class$. For $location \in A$, $V_{location}$ is defined as $\{occular, lateral, whole\}$.

Table 9.1. An example of a data set.

No.	Age	Location	Nature	Prodrome	Nausea	M1	Class
1	50–59	occular	persistent	no	no	yes	m.c.h.
2	40–49	whole	persistent	no	no	yes	m.c.h.
3	40–49	lateral	throbbing	no	yes	no	migra
4	40–49	whole	throbbing	yes	yes	no	migra
5	40–49	whole	radiating	no	no	yes	m.c.h.
6	50–59	whole	persistent	no	yes	yes	psycho

M1: tenderness of M1; m.c.h.: muscle contraction headache; migra: migraine; psycho: psychological pain.

The atomic formulas over $B \subseteq A \cup \{d\}$ and V are expressions of the form $[a = v]$, called descriptors over B, where $a \in B$ and $v \in V_a$. The set $F(B, V)$ of formulas over B is the least set containing all atomic formulas over B and closed with respect to disjunction, conjunction, and negation. For example, $[location = occular]$ is a descriptor of B.

For each $f \in F(B, V)$, f_A denotes the meaning of f in A, i.e., the set of all objects in U with property f, defined inductively as follows:

1. If f is of the form $[a = v]$, then $f_A = \{s \in U | a(s) = v\}$.
2. $(f \wedge g)_A = f_A \cap g_A$; $(f \vee g)_A = f_A \vee g_A$; $(\neg f)_A = U - f_a$.

For example, $f = [location = whole]$ and $f_A = \{2, 4, 5, 6\}$. As an example of a conjunctive formula, $g = [location = whole] \wedge [nausea = no]$ is a descriptor of U and f_A is equal to $g_{location,nausea} = \{2, 5\}$.

9.3.2 Classification Accuracy and Coverage

Definition of Accuracy and Coverage. By use of the preceding framework, classification accuracy and coverage, or true positive rate are defined as follows.

Definition 9.3.1. *Let R and D denote a formula in $F(B, V)$ and a set of objects that belong to a decision d. Classification accuracy and coverage(true positive rate) for $R \rightarrow d$ is defined as:*

$$\alpha_R(D) = \frac{|R_A \cap D|}{|R_A|} (= P(D|R)) \text{ and}$$
$$\kappa_R(D) = \frac{|R_A \cap D|}{|D|} (= P(R|D)),$$

where $|S|$, $\alpha_R(D)$, $\kappa_R(D)$, and $P(S)$ denote the cardinality of a set S, a classification accuracy of R as to classification of D, and coverage (a true positive rate of R to D), and probability of S, respectively.

Figure 9.2 depicts the Venn diagram of relations between accuracy and coverage. Accuracy views the overlapped region $|R_A \cap D|$ from the meaning of a relation R. On the other hand, coverage views the overlapped region from the meaning of a concept D.

In the preceding example, when R and D are set to $[nau = yes]$ and $[class = migraine]$, $\alpha_R(D) = 2/3 = 0.67$ and $\kappa_R(D) = 2/2 = 1.0$.

It is notable that $\alpha_R(D)$ measures the degree of the sufficiency of a proposition, $R \rightarrow D$, and that $\kappa_R(D)$ measures the degree of its necessity. For example, if $\alpha_R(D)$ is equal to 1.0, then $R \rightarrow D$ is true. On the other hand, if $\kappa_R(D)$ is equal to 1.0, then $D \rightarrow R$ is true. Thus, if both measures are 1.0, then $R \leftrightarrow D$.

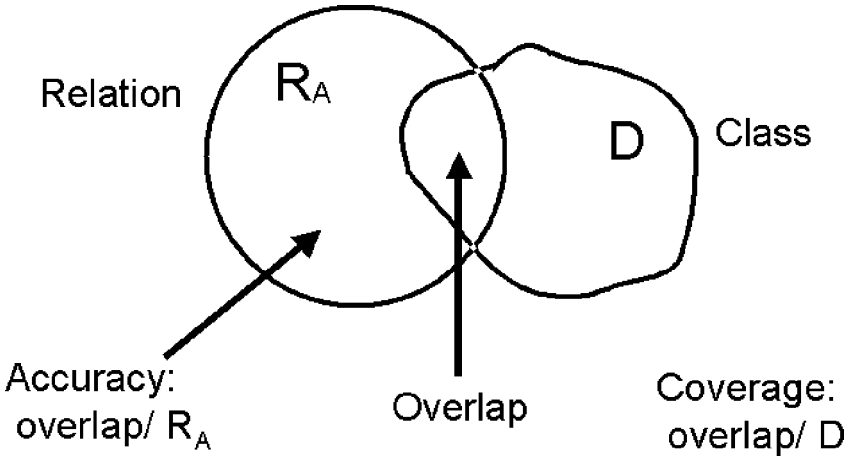


Fig. 9.2. Venn diagram of accuracy and coverage.

9.3.3 Probabilistic Rules

By use of accuracy and coverage, a probabilistic rule is defined as:

$$R \overset{\alpha,\kappa}{\rightarrow} d \quad s.t. \quad R = \wedge_j [a_j = v_k], \alpha_R(D) \geq \delta_\alpha \text{ and } \kappa_R(D) \geq \delta_\kappa.$$

If the thresholds for accuracy and coverage are set to high values, the meaning of the conditional part of probabilistic rules corresponds to the highly overlapped region. Figure 9.3 depicts the Venn diagram of probabilistic rules with highly overlapped region. This rule is a kind of probabilistic proposi-

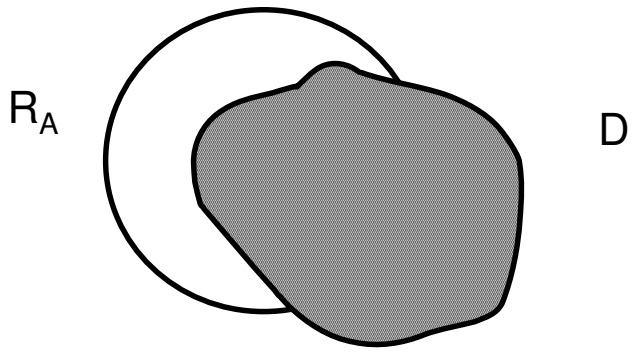


Fig. 9.3. Venn diagram for probabilistic rules.

tion with two statistical measures, which is an extension of Ziarko's variable precision model (VPRS) [9.15].[1]

It is also notable that both a positive rule and a negative rule are defined as special cases of this rule, as shown in the next sections.

9.3.4 Positive Rules

A positive rule is defined as a rule supported by only positive examples, the classification accuracy of which is equal to 1.0. It is notable that the set supporting this rule corresponds to a subset of the lower approximation of a target concept, which is introduced in rough sets [9.5]. Thus, a positive rule is represented as:

$$R \rightarrow d \quad s.t. \qquad R = \wedge_j [a_j = v_k], \quad \alpha_R(D) = 1.0$$

. Figure 9.4 shows the Venn diagram of a positive rule. As shown in this figure, the meaning of R is a subset of that of D. This diagram is exactly equivalent to the classic proposition $R \rightarrow d$.

In the preceding example, one positive rule of m.c.h. (muscle contraction headache) is:

$$[nausea = no] \rightarrow m.c.h. \quad \alpha = 3/3 = 1.0.$$

This positive rule is often called a deterministic rule. However, we use the term, positive (deterministic) rules, because a deterministic rule supported only by negative examples, called a negative rule, is introduced in the next section.

[1] This probabilistic rule is also a kind of *rough modus ponens* [9.6].

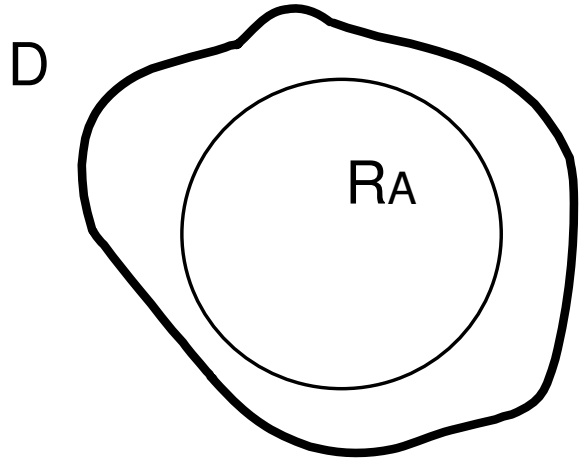


Fig. 9.4. Venn diagram of positive rules.

9.3.5 Negative Rules

Before defining a negative rule, let us first introduce an exclusive rule, the contrapositive of a negative rule [9.10]. An exclusive rule is defined as a rule supported by all the positive examples, the coverage of which is equal to 1.0. That is, an exclusive rule represents the necessity condition of a decision. It is notable that the set supporting an exclusive rule corresponds to the upper approximation of a target concept, which is introduced in rough sets [9.5]. Thus, an exclusive rule is represented as:

$$R \rightarrow d \quad s.t. \qquad R = \vee_j [a_j = v_k], \quad \kappa_R(D) = 1.0.$$

Figure 9.5 shows the Venn diagram of an exclusive rule. As shown in this figure, the meaning of R is a superset of that of D. This diagram is exactly equivalent to the classic proposition $d \rightarrow R$.

In the preceding example, the exclusive rule of m.c.h. is:

$$[M1 = yes] \vee [nau = no] \rightarrow m.c.h. \quad \kappa = 1.0,$$

From the viewpoint of propositional logic, an exclusive rule should be represented as:

$$d \rightarrow \vee_j [a_j = v_k],$$

because the condition of an exclusive rule corresponds to the necessity condition of conclusion d. Thus, it is easy to see that a negative rule is defined as the contrapositive of an exclusive rule:

$$\wedge_j \neg [a_j = v_k] \rightarrow \neg d,$$

which means that if a case does not satisfy any attribute value pairs in the condition of a negative rule, then we can exclude a decision d from candidates. For example, the negative rule of m.c.h. is:

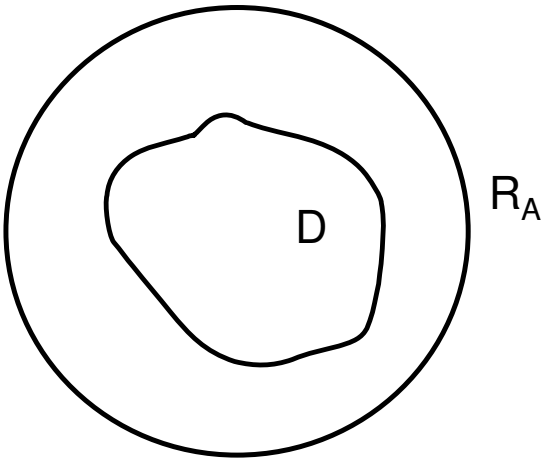


Fig. 9.5. Venn diagram of exclusive rules.

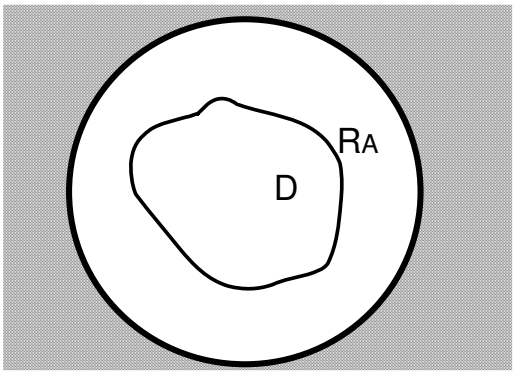


Fig. 9.6. Venn diagram of negative rules.

$$\neg[M1 = yes] \wedge \neg[nausea = no] \rightarrow \neg m.c.h.$$

In summary, a negative rule is defined as:

$$\wedge_j \neg[a_j = v_k] \rightarrow \neg d \quad s.t. \quad \forall [a_j = v_k]\ \kappa_{[a_j = v_k]}(D) = 1.0,$$

where D denotes a set of samples that belong to a class d. Figure 9.6 shows the Venn diagram of a negative rule. As shown in this figure, it is notable that this negative region is the "positive region" of "negative concept."

Negative rules should also be included in a category of deterministic rules, because their coverage, a measure of negative concepts, is equal to 1.0. It is also notable that the set supporting a negative rule corresponds to a subset of negative region, which is introduced in rough sets [9.5].

In summary, positive and negative rules correspond to positive and negative regions defined in rough sets. Figure 9.7 shows the Venn diagram of those rules.

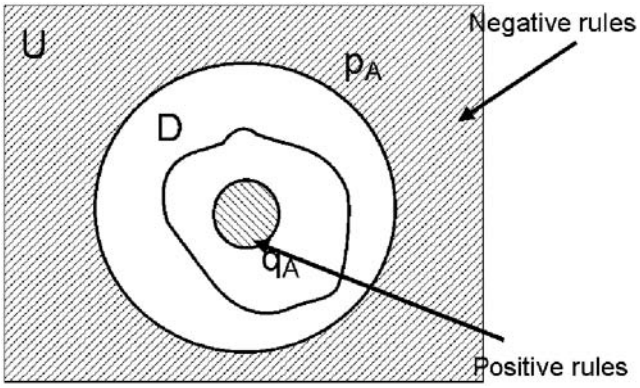


Fig. 9.7. Venn diagram of defined rules.

9.4 Algorithms for Rule Induction

The contrapositive of a negative rule, an exclusive rule, is induced as an exclusive rule by the modification of the algorithm introduced in PRIMEROSE-REX [9.10], as shown in Fig. 9.8. This algorithm works as follows. (1) First it selects a descriptor $[a_i = v_j]$ from the list of attribute-value pairs, denoted by L. (2) Then it checks whether this descriptor overlaps with a set of positive examples, denoted by D. (3) If so, this descriptor is included in a list of candidates for positive rules and the algorithm checks whether its coverage is equal to 1.0. If the coverage is equal to 1.0, then this descriptor is added to $R_e r$, the formula for the conditional part of the exclusive rule of D. (4) Then $[a_i = v_j]$ is deleted from the list L. This procedure, from (1) to (4), will continue unless L is empty. (5) Finally, when L is empty, this algorithm generates negative rules by taking the contrapositive of induced exclusive rules.

On the other hand, positive rules are induced as inclusive rules by the algorithm introduced in PRIMEROSE-REX [9.10], as shown in Fig. 9.9. For induction of positive rules, the threshold of accuracy and coverage is set to 1.0 and 0.0, respectively.

This algorithm works in the following way. (1) First it substitutes L_1, which denotes a list of formulas composed of only one descriptor, with the list L_{er} generated by the former algorithm shown in Fig. 9.1. (2) Then until L_1 becomes empty, the following steps will continue: (a) A formula $[a_i = v_j]$ is removed from L_1. (b) Then the algorithm checks whether $\alpha_R(D)$ is larger than the threshold. (For induction of positive rules, this is equal to checking whether $\alpha_R(D)$ is equal to 1.0.) If so, then this formula is included a list of the conditional parts of positive rules. Otherwise, it will be included in M, which is used for making conjunctions. (3) When L_1 is empty, the next list L_2 is generated from the list M.

9.5 Experimental Results

For experimental evaluation, a new system, called PRIMEROSE-REX2 (Probabilistic Rule Induction Method for Rules of Expert System ver. 2.0), was developed, where the algorithms discussed in Section 9.4 were implemented.

PRIMEROSE-REX2 was applied to the following three medical domains: (1) headache (RHINOS domain), whose training samples consist of 52,119 samples, 45 classes, and 147 attributes; (2) cerebulovasular diseases (CVD), whose training samples consist of 7620 samples, 22 classes, and 285 attributes; and (3) meningitis, whose training samples consist of 1211 samples, 4 classes, and 41 attributes (Table 9.2).

For evaluation, we used the following two types of experiments. One experiment was to evaluate the predictive accuracy using the cross-validation method, which is often used in the machine-learning literature [9.9]. The

```
procedure Exclusive and Negative Rules;
  var
    L : List;
      /* A list of elementary attribute-value pairs */
  begin
    L := P0;
    /* P0: A list of elementary attribute-value pairs given in a database */
    while (L ≠ {}) do
      begin
        Select one pair [ai = vj] from L;
        if ([ai = vj]A ∩ D ≠ φ) then do /* D: positive examples of a target class d */
          begin
          Lir := Lir + [ai = vj];    /* Candidates for Positive Rules */
          if (κ[ai=vj](D) = 1.0)
          then Rer := Rer ∧ [ai = vj];
            /* Include [ai = vj] into the formula of Exclusive Rule */
          end
      L := L − [ai = vj];
      end
    Construct Negative Rules:
        Take the contrapositive of Rer.
  end {Exclusive and Negative Rules};
```

Fig. 9.8. Induction of exclusive and negative rules.

```
procedure Positive Rules;
  var
    i : integer;    M, Li : List;
  begin
    L1 := Lir;
    /* Lir: A list of candidates generated by induction of exclusive rules */
    i := 1;    M := {};
    for i := 1 to n do
    /* n: Total number of attributes given
      in a database */
      begin
        while ( Li ≠ {} ) do
          begin
            Select one pair R = ∧[ai = vj] from Li;
            Li := Li − {R};
            if    (αR(D) > δα)
              then  do Sir := Sir + {R};
          /* Include R in a list of the Positive Rules */
            else M := M + {R};
          end
        Li+1 := (A list of the whole combination of the conjunction formulae in M);
      end
  end {Positive Rules};
```

Fig. 9.9. Induction of positive rules.

Table 9.2. Databases.

Domain	Samples	Classes	Attributes
Headache	52,119	45	147
CVD	7620	22	285
Meningitis	1211	4	41

other experiment was to evaluate the induced rules by medical experts and to check whether these rules lead to a new discovery.

9.5.1 Performance of Rules Obtained

For comparison of performance, the experiments are conducted by the following four procedures. First, rules are acquired manually from experts. Second, the data sets are randomly split into new training samples and new test samples. Third, PRIMEROSE-REX2, conventional rule-induction methods, AQ15 [9.4] and C4.5 [9.7] are applied to the new training samples for rule generation. Fourth, the induced rules and rules acquired from experts are tested using new test samples. The second through fourth steps are repeated 100 times, and the average classification accuracy over 100 trials is computed. This process is a variant of repeated two-fold cross-validation, introduced in [9.10].

Experimental results (performance) are shown in Table 9.3. The first and second rows show the results obtained using PRIMROSE-REX2; the results in the first row are derived using both positive and negative rules and those in the second row are derived by only positive rules. The third row shows the results derived from medical experts. For comparison, we compare the classification accuracy of C4.5 and AQ-15, which is shown in the fourth and fifth rows.

These results show that the combination of positive and negative rules outperforms positive rules, although it is a little worse than medical experts' rules.

Table 9.3. Experimental results (accuracy: averaged).

Method	Headache	CVD	Meningitis
PRIMEROSE-REX2 (positive+negative)	91.3%	89.3%	92.5%
PRIMEROSE-REX2 (positive)	68.3%	71.3%	74.5%
Experts	95.0%	92.9%	93.2%
C4.5	85.8%	79.7%	81.4%
AQ15	86.2%	78.9%	82.5%

9.6 What Is Discovered?

9.6.1 Interesting Rules Are Very Few

One of the most important observations is that there were very few rules interesting or unexpected to medical experts compared to the number of rules extracted from the data sets.

Table 9.4 shows the mentioned results. The second column denotes the number of positive and negative rules obtained from each data set. The third column denotes the number of positive and negative rules interesting or unexpected for domain experts. For example, the first row shows that the number of induced positive rules in headache is 24,335, but the number of interesting rules are 114, which shows that only 0.47% of rules are interesting for domain experts.

This table shows that the number of interesting rules are very few: even in the case of meningitis, only 6.5% are interesting, which suggests that the interpretation part of domain experts is very hard for the knowledge discovery process.

Next, we show several examples of rules that are interesting to domain experts.

9.6.2 Positive Rules in Differential Diagnosis of Headache

In the domain of differential diagnosis of headache, the following interesting positive rules were found.

$$[Age < 20] \wedge [History : paroxysmal] \rightarrow Common\ Migraine$$
$$(Coverage : 0.75)$$

This rule is said to be interesting if it is compared with the following rule:

$$[Age > 40] \wedge [History : paroxysmal] \rightarrow Classic\ Migraine$$
$$(Coverage : 0.72)$$

These two rules include two parts; although the values of the attribute "Age" are different, those of the attribute "History" are the same. This suggests that the attribute "Age" is important for differential diagnosis between common migraine and classic migraine.

Table 9.4. Number of extracted rules and interesting rules.

	Induced positive rules	Interesting rules
Headache	24,335	114 (0.47%)
CVD	14,715	106 (0.72%)
Meningitis	1,922	40 (2%)
	Induced negative rules	Interesting rules
Headache	12,113	120 (0.99%)
CVD	7,231	155 (2.1%)
Meninigitis	77	5 (6.5%)

9.6.3 Negative Rules in Differential Diagnosis of Headache

In the domain of differential diagnosis of headache, the following interesting negative rule was found.

$$\neg[Nature : Persistent] \wedge \neg[History : acute] \wedge \neg[History : paroxysmal]) \\ \wedge\neg[Neck\ Stiffness : yes] \rightarrow \neg Common\ Migraine$$

It is notable that it is difficult even for domain experts to interpret the interestingness of this rule if only this rule is shown. The domain experts pointed out that the rule is interesting when compared with the following rules.

$$[Nature : Persistent] \wedge [History : acute] \wedge [Neck\ Stiffness : yes] \\ \rightarrow Meningitis \\ [Nature : Persistent] \wedge [History : chronic] \rightarrow Brain\ Tumor$$

This means that the rule includes information that is very important for differential diagnosis between common migraine and meningitis or brain tumor. Note that these two rules are very similar to the negative rule except for the negative symbols.

9.6.4 Positive Rules in Meningitis

In the domain of meningitis, the following positive rules, which medical experts do not expect, are obtained.

$$[WBC < 12000] \wedge [Sex = Female] \wedge [Age < 40] \\ \wedge[CSF\_CELL < 1000] \rightarrow Virus\ (\text{Coverage: } 0.91) \\ [Age \geq 40] \wedge [WBC \geq 8000] \wedge [Sex = Male] \\ \wedge[CSF\_CELL \geq 1000] \rightarrow Bacteria(Coverage : 0.64)$$

The former rule means that if WBC (white blood cell count) is less than 12000, the gender of a patient is female, the age is less than 40, and CSF_CELL (cell count of cerebulospinal fluid), then the type of meningitis is Viral. The latter means that the age of a patient is less than 40, WBC is larger than 8000, the gender is male, and CSF_CELL is larger than 1000, then the type of meningitis is Bacterial.

The most interesting points are that these rules included information about age and gender, which often seems to be unimportant attributes for differential diagnosis of meningitis. The first discovery was that women did not often suffer from bacterial infection compared with men, because such relationships between gender and meningitis has not been discussed in medical context [9.1]. By a close examination of the database on meningitis, it was found that most of the patients suffered from chronic diseases, such as DM, LC, and sinusitis, which are the risk factors of bacterial meningitis. The second discovery was that $[age < 40]$ was also an important factor not to

suspect viral meningitis, which also matches the fact that most old people suffer from chronic diseases.

These results were also reevaluated in medical practice. Recently, the preceding two rules were checked by an additional 21 cases who suffered from meningitis (15 cases viral, 6 cases bacterial meningitis.) Surprisingly, the rules misclassified only three cases (two viral, the other bacterial), that is, the total accuracy was equal to $18/21 = 85.7\%$, and the accuracies for viral and bacterial meningitis were equal to $13/15 = 86.7\%$ and $5/6 = 83.3\%$. The reasons for misclassification are the following: a case of bacterial infection involved a patient who had a severe immunodeficiency, although he is very young. Two cases of viral infection involved patients who suffered from herpes zoster. It is notable that even those misclassified cases could be explained from the viewpoint of the immunodeficiency: that is, it was confirmed that immunodeficiency is a key factor for meningitis.

The validation of these rules is still ongoing; it will be reported in the near future.

9.6.5 Positive and Negative Rules in CVD

Concerning the database on CVD, several interesting rules were derived. The most interesting results were the following positive and negative rules for thalamus hemorrhage:

$$\begin{aligned}[Sex = Female] \wedge [Hemiparesis = Left] \\ \wedge [LOC : positive] \rightarrow Thalamus \\ \neg[Risk : Hypertension] \wedge \neg[Sensory = no] \\ \rightarrow \neg Thalamus\end{aligned}$$

The former rule means that if the gender of a patient is female and he or she suffered from the left hemiparesis ([Hemiparesis=Left]) and loss of consciousness ([LOC:positive]), then the focus of CVD is thalamus. The latter rule means that if he or she suffers neither from hypertension ([Risk: Hypertension]) or sensory disturbance ([Sensory=no]), then the focus of CVD is thalamus.

Interestingly, LOC (loss of consciousness) under the condition of $[Gender = Female] \wedge [Hemiparesis = Left]$ was found to be an important factor to diagnose thalamic damage. In this domain, any strong correlations between these attributes and others, like the database of meningitis, have not been found yet. It will be our future work to find what factor is behind these rules.

9.7 Rule Discovery as Knowledge Acquisition and Decision Support

9.7.1 Expert System: RH

Another point of discovery of rules is automated knowledge acquisition from databases. Knowledge acquisition is referred to as a bottleneck problem in development of expert systems [9.2], which has not fully been solved and is expected to be solved by induction of rules from databases. However, there are few papers that discuss the evaluation of discovered rules from the viewpoint of knowledge acquisition [9.12].

For this purpose, we have developed an expert system, called RH (rule-based system for headache) using the acquired knowledge. The reason for selecting the domain of headache is that earlier we developed an expert system RHINOS (rule-based headache information organizing system), which makes a differential diagnosis in headache [9.3]. In this system, it takes about six months to acquire knowledge from domain experts. RH consists of two parts. First, it requires inputs and applies exclusive and negative rules to select candidates (focusing mechanism). Then, it requires additional inputs and applies positive rules for differential diagnosis between selected candidates. Finally, RH outputs diagnostic conclusions.

9.7.2 Evaluation of RH

RH was evaluated in clinical practice with respect to its classification accuracy by using 930 patients who came to the outpatient clinic after the development of this system. Experimental results about classification accuracy are shown in Table 9.5. The first and second rows show the performance of rules obtained using PRIMROSE-REX2; the results in the first row are derived using both positive and negative rules and those in the second row are derived using only positive rules. The third and fourth rows show the results derived using both positive and negative rules and those by positive rules acquired directly from medical experts. These results show that the combination of positive and negative rules outperforms positive rules and gains almost the same performance as those by experts.

Table 9.5. Evaluation of RH (accuracy: averaged).

Method	Accuracy
PRIMEROSE-REX2 (positive and negative)	91.4% (851/930)
PRIMEROSE-REX2 (positive)	78.5% (729/930)
RHINOS (positive and negative)	93.5% (864/930)
RHINOS (positive)	82.8% (765/930)

9.8 Discussion

9.8.1 Hierarchical Rules for Decision Support

One of the problems with rule induction is that conventional rule-induction methods cannot extract rules that plausibly represent experts' decision processes [9.12]. The description length of induced rules is too short, compared to the experts' rules. (It may be observed that this length part does not contribute much to the classification performance.) For example, rule-induction methods introduced in this chapter induced the following common rule for muscle contraction headache from databases on differential diagnosis of headache:

$$[location = whole] \quad \wedge[\text{Jolt Headache} = no] \wedge[\text{Tenderness of M1} = yes] \rightarrow \text{muscle contraction headache.}$$

This rule is shorter than the following rule given by medical experts:

$$\begin{aligned}
&[\text{Jolt Headache} = no] \\
&\wedge([\text{Tenderness of M0} = yes] \vee [\text{Tenderness of M1} = yes] \\
&\qquad \vee[\text{Tenderness of M2} = yes]) \\
&\wedge[\text{Tenderness of B1} = no] \wedge [\text{Tenderness of B2} = no] \wedge [\text{Tenderness of B3} = no] \\
&\wedge[\text{Tenderness of C1} = no] \wedge [\text{Tenderness of C2} = no] \wedge [\text{Tenderness of C3} = no] \\
&\wedge[\text{Tenderness of C4} = no] \\
&\quad \rightarrow \text{muscle contraction headache}
\end{aligned}$$

These results suggest that conventional rule-induction methods do not reflect a mechanism of knowledge acquisition of medical experts.

Typically, rules acquired from medical experts are much longer than those induced from databases, the decision attributes of which are given by the same experts. This is because rule induction methods generally search for shorter rules, compared with decision tree induction. In the case of decision tree induction, the induced trees are sometimes too deep, and in order for the trees to be useful for learning, pruning and examination by experts are required. One of the main reasons rules are short and decision trees are sometimes long is that these patterns are generated by only one criteria, such as high accuracy or high information gain. The comparative study in this section suggests that experts should acquire rules by usage of several measures. Those characteristics of medical experts' rules are fully examined not by comparing between those rules for the same class, but by comparing experts' rules with those for another class. For example, a classification rule for muscle contraction headache is given by:

$$\begin{aligned}
&[\text{Jolt Headache} = no] \\
&\wedge([\text{Tenderness of M0} = yes] \vee [\text{Tenderness of M1} = yes] \\
&\qquad \vee[\text{Tenderness of M2} = yes]) \\
&\wedge[\text{Tenderness of B1} = no] \wedge [\text{Tenderness of B2} = no] \\
&\qquad \wedge[\text{Tenderness of B3} = no] \\
&\qquad \wedge[\text{Tenderness of C1} = no] \wedge [\text{Tenderness of C2} = no] \\
&\qquad \wedge[\text{Tenderness of C3} = no] \wedge[\text{Tenderness of C4} = no] \\
&\quad \rightarrow \text{muscle contraction headache}
\end{aligned}$$

This rule is very similar to the following classification rule for disease of cervical spine:

[Jolt Headache = *no*]
∧([Tenderness of M0 = *yes*] ∨ [Tenderness of M1 = *yes*]
∨[Tenderness of M2 = *yes*])
∧([Tenderness of B1 = *yes*] ∨ [Tenderness of B2 = *yes*]
∨[Tenderness of B3 = *yes*]
∨[Tenderness of C1 = *yes*] ∨ [Tenderness of C2 = *yes*]
∨[Tenderness of C3 = *yes*] ∨[Tenderness of C4 = *yes*])
→ disease of cervical spine

The differences between these two rules are attribute-value pairs, from tenderness of B1 to C4. Thus, these two rules can be simplified into the following form:

$$a_1 \wedge A_2 \wedge \neg A_3 \rightarrow \textit{muscle contraction headache,}$$
$$a_1 \wedge A_2 \wedge A_3 \rightarrow \textit{disease of cervical spine.}$$

The first two terms and the third one represent different reasoning. The first and second terms a_1 and A_2 are used to differentiate muscle contraction headache and disease of cervical spine from other diseases. The third term A_3 is used to make a differential diagnosis between these two diseases. Thus, medical experts first select several diagnostic candidates, which are similar to each other, from many diseases and then make a final diagnosis from those candidates. This problem has been partially solved; Tsumoto introduced a new approach for inducing these rules in [9.13], as induction of hierarchical decision rules. In that paper, the characteristics of experts' rules are closely examined and a new approach to extract plausible rules is introduced, which consists of the following three procedures. First, the characterization of decision attributes (given classes) is done from databases and the classes are classified into several groups with respect to the characterization. Then two kinds of subrules, characterization rules for each group and discrimination rules for each class in the group, are induced. Finally, those two parts are integrated into one rule for each decision attribute. The proposed method was evaluated on a medical database, the experimental results of which show that induced rules correctly represent experts' decision processes.

This observation also suggests that medical experts implicitly look at the relation between rules for different concepts. Future work should discover the relations between induced rules.

9.8.2 Relations Between Rules

In [9.14], Tsumoto focuses on the characteristics of medical reasoning(focusing mechanism) and introduces three kinds of rules, positive rules, exclusive rules and total covering rules, as a model of medical reasoning, which is an extended formalization of rules defined in [9.12].

Interestingly, from the set-theoretic point of view, sets of examples supporting these rules correspond to the lower and upper approximations in rough sets. Furthermore, from the viewpoint of propositional logic, both inclusive and exclusive rules are defined as classical propositions, or deterministic rules with two probabilistic measures, classification accuracy and coverage. Total covering rules have several interesting relations with inclusive and exclusive rules, which reflects the characteristics of medical reasoning.

Originally, a total covering rule is defined as a set of symptoms that can be observed in at least one case of a target disease. That is, this rule is defined as a collection of attribute-value pairs whose accuracy is larger than 0:

$$R \rightarrow d \quad s.t. \qquad R = \vee_j [a_j = v_k], \quad \alpha_R(D) > 0.$$

From the definition of accuracy and coverage, this formula can be transformed into:

$$R \rightarrow d \quad s.t. \qquad R = \vee_j [a_j = v_k], \quad \kappa_R(D) > 0.$$

For each attribute, the attribute-value pairs form a partition of U. Thus, for each attribute, total covering rules include a covering of all the positive examples. According to this property, the preceding formula is redefined as:

$$R \rightarrow d \quad s.t. \qquad R = \vee_j R(a_j), \quad R(a_j) = \vee_k [a_j = v_k] \quad s.t. \kappa_R(D) = 1.0.$$

It is notable that this definition is an extension of exclusive rules and this total covering rule can be written as:

$$d \rightarrow \vee_j \vee_k [a_j = v_k] s.t. \kappa_{[a_j = v_k]}(D) = 1.0.$$

Let $S(R)$ denote a set of attribute-value pairs of rule R. For each class d, let $R_{pos}(d)$, $R_{ex}(d)$, and $R_{tc}(d)$ denote the positive exclusive rule and total covering rules, respectively. Then

$$S(R_{ex}(d)) \subseteq S(R_{tc}(d)),$$

because a total covering rule can be viewed as an upper approximation of exclusive rules. It is also notable that this relation will hold in the relation between the positive rule (inclusive rule) and total covering rule. That is,

$$S(R_{pos}(d)) \subseteq S(R_{tc}(d)).$$

Thus, the total covering rule can be viewed as an upper approximation of inclusive rules. This relation also holds when $R_{pos}(d)$ is replaced with a probabilistic rule, which shows that total covering rules are the weakest form of diagnostic rules.

In this way, rules that reflect the diagnostic reasoning of medical experts have sophisticated background from the viewpoint of set theory. Especially, the rough set framework provides a good tool for modeling such focusing mechanisms. Our future work will investigate the relation between these three types of rules from the viewpoint of rough set theory.

9.9 Conclusions

In this chapter, the characteristics of two measures, classification accuracy and coverage, are discussed, which show that both measures are dual and that accuracy and coverage are measures of both positive and negative rules, respectively. Then an algorithm for induction of positive and negative rules is introduced. The proposed method is evaluated on medical databases. The experimental results have demonstrated that the induced rules are able to correctly represent experts' knowledge. We also demonstrated that the method can discover several interesting patterns.

References

9.1 Adams, R. D., and Victor, M., *Principles of Neurology*, 5th ed, McGraw-Hill, New York, 1993.

9.2 Buchnan, B. G., and Shortliffe, E. H.,(eds.), *Rule-Based Expert Systems*. Addison-Wesley, 1984.

9.3 Matsumura, Y., Matsunaga, T., Hata, Y., Kimura, M., and Matsumura, H., Consultation system for diagnoses of headache and facial pain: RHINOS, *Medical Informatics* **11**: 145–57, 1988.

9.4 Michalski, R. S., Mozetic, I., Hong, J., and Lavrac, N., The multi-purpose incremental learning system AQ15 and its testing application to three medical domains, *Proc. 5th National Conference on Artificial Intelligence*, AAAI Press, Palo Alto, CA, 1041–5, 1986.

9.5 Pawlak, Z., *Rough Sets*, Kluwer Academic Publishers, Dordrecht, 1991.

9.6 Pawlak, Z., Rough Modus Ponens, in *Proc. International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems 98*, Paris, 1998.

9.7 Quinlan, J. R., *C4.5–Programs for Machine Learning*, Morgan Kaufmann, Palo Alto, CA, 1993.

9.8 Skowron, A., and Grzymala-Busse, J., From rough set theory to evidence theory, in Yager, R., Fedrizzi, M., and Kacprzyk, J.(eds.), *Advances in the Dempster-Shafer Theory of Evidence*, pp.193–236, John Wiley & Sons, New York, 1994.

9.9 Shavlik, J. W., and Dietterich, T. G. (eds.), *Readings in Machine Learning*, Morgan Kaufmann, Palo Alto, CA, 1990.

9.10 Tsumoto, S., and Tanaka, H., Automated discovery of medical expert system rules from clinical databases based on rough sets, in *Proc. 2nd International Conference on Knowledge Discovery and Data Mining 96*, AAAI Press, Palo Alto, CA, 63–9, 1996.

9.11 Tsumoto, S., Modelling medical diagnostic rules based on rough sets, in Polkowski, L., and Skowron, A. (eds.), *Rough Sets and Current Trends in Computing*, Lecture Note in Artificial Intelligence **1424**, 1998.

9.12 Tsumoto, S., Automated extraction of medical expert system rules from clinical databases based on rough set theory, *Journal of Information Sciences*, **112**, 67–84, 1998.

9.13 Tsumoto, S., Extraction of experts' decision rules from clinical databases using rough set model, *Intelligent Data Analysis*, 2(3), 1998.

9.14 Tsumoto, S., Medical diagnostic rules as upper approximation of rough sets, *Proc. FUZZ-IEEE2001*, 2001.

9.15 Ziarko, W., Variable precision rough set model, *Journal of Computer and System Sciences*, **46**:39–59, 1993.

Index