THE HONG KONG
POLYTECHNIC UNIVERSITY
香港理工大學

# Soft Computing Techniques for Case Knowledge Extraction in CBR System Development

**LI YAN**

A Thesis Submitted in Partial Fulfillment

of the Requirements for

the Degree of Doctor of Philosophy

Department of Computing

**The Hong Kong Polytechnic University**

**October 2005**

## CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

Signature:

Name of student: Li Yan

Abstract of the thesis titled "Soft Computing Techniques for Case Knowledge Extraction in CBR System Development", Submitted by LI YAN for the degree of Doctor of Philosophy at the Hong Kong Polytechnic University, October 2005.

# Abstract

The performance of a case-based reasoning (CBR) system depends on its problem-solving quality, efficiency and competence. In a case base, a case can be defined as a piece of contextual and specific knowledge. The more the cases, the better the competence (coverage) of the problem domain, and therefore larger CBR systems tend to provide better solutions than the smaller ones. However, this is not always true because not all the cases collected in the system are useful for problem solving. For example, cases may be in conflict with each other; many cases may be redundant because of their close similarity; some cases may be noises in the system because they are not offering any help in the problem solving, and sometimes may even cause confusion. Another important aspect of CBR system is its efficiency (or speed) in providing helps. The purpose of this research is to examine closely these two aspects, and develop feasible computational techniques that will facilitate the development of CBR systems. This research question leads us to think deeply what constitute the problem solving ability of a CBR system; and also how to strike a balance between efficiency and problem-solving quality. Furthermore, in many real world situations, data and information collected are always incomplete, uncertain and vague, thus, the use of soft computing principles to achieve tractability, robustness and low solution cost is inevitable.

Having the above understanding in mind, we then built up a set of soft computing based techniques for the extraction of case knowledge from data. They aim at (i) removing the redundancy and noises; (ii) reducing the size of the case base; and (iii) preserving the problem solving ability (or competence in CBR terminology). The developed algorithms deal with the processes of feature selection and reduction; similarity learning among features; case selection and case generation; and competence model development. Specific concepts and techniques, like approximate reducts; GA-based case-matching; redefined case coverage and reachability measurement; boundary cases with NN guiding principle; fast rough set-based feature reduction; rough LVQ based case generation; fuzzy integral-based case base competence model, are developed, tested and compared with traditional methods such as KPCA and SVM. The experimental results are very promising, and support our objective of trying to develop a compact and competent CBR system through case knowledge extraction.

**Keywords:** Case knowledge; soft computing; feature reduction; case selection; case generation; similarity measure; competence model; case coverage; case reachability.

# Acknowledgements

The completion of this thesis would not have been possible without the help of many professors and friends, to whom I would like to express my heartfelt appreciation.

First, I would like to express my deepest thanks and gratitude to my supervisor, Dr. Simon Chi Keung Shiu. I would like to thank him for his kind supervision and continuous support during my PhD study at the Hong Kong Polytechnic University. He is a very kind and optimistic person with positive attitudes even facing difficult situations and challenges. Each time when I feel depressed during my study, he will come and encourage me to be more confident in dealing with the problems. I have learned a lot from him, both in terms of the scientific knowledge as well as the proper way to conduct research work. The experience as his student in these three years is of great benefit to me for the rest of my life.

Another excellent and distinguished person I would like to thank is my co-supervisor, Prof. Sankar Kumar Pal. I would like to express my deepest gratitude to him for his valuable advices during my study. I learned a lot from his rigorous and strictness attitude in doing research, and his almost religious devotion for that. Every time when he read my manuscript, he could immediately find out some important aspects for further improvement. His acuminous insight and guidance is of great help for my study.

Next, I want to thank Prof. Xizhao Wang and Prof. Minghu Ha, who are the supervisors of my master degree study in the Department of Computer and Mathematics, Hebei University. They have given me many useful suggestions and comments about my research project.

I would like to thank all the members of the CBR research group in the department: Feng Zhang, Ben Niu and Lan Zhen Yang. I appreciate very much their feedbacks, discussions, assistances, advices, and supports.

Thanks also to the board of examiners who spent their time and effort in assessing this research work and provided many good suggestions for me to improve the thesis. They are Dr. Korris F. L. Chung from Department of Computing, the Hong Kong Polytechnic University, Prof. Qiang Yang from the Department of Computer Science, the Hong Kong University of Science and Technology, and Prof. Kalyanmoy Deb from the Department of Mechanical Engineering, Indian Institute of Technology Kanpur, India.

I would like to thank many of my colleagues: Meng Wang, Yun Xiao, Jian Yang, Gang Yao, Bo Feng, and Yongjie Zheng, who share the pleasure of life with me in the Hong Kong Polytechnic University.

Last, but not the least, I wish to express my deepest appreciation to my parents for their endless love and unwavering support. Especially, I thank my husband, Dr. Qinghua Zhou, for his love, support and encouragement.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

The purpose of this research is to study case knowledge extraction in the context of building case-based reasoning systems, and develop soft computing based techniques to construct both compact and competent case knowledge bases. This chapter explains the motivation and objectives of this research, introduces different tasks for case knowledge extraction and presents an overview of this thesis.

The organization of this chapter is as follows: Section 1.1 explains the importance of conducting research in case knowledge extraction, and describes the objectives and the main tasks involved. Section 1.2 outlines the organization of this thesis. Section 1.3 summarizes the contribution of this research work. The publications arising from this research are listed in Section 1.4. Finally, Section 1.5 concludes this chapter.

## 1.1 Motivation and Objectives

Case-based Reasoning (CBR) is a reasoning methodology that is based on prior experience and examples. It includes retaining a memory of previous problems and their solutions, and solving new problems by reference to this knowledge. Generally, a CBR reasoner will be presented with a problem, and it then searches its memory of past cases (called the case base) and attempts to find a case or multiple cases that most closely match the current case. In some situations, the found cases need to be adapted to meet the requirement of the new problems. Since the adaptation process is depended on the domains and problems in question, we do not consider this issue in this research. CBR systems usually require significantly less knowledge acquisition than rule-based systems since they involve the collection of a set of past experiences without requiring the extraction of a formal domain model from these cases. CBR systems have been widely used in many applications: design, planning, prediction and classification, knowledge inference and evaluation, and many others.

Case-based reasoning (CBR) systems are built completely based on the case bases which represent and store the previously solved problems. These case bases are considered as the knowledge bases which provide the basis for the use of the CBR systems by either a user or a system. The computational process of CBR always assumes that the involved case bases are adequate and useful. This is not necessarily true in many real-life applications of CBR systems, in which the case bases may be incomplete or contain redundancy and noisy cases. These case bases will degrade the performance of CBR systems with respect to three main criteria: problem-solving quality, efficiency and completeness.

In the following, we explain the three performance criteria and how the characteristics of the cases being collected in the case bases affect them.

The problem-solving quality of a CBR system is the average quality of the proposed solutions, which can be usually described by accuracy and the required adaptation effort. The accuracy is the percentage of the problems which can be successfully solved. The required adaptation effort is the cost for modifying the proposed solutions derived from the retrieved case(s) to solve the problems. In other words, the basic qualification of a CBR system is whether the proposed solution can be used to solve the new problem. The existence of redundant and noisy cases will cause problem in the case retrieval, i.e., the wrong case or cases may be retrieved even though the correct case is contained in the case base. As a result, the accuracy will decrease and the required adaptation effort will increase.

Secondly, the problem-solving efficiency is also important, which can be defined as the average time for solving a problem. With increasing storage of cases (i.e., problems), the case bases tend to become larger and larger which will slow down the case retrieval speed.

Finally, the completeness of a case base is a measure of its problem solving coverage of all the potential problems. It can be measured using the concept of competence [Smyt 1995 1998a 1998b], i.e., the range of problems that the case base can successfully solve. According to Smyth and McKenna, a case can be used to solve a target problem, if and only if the case must be retrieved for the target and it must be possible to adapt its solution to solve the target problem. The competence of a case base can be described by two important competence properties: the coverage set and the reachability set. The coverage set of a case is the set of all target problems that this case can be used to solve. On the other hand, the reachability set of a target problem is the set of all cases that can be used to solve it. Cases with large coverage sets are more important because they can solve many other problems and therefore should solve many of the future target problems. Cases with small reachability sets are more important because they represent regions of the target problem space that are difficult to solve.

The two performance criteria of accuracy and competence are closely related. Generally speaking, if more cases can be successfully solved by a case base, the accuracy will increase. Therefore, for a given case base, larger competence achieves higher accuracy. Therefore, the desirable characteristics of case bases are small in size, large in competence, and contain no redundancy and noises.

In a case base, each case is a piece of contextual knowledge, therefore they (i.e., all the collected cases) cover many specific situations in the given problem domain. In contrast to the specificity of cases, the other extreme type of knowledge representation form is using abstract models to capture the generality of knowledge, e.g., mathematical models, which represent abstract rules that are normally true in the problem domain. General knowledge model allows economic storage and therefore provides high problem-solving efficiency, and it contains no redundancy, inconsistency and noise. However, to develop general knowledge model usually requires much training effort, and in many situations the model can not be constructed successfully. This may be due to the non-linear characteristics of the problem domain, or the limitation of the training algorithms.

Comparing with constructing general knowledge models, CBR systems do not need training effort and can cover more possible problems as well as provide better understandability. Therefore, they are more operational (and with higher acceptance) by non-experts [Kolo 1993] [Pal 2004a]. The disadvantages of using cases are that they require larger storage space and much processing time in case matching and retrieval. Case bases tend to grow larger and larger very quickly if all incoming cases are also stored. However, not all the cases collected in the system are useful for problem solving. For example, cases may be in conflict with each other; many cases may be redundant because of their close similarity; some cases may be noises in the system because they are not offering any help in the problem solving, and sometimes may even cause confusion.

In this research, we attempt to study the specificity and abstractness of knowledge representation, and especially those principles that are relevant and applicable to CBR systems. The objective is to develop a knowledge representation scheme for CBR system development. This scheme should reduce the size of the case bases to improve the efficiency and maintain the understandability and competence to preserve the operational ability and problem-solving accuracy. Therefore, the central focus of this research is the concept of "case knowledge" and the techniques of how to extract "case knowledge". As we have mentioned above, there are two traditional forms of case knowledge: (1) cases, which is too specific and loses the generalization power; and (2) mathematical models, which will lose much detail information about the specific nature of the problem domain, and making it less applicable in real world.

In this research, we attempt to re-define the meaning this "case knowledge", and provide feasible techniques to extract them. This "case knowledge" can be considered as something in between from specific cases to generalize abstract model. For example, (i) after removal of irrelevant features and selection of representative cases from the original data, this collection can be regarded as the "case knowledge"; (ii) after generalizing a number of cases into a prototypical case, and a selection of a set of prototypical cases can also be regarded as the "case knowledge".

Here we give an example to illustrate the previously mentioned different knowledge representation forms, i.e., cases, case knowledge, and abstract knowledge. In Table 1.1, each row is a specific case which has 5 features. Among these features, $x$, $y$, $a$, $b$ are the conditional attributes, and $f$ is the decision attribute. Using some learning methods such as neural networks, the relationship of $f$ and the conditional attributes can be discovered as two functions: $f_1 = x^2 + y$, and $f_2 = x^{1/2} + y^{1/2}$. These two functions are considered to be the abstract knowledge which represented by mathematical models. According to $f_1$ and $f_2$, the decision attribute value is independent to features $a$ and $b$. Table 1.2 represents the case knowledge extracted from Table 1.1. There are fewer features and cases in this table than those in Table 1.1. The reduction of the two features $a$ and $b$ is consistent with the discovered mathematical models. The cases in Table 1.2 are considered to be the most representative cases.

Table 1.1 Illustrative example of a case base

| *ID* | *f* | *x* | *y* | *a* | *b* |
|---|---|---|---|---|---|
| 1 | 5 | 2 | 1 | 1 | 2 |
| 2 | 3 | 1 | 2 | 1 | 1 |
| 3 | 1.25 | 0.5 | 1 | 2 | 1 |
| 4 | 1.5 | 1 | 0.5 | 10 | 0 |
| 5 | 5 | 2 | 1 | 3 | 4 |
| 6 | 0.75 | 0.5 | 0.5 | 100 | 50 |
| 7 | 4 | 4 | 4 | 0 | 1 |
| 8 | 3.5 | 2.25 | 4 | 3 | 2 |
| 9 | 5 | 9 | 4 | 2 | 1 |
| 10 | 3.5 | 2.25 | 4 | 1 | 1 |
| 11 | 5 | 9 | 4 | 6 | 5 |

The discovered mathematical model:

$$f_1 = x^2 + y, \text{ and}$$

$$f_2 = x^{1/2} + y^{1/2}.$$

Table 1.2 Extracted case knowledge base

| *ID* | *f* | *x* | *y* |
|---|---|---|---|
| 1’ | 5 | 2 | 1 |
| 2’ | 3 | 1 | 2 |
| 3’ | 1.25 | 0.5 | 1 |
| 4’ | 1.5 | 1 | 0.5 |
| 5’ | 0.75 | 0.5 | 0.5 |
| 6’ | 4 | 4 | 4 |
| 7’ | 3.5 | 2.25 | 4 |
| 8’ | 5 | 9 | 4 |

It can be seen from this example that, the mathematical model requires the least storage and therefore can achieve the highest efficiency among these three knowledge representation methods. Since the functions can accurately describe the relationship among the *f* values and the values of *x*, *y*, *a* and *b*, the mathematical model can also achieve high accuracy. Unfortunately, in most of the practical situations, it is very difficult to discover such functions because of the highly non-linear properties of the problems and the non-trivial training cost in the required learning process. Furthermore, compared with the specific cases, the abstract knowledge representation is more difficult to be understood by non-experts. Case knowledge is the compromise method of specific cases and the abstract knowledge. This research work involves developing different techniques to extract case knowledge from the raw case bases.

In the following chapters, we will illustrate how this can be done. Before that, we would like to summarize the case knowledge extraction processes in Fig. 1.1.

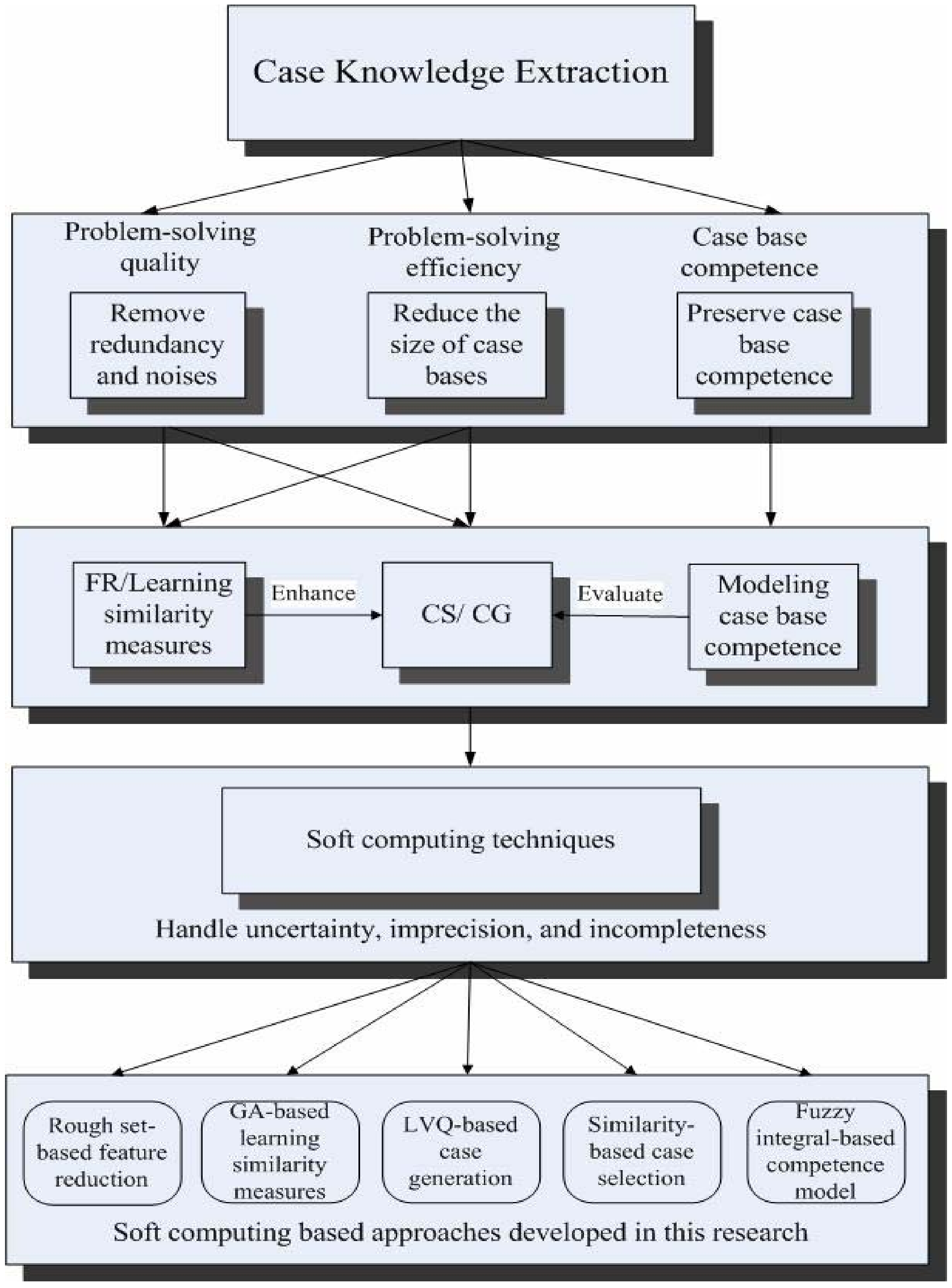
Case Knowledge Extraction
Problem-solving quality
Problem-solving efficiency
Case base competence
Remove redundancy and noises
Reduce the size of case bases
Preserve case base competence
FR/Learning similarity measures
Enhance
CS/ CG
Evaluate
Modeling case base competence
Soft computing techniques
Handle uncertainty, imprecision, and incompleteness
Rough set-based feature reduction
GA-based learning similarity measures
LVQ-based case generation
Similarity-based case selection
Fuzzy integral-based competence model
Soft computing based approaches developed in this research

Fig. 1.1 The methodology of case knowledge extraction

For removing redundancy and noises, reducing sizes and preserving the competence, we consider the following four tasks: (1) feature reduction (FR), (2) learning similarity measures, (3) case selection (CS) and case generation (CG); and (4) competence model development.

The first two tasks aim to reduce the redundancy contained in the features and thus avoid the negative effect of the non-informative features. Based on the reduced feature set and learned similarity measures, we can remove those redundant and noisy cases, thus resulted a better data clustering performance. Since the dimensionality of the case bases has been reduced, the problem-solving efficiency is also improved.

The next two tasks, i.e., CS and CG, are performed to obtain a smaller set of cases. While CS is to identify and remove redundant and noisy cases, CG generates new prototypical cases by merging multiple cases into one case and extracting the central cases from different clusters.

Finally, in order to assess the completeness of the case bases, we develop competence model to describe and predict the range of problems that the CBR systems can successfully solve.

There are much related research works which address the previously mentioned objectives and tasks. For example, M. A. Kramer [Kram 1991] proposed the method of using Kernel Principle Component Analysis (KPCA) to reduce the dimensionality of the case bases through discovering and extracting the uncorrelated features; K. Chidananda Gowda and E. Diday [Gowd 1992] presented a new similarity measure which took into account the “position”, “span” and “content” of both numerical and symbolic features; V. N. Vapnik [Vapn 1998] developed the Support Vector Machine algorithm which can be used for case generation; D. R. Wilson and L. Dennis [Wils 1972] used the nearest neighbor principle to detect and remove noisy cases. These together with other related work will be briefly reviewed in Chapter 2.

Most of the available work, however, does not consider the situations in which the case bases are incomplete and contain uncertainty and imprecision. In this research, soft computing based techniques are used to deal with such situations. In general, soft computing is a consortium of computing tools and techniques, including fuzzy logic (FL), neural network theory (NN), evolutionary computing (EC), rough set theory (RST), chaos theory, and parts of learning theory. The individual tool can be used synergistically, not competitively, in enhancing the application domain of the other. As mentioned in [Pal 2004a], the significance of soft computing methodologies can be described as: "The purpose is to develop flexible information-processing systems that can exploit the tolerance for imprecision, uncertainty, approximate reasoning, and partial truth in order to achieve tractability, robustness, low solution cost, and close resemblance to human decision making". Therefore, in our research, we have developed a number of soft computing techniques for: (1) feature reduction (FR), (2) learning similarity measures, (3) case selection (CS) and case generation (CG); and (4) competence model development. These are briefly explained in the following paragraphs.

For the task of FR, we focus on rough set-based feature reduction. Rough sets allow the most informative features to be detected and then selected through the reduct computation. However, the traditional rough set-based feature reduction [Pawl 1991] requires much computational effort to obtain the crisp reduct. To improve the efficiency of feature reduction, we introduce a new concept of approximate reduct, which is a generalization of the crisp reduct and can be obtained very quickly. A set of informative features are selected through finding an approximate reduct, which can be considered as an approximation of the optimal feature set (i. e., the minimum set which can preserve the discernibility of the original case base). To demonstrate its effectiveness, the developed feature reduction method is compared with the widely used Kernel Principle Component Analysis (KPCA). KPCA is an unsupervised learning method and can be used to transform the cases to a high dimensional feature space and obtains a set of transformed features rather than a subset of the original features. The experimental results show that KPCA needs much more processing time, including the time for training and transforming the feature spaces, in order to achieve a slightly higher problem-solving accuracy.

The similarity measures used in case matching and case retrieval are crucial in identifying the most similar case or cases for a given problem (an unseen case). Most of the existing related work to learning feature measures, such as the feature weighting methods, is used on numerical features instead of symbolic features. One type of symbolic features: nominal feature is even more problematic because they consist of totally unordered feature values. In most similarity measures for nominal features, if two feature values are the same, the similarity value is defined as one; otherwise, the similarity values is defined as zero. This similarity measure defined on the nominal features divides a case base into coarse-grained granules, in which all the different feature values are considered to be totally dissimilar to each other. In this research, we generalize the domain of similarity values from {0, 1} to [0, 1] through supervised learning using genetic algorithms. Here the genetic algorithm provides a natural search until the accuracy obtains its maximum. As a result, much finer similarity measures can be obtained which are then used in case retrieval and classifying unseen cases. Some real-life data sets are used to conduct experiments and the results demonstrate that the classification accuracy is improved and better clustering performance is also achieved.

Using the reduced features and learned similarity measures, case selection is implemented to select a subset of cases through identifying and removing the redundant and noisy cases. In our methods, the importance of each case is evaluated by the similarity measures, case coverage and case reachability, and the k-nearest neighbor (k-NN) principle. The concepts of case coverage and reachability are redefined by specifying what is meant by "a case covers another case", which is based on the boundary cases and the similarity computation. The reachability set of a case can be derived by the definition of the coverage set. The larger the coverage set and the smaller the reachability set, the more important of the given case. Based on these concepts, a subset of cases can be selected which can cover the whole original case base. This subset can be considered as the optimal one which has the minimum number of cases while maintaining the original competence.

The case selection method performs well when the case bases do not contain noisy cases. This is because the existence of noisy cases will give wrong identification of boundary cases which will affect the computations of case coverage and reachability. It is because a noisy case can be easily recognized as a boundary case. Therefore, if the case bases contain noisy cases, we should identify these cases first using k-NN principle, and remove them from the case bases. Then the cases are evaluated by the coverage set and reachability set and the most important cases are selected until they cover all the cases in the original case base. Our case selection methods are compared with Wilson Editing and support vector machine (SVM) and show much better performance in terms of accuracy, case selection speed and case retrieval time.

Case generation is to extract case knowledge by generating a set of prototypical cases instead of selecting a subset of cases from the original case base. The case generation methods modify or combine or merge the original cases to form new prototypical cases. The generated cases are considered to be representative cases which form the case knowledge base. In this research, we propose a case generation method by integrating fuzzy sets, rough sets, and learning vector quantization (LVQ). LVQ is the supervised version of the Self-organizing map (SOM) which is a clustering tool of high-dimensional data. Compared with SOM, LVQ is more suitable to be used in classification problems and it is more robust to redundant features and cases as well as to the change of learning rate. Since the learning process of LVQ is affected by the similarity measure that is used for the clustering, we should identify the most informative features first. This is done by the rough set-based feature reduction method using our newly defined concept of approximate reduct. If the feature values are numeric, fuzzy sets are used to discretize the feature space for facilitating the rough set-based approximate reduct generation. Some experiments are conducted to evaluate the classification accuracy, the storage space, case retrieval time and clustering performance in terms of intra-similarity and inter-similarity.

The issue of building competence model arises from the area of case base maintenance (CBM), which implements policies of revising the organization and contents of the case bases to facilitate the future reasoning for a particular set of performance objectives. Since competence is one of the most important problem-solving criteria, there was a spurt of interest in competence-guided CBM among many researchers. Smyth and Mckenna [Smyt 1998a] developed a case base competence model, in which a new concept of competence group is introduced. For a given competence group, the group coverage is computed based on the size of the competence group, the case density in this group, and the case coverage and reachability. The overall case base coverage is obtained by simply sum up the group coverage. This model assumes that the cases are uniformly distributed and there are no overlaps among the competence groups. However, in real life, the case distribution is often non-uniformed, and the boundaries of the competence groups are not crisp. In such situations, the model is not a good predictor of case base competence.

In this study, we use fuzzy integral to deal with the possible interaction among the competence groups and our model does not assume uniform case distribution. Fuzzy integral is a powerful mathematical tool which is based on a variety of fuzzy measures (also called non-additive measures). The traditional measures on a set of objects must satisfy the additive property, which assumes that there does not have any interaction among the objects. In contrast, fuzzy measures are non-additive and therefore can be used to deal with various kinds of interactions among objects. In our developed model, the fuzzy measure is determined first and then used to handle the possible interaction among different competence groups. Based on the fuzzy measure, the fuzzy integral is used to compute the overall competence of a given case base. Our competence model has better ability to deal with the overlaps among competence groups and can describe the case base competence more accurately.

To make the research work more complete, we apply the fuzzy integral-based competence model to guide a query dispatching application in collaborative CBR systems. We assume that there are multiple CBR systems distributed in different physical locations, each of which can solve the coming problems independently, but with different degree of competence. There are many possible dispatching strategies when a query (or an unseen case) occurs. We propose three policies based on the competence model and proved to be successful. The main idea is to select the most competent CBR system and send the current query to this system.

## 1.2 Experimental Methodology and Success Metrics

In this research work, some experiments are conduced to demonstrate the effectiveness of these techniques in terms of several given metrics.  This section firstly explains the experimental methodology in two aspects: (1) the used data sets; (2) the benchmark methods which are used to be compared with our techniques. The success metrics are then described which are used as evaluation criteria.

We mainly use the real life data from UCI [Hett 1998] in this research, which are of different size and have different types of features. For example, *House-votes-84* has 17 Boolean valued features and 435 cases; *Multiple Features* has 10 numerical features and 2000 cases; and *Mushroom* database contains 23 nominally valued features and 8124 cases. Most of these data has relatively small number of features. Therefore, another data set from Reuters21578 [Lewi 1999] is also used in this research work, which shows the characteristic of high dimensionality. The purpose of using variety of data sets is to better reflect the performance of the proposed techniques through the experimental results.

As mentioned in Section 1.1, after applying the proposed techniques on these given data sets, they can be reduced to some smaller case bases with fewer features and cases. The reduced case bases are considered as the extracted case knowledge bases. Some given metrics are then used on these case knowledge bases to test the performance of the proposed techniques. Here we assume the bottom-line of the performance is that using the original data sets (i.e., without being processed by the case knowledge extraction techniques). Throughout this research, we focus on classification problems. Therefore, one basic comparison required to be made is the classification performance with the original data and the extracted case knowledge.

Besides, some other benchmark methods are employed to demonstrate the effectiveness of our techniques. They are summarized as follows. For details, please refer to the related chapters.

Kernel principle component analysis (KPCA) is one of the widely used techniques for feature selection. It is a non-linear version of PCA, which can discover the dominant nonlinear features of the original data. This technique is used as a benchmark to evaluate the proposed rough set-based feature reduction methods.

On the other hand, Wilson Editing is a traditional case selection method built on the basis of k-NN principle. It has been shown to be effective to detect and remove noisy cases in [Wils 1972][Gate 1972][Ritt 1975][Tome 1976]. In the experiments on case selection, the proposed CS methods are compared with Wilson Editing method. Support vector machine (SVM) or SVM ensemble are another group of promising techniques to reduce the size of the case bases. In this research, SVM ensemble is used together with KPCA to demonstrate the performance of the combination of our FR and CS methods.

For the task of case generation, three methods are used as benchmarks: (1) Random method, which randomly selects a given number of cases; (2) Self-Organizing Map (SOM) algorithm, which is an unsupervised competitive algorithm to generate prototypical cases; (3) Learning vector quantization (LVQ), which is a supervised version of SOM.

In these experiments, several success metrics are predefined to reflect different aspects of the performance.

(1) Storage. It describes the size of case base in term of both the number of features and cases. In this research, this metric reflects the compactness of the extracted case knowledge bases after applying the proposed techniques. The storage values of the original case base with respect of features and cases are assumed to 1 (or 100%).

Therefore, the storage of the reduced case base is less than 1. The less the storage is required, the more effective the techniques for building compact case bases.

(2) Accuracy. It is one of the most often used measures in performance evaluation, especially in classification problems. Generally, the classification accuracy is the percentage of the unseen cases which can be correctly classified, i.e., the ratio of the number of correctly classified unseen cases to the total number of unseen cases. Improving the accuracy is one of the most important objectives for developing the case knowledge extraction techniques.

(3) Efficiency. The efficiency of a given technique can be described by the speed or cost time of implementing this technique. For a given classifier, its efficiency can be defined as the average time to predict the class label of an unseen case by using this classifier. In this research, we use the improved efficiency after applying the proposed case knowledge extraction techniques to demonstrate their advantages. Here the improved efficiency is computed as the saved time in implementing the classifiers, i.e., the difference between the required time to classify an unseen case using the original case base and that using the reduced case base after applying the proposed case knowledge extraction techniques.

(4) Clustering performance. This metric can be described by the intra-similarity and inter-similarity of the case base in question. It is expected that the intra-similarity (i.e., the average similarity between two cases in the same class) is as large as possible whereas the inter-similarity (i.e, the average similarity between two cases in different classes) is as small as possible.

## 1.3 Structure of the Thesis

This chapter introduces the motivation and objectives of the research work, and also explains the four different tasks, namely (1) feature reduction (FR), (2) learning similarity measures, (3) case selection (CS) and case generation (CG); and (4) competence model development, to achieve these objectives. The remaining chapters of the thesis are organized as follows.

Chapter 2 briefly presents the literature review of the relevant topics and provides some necessary preliminary knowledge of rough set theory which is used in the development of the techniques for case knowledge extraction.

Chapter 3 develops a rough set-based FR which is fast and effective. The concept of a reduct is generalized to the approximate reduct. Correspondingly, some primary concepts in rough set theory, such as dispensable and indispensable attribute, reduce and core, are also extended. Using these modified concepts, a fast rough set-based method is developed to find the approximate reduct. Some experiments are conducted using several real life data sets to evaluate the performance of this developed FR.

Chapter 4 presents a GA-based supervised learning process to determine similarity measures of nominal features. The domain of the similarity values is extended from the traditional {0, 1} to [0, 1] to obtain a fine-grained measure of nominal features. An artificial data set and the Balloons database are used to demonstrate the effectiveness of this method.

Chapter 5 provides several different CS strategies based on the concepts of case coverage and reachability, the similarity measure, and the k-NN principle. They are also combined with the feature reduction approach developed in Chapter 3 to extract case knowledge through reducing both the dimensionality and the size of the original case base.

Chapter 6 describes a CG technique where rough sets and learning vector quantization (LVQ) are used to extract the prototypical cases from the original case base. This supervised learning uses fuzzy sets and rough sets to pre-process the given case base and then LVQ is applied to generate the representative cases. The clustering performance is improved and the case retrieval time is reduced based on the generated prototypical cases.

Chapter 7 builds a fuzzy integral-based case base competence model which can be used to describe the range of problems that the given case base solves. Compared with the previously related work, the case distribution is taken into account and the interaction among different competence groups is reflected in the determined fuzzy measures.

Chapter 8 presents an application of fuzzy integral-based competence model to query dispatching in the collaborative CBR systems. The competence model is used to guide the query dispatching strategies which aim to improve the efficiency and quality of the case retrieval.

Chapter 9 provides a summary evaluation of the developed soft computing based case knowledge extraction techniques, and then gives the conclusions and some possible future work.

## 1.4 List of Contributions

In summary, we have developed the following soft computing based techniques for case knowledge extraction.

1. A new rough set-based feature reduction method, which is based on the concept of approximate reduct, is developed to quickly identify and remove the non-informative features.

2. A new GA-based supervised learning process is developed and used to determine the similarity measures for nominal features.

3. A new case selection method is developed using the concept of case coverage, case reachability, similarity measures and the nearest neighbor principle to select the approximate optimal set of cases from the original case bases.

4. A new rough learning vector quantization (LVQ) method is developed and used to extract the most representative prototypical cases in case generation.

5. A new fuzzy integral-based case base competence model is built to describe the completeness of a given case base. The built fuzzy integral-based competence model is applied in a collaborative CBR system environment to develop query dispatching policies.

## 1.5 List of Publications

Journal Papers:

1. Li Y., Shiu S. C. K, and Pal S. K., Combining Feature Reduction and Case Selection in Building CBR Classifiers. *IEEE Transactions on Knowledge and Data Engineering* (*TKDE*), vol. 18, no. 3, pp. 415-429, 2006.

2. Li Y., Shiu S. C. K, Pal S. K., and Liu J. N. K. A Rough Set-based Case-Based Reasoner for Text Categorization. *International Journal of Approximate Reasoning*, vol. 41, pp. 229-255, 2006.

3. Shiu S. C. K, Li Y., Zhang F. A Fuzzy Integral Based Query Dispatching Model in Collaborative Case-Based Reasoning. *Applied Intelligence*, vol. 21, no. 3, pp. 301-310, 2004.

Book Chapter:

4. Li Y., Shiu S. C. K, Pal S. K, Combining Feature Reduction and Case Selection in Building CBR Classifiers, will be published in *Case-Based Reasoning in Knowledge Discovery and Data Mining*, edited by Sankar Pal，David Aha.

Conference Papers:

5. Li Y., Shiu S. C. K., Pal S. K., Liu J. N. K., Learning Similarity Measure for Nominal Features in CBR Classifiers, in Proceeding of the *First International Conference on Pattern Recognition and Machine Intelligence* (*PReMI 2005*), pp. 780-785, Indian Statistical Institute, Kolkata, India, December, 18-22, 2005.

6. Yan Li, Simon C. K Shiu, Sankar K. Pal, James N. K. Liu, Rough Learning Vector Quantization Case Generation for CBR Classifiers, in Proceedings of the *Tenth International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing (RSFDGrC 2005)*, vol. 2, pp. 128-138, Regina, Canada, August 31-September 3, 2005.

7. Li Y., Shiu S. C. K, Pal S. K, and Liu J. N. K. A Fuzzy-Rough Method for Concept-based Document Expansion, in *Proceedings of the Fourth International Conference on Rough Sets and Current Trends in Computing* (*RSCTC 2004*), pp. 699-707, Uppsala , Sweden, June 1-5, 2004.

8．Li Y., Shiu S. C. K., Pal S. K., and Liu J. N. K. A Rough Set-Based CBR Approach for Feature and Document Reduction in Text Categorization, in *Proceedings of the Third International Conference on Machine Learning and Cybernetics* (*ICMLC 2004*), vol. 4, pp. 2438-2443, Shanghai, China, Aug. 26-29, 2004.

9. Li Y., Shiu S. C. K., Pal S. K., Liu J. N. K. Case-base Maintenance Using Soft Computing Techniques, in *Proceedings of the Second International Conference on Machine Learning and Cybernetics* (*ICMLC 2003*), vol. 3, pp. 1768-1773, Xi'an, China, Nov. 2-5, 2003.

10. Li Y.**,** Wang X. Z., Ha M. H. On-line Multi-CBR Agent Dispatching, in *Proceedings of the Second International Conference on Machine Learning and Cybernetics* (*ICMLC 2003*), vol. 4, pp. 2071-2075, Xi'an, China, Nov. 2-5, 2003.

## 1.6 Summary

In this chapter, we have firstly presented the motivation and objectives of the research work and then explained the different tasks for case knowledge extraction. This is followed by a description of the organization of the thesis. The list of contributions and publications derived from this research is given at the end of this chapter.

# Chapter 2

# Basic Knowledge and Literature Review

This chapter firstly provides a brief literature review of case knowledge extraction that is relevant to our four tasks of (i) feature reduction, (ii) learning similarity measures, (iii) case selection and case generation, and (iv) the competence model development. This is followed by an introduction of the basic concepts in rough set theory because rough sets play an important role in the developed soft computing based techniques.

## 2.1 Literature Review of Case Knowledge Extraction

As mentioned in [Kolo 1993], cases represent specific knowledge tied to specific situations, and therefore represent knowledge at an operational level. They make explicit how a task was carried out or how a piece of knowledge was applied or what particular strategies for accomplishing a goal were used. A case can be therefore defined as a piece of specific contextual knowledge corresponding to a certain situation. In many real world applications, a case base often has hundreds or even thousands of features and cases, which requires much processing time in case retrieval. The non-informative features and redundant cases, which are inevitably contained in a case base, may degrade problem-solving quality and efficiency of the CBR systems. Therefore, it is highly desirable to extract case knowledge from the original case bases for CBR applications.

In the following, we will give a brief review of the related work to case knowledge extraction including those soft computing based methods. The following sections are organized according to the previously mentioned four tasks which address different aspects of the overall research objective.

### 2.1.1 Feature Reduction

The purpose of feature reduction (FR) is to reduce the irrelevant features and emphasize the importance of the informative features. This is done by detecting and removing the non-informative features or by extracting the most informative features.

The related work to FR includes feature transformation, feature selection, and their combinations [Liu 1998]. Feature transformation can be further divided into feature extraction and construction, which transform the feature spaces and generate a new set of features. On the other hand, feature selection simplifies the feature spaces and selects a subset of the original features.

Principle component analysis (PCA) [Joll 1986][Gela 1989] is one of the widely used unsupervised techniques of feature transformation to detect the data structure and reduce the data dimensionality. This technique can discover uncorrelated features while retaining maximum variance about the original data. Recently, a non-linear version of PCA, called kernel PCA (KPCA) [Kram 1991], is used to capture the dominant nonlinear features of the original data. It transforms the data to a high dimensional feature space and obtains a set of transformed features rather than a subset of the original features. However, since it is based on the data variance, this technique can only be used to deal with numerical features. One may find some of the research work of applying KPCA to feature extraction in [Mika 1999][Scho 1999][Kocs 2004]. Some other classical feature transformation techniques include linear discrimination [Gama 1998], Fourier transformation, wavelet transformation [Mall 1998], function decomposition [Zupa 1998].

The feature selection methods based on the measure of information gain such as the decision trees [Quin 1986, 1987, 1993] are alternative ways to reduce the dimensionality of the feature spaces through feature extraction. These methods are built based on inductive learning and they require the computations of the information gain for each feature in any iteration. In a given iteration, the feature which obtains the maximum information gain should be selected. For a certain case base, generating the optimal decision tree has been proved to be a NP-hard problem. Since the traditional decision trees can only deal with the crisp feature values, fuzzy decision tree algorithms [Yuan 1995][Sing 2001][Mitr 2002][Pal 2001][Wang 2001a] are therefore developed to handle fuzzy feature values. However, the disadvantage of these methods is the requirement of much computation effort especially with high dimensionality (e. g., thousands of features). Another common method of feature extraction is to use an artificial neural network. Guo and Gelfand [Guo 1992] used an efficient small mutilayer net to reduce the nodes of a binary tree and extract nonlinear features. In [Hu 1998] [Seti 1998] [Utgo 1998], the hidden unit activations are interpreted as new extracted features from the original data. An innovative procedure is applied in [Fore 2002] to extract invariant surface features from each pixel of the range image, which are robust to noise, and invariant to scale, shift, rotations, curvature variations, and direction of the normal.

An often used approach in FR is rough sets [Pawl 1982, 1991], which allow the most informative features to be detected and then selected through the reduct computation. Different from PCA, this approach is supervised and selects a subset of the original features. Furthermore, rough sets are often used on symbolic features, and PCA is used primarily for numerical features. When handling numerical data, rough sets requires data discretization before generating reducts. There are two main groups of rough set-based FR methods: discernibility function-based and attribute dependency-based. Such methods, are computational intensive, i.e., in the former, during the generation of the discernibility matrix and in the latter, during the discovery of the positive regions.

To reduce the computational load inherent in these methods, Han et al. [Han 2004] proposed a relative attribute dependency approach, which can generate a reduct by counting the distinct rows in the sub-decision tables that are generated from the attribute sub-sets. In this approach, however, the information systems are always assumed to be consistent, i.e., the cases having the same feature values must be in the same class, which is not necessarily true in real world applications. To solve this problem, in this research, we have developed a fast rough set-based method to find the approximate reduct instead of the crisp reduct. In order to introduce the concept of approximate reduct, some primary concepts in rough set theory are also modified.

Feature weighting methods can also be used in feature reduction. There are a variety of such methods including gradient descend methods, neural networks, genetic algorithms to learn and optimize the feature weights. Those features with very small weights, say, smaller than a given threshold, are considered as those which can be ignored. In this research, we do not discuss this group of methods and only focus on the rough set-based methods, which do not need any additional domain knowledge and can be directly used to reduce the dimensionality.

## 2.1.2 Learning Similarity Measures of Nominal Features

There are two main categories of features: numerical features and symbolic features. Nominal feature is a type of symbolic feature whose feature values are completely unordered such as “red”, “green” and “blue” for the feature of color. Based on these completely unordered values, there are only two relationships for a given pair of cases: either they are the same or they are different. From the point of view of information system, the nominal features lead to coarse information granules, which may bring the difficulty of determining an accurate similarity measure in case matching and retrieval.

There are various forms of distance metrics and similarity measures for different types of features. The most often used metrics are Euclidean Distance, Hamming distance, and Cosine coefficients.

Euclidean distance is the most common type of distance metric which is based on the location of objects in Euclidean space. The distance is calculated as the square root of the sum of the squares of the arithmetical differences between the corresponding coordinates of two objects. In information theory, the Hamming distance is defined as the number of positions in two strings of equal length for which the corresponding elements are different. In the context of CBR, the Hamming distance of two cases is the number of features which have different feature values. In the field of information retrieval (IR), a variety of these metrics have been proposed for comparing the text documents, such as Dice’s coefficient, Jaccard’s coefficient, Cosine coefficient, and Overlap coefficient [Ande 1973] [Rijs 1979]. Due to the simplicity and normalization, Cosine coefficients, which computed based on the term frequency and inverse document frequency, are the most widely used similarity measures. There are also some other similarity measures [Liao 1998]. A useful one is proposed by Tversky [Tver 1977] which is computed as the ration of the number of common feature values and the total number of features. The similarity value between two documents reflects how relevant of one document to the other. Some probabilistic and statistical methods are also proposed to measure the relevance between a document and a query [Robe 1977][Crof 1979][Brow 1990][Fuhr 1992][Jone 2000][Laff 2003].

In the previously mentioned distance and similarity measures, Euclidean distance is usually applied to cases with numerical features while Hamming distance and that proposed by Tversky are suitable for cases with symbolic features, and the Cosine coefficients are more appropriate for the text-based cases.

In most of the existing work, the numerical features and symbolic features are often handled in separation in real world applications. To deal with these two types of features simultaneously, many researchers proposed various heterogonous similarity measures [Aha 1991, 1992][Wils 1997]. For example, Gowda and Diday [Gowd 1992] presented a new similarity measure for symbolic clustering which takes into account the “position”, “span” and “content” of both numerical and symbolic features. “Position” is the relative positions of two feature values on the real number line and therefore can only be used on numerical features. “Span” is used to indicate the relative sizes of the feature values without considering their common parts. “Content” is a measure of the common parts between two feature values. “Span” and “content” are suitable to deal with symbolic feature values. In these definitions, the similarity for nominal features is still determined as: when two feature values are equal, the similarity is defined as one; otherwise, the similarity is defined as zero.

These similarity measures for nominal features are coarse-grained which cannot provide enough information for case retrieval and then may affect the classification accuracy. In this research, we have showed that, for classification problems, the similarity should not always be set as zero when two nominal feature values are different. The domain of the similarity needs to be extended from {0, 1} to [0, 1] to obtain a fine-grained measure of nominal features.

### 2.1.3 Case Selection and Case Generation

Case selection (CS) and case generation (CG) are two groups of methods which can reduce the size of a given case base. CS is to select a subset of the cases in the original case base and CG is to extract prototypical cases to cover the whole case base. The CS methods can be divided into three main groups and most of the CG methods are prototypical-based, which are respectively presented as follows.

The first group of CS methods is k-NN based, e.g., the Condensed Nearest Neighbor Rule (CNN) proposed by Hart [Hart 1968], and the Wilson Editing method developed by Wilson and Dennis [Wils 1972]. There are several variations of the CNN and Wilson Editing methods, such as Reduced Nearest Neighbor (RNN) [Gate 1972], Selective Nearest Neighbor Rule [Ritt 1975], Repeated Wilson Editing, and all k-NN [Tome 1976]. These methods are very useful in identifying noisy cases because they closely examine the k-nearest neighbours of each case. They are all based on the assumption that similar problems should have similar solutions. Therefore, they regard noisy cases as those cases that are very similar in their problem specifications, yet propose different (or may be conflicting) solutions. These methods are very useful for identifying and removing noisy cases because they closely examine the *k*-nearest neighbours of each case. In this research, this group of CS methods are referred to as *k*-NN based methods.

The second group of CS methods is based on the concepts of case coverage and case reachability. Coverage of a case is the set of target problems (i.e., cases) that this case in question can be used to solve. The reachability of a target problem (i.e., a case), on the other hand, is the set of all cases that can be used to solve it. These two concepts are very useful in identifying redundant cases because they examine the problem solving ability of each case. Based on these two concepts, algorithms such as the footprint set based method [Smyt 1999a], case addition/ deletion policies [Smyt 1999b], redundant and inconsistent case detection [Raci 1997], and fuzzy-rough case base maintenance techniques [Cao 2003] are developed. Most of these methods are sensitive to the presence of noise and outliers. To address this problem, Pan et al. [Pan 2005] proposed a kernel-based greedy case selection algorithm. This algorithm has been showed to be more robust than the case deletion policy given by Smyth and Keane [Smyt 1995] and work better on nonlinear problems. This research constructs and compares different case selection approaches based on the similarity measure and the concepts of case coverage and reachability, which are closely related to the *k*-NN based methods.

The third group of CS methods are the density-based approaches. In these, each case is regarded as a point in a vector space. The density of each point is estimated and points with higher densities are selected. One such example is the data reduction algorithm proposed by Astrahan [Astr 1970], in which a hyper-sphere of radius *d* about a point is used to obtain an estimate of the density at that point. The densest point is selected repeatedly until all the points (i.e., cases) are covered by the selected point set. Pal and Mitra [Mitr 2002] propose another density-based multi-resolution data reduction algorithm that uses a similar idea. Their approach uses hyper-spheres with adaptive radii for both density estimation and case pruning.

Finally, the CG schemes are prototype-based approaches in which the original cases are modified or combined or merged to form new prototype cases. One representative algorithm is introduced by Chang [Chan 1974], where the nearest two cases with the same class label are merged into a single prototype using a weighted averaging scheme. The decision tree algorithms also belong to this group of case selection schemes, which try to extract rules as prototype cases from the case base by inductive learning. RISE [Domi 1995] and EACH [Salz 1991] are two systems which modify cases instead of simply deciding which case should be removed or selected. Using a similar idea, Pal and Mitra [Pal 2004] proposed a case generation approach based on rough-fuzzy techniques. In their approach, the cases are in the form of cluster granules rather than sample points. On the other hand, the support vectors produced by SVM can be considered as cases selected as a subset of the original case base. Recently, SVM ensemble which consists of several SVMs is proposed to expand the correctly classified area by each individual SVM. The widely used competitive algorithms of Self-Organizing Map (SOM) and Learning Vector Quantization (LVQ) [Koho 1988] can be also used for prototype-based CG. LVQ inherits almost all the features of SOM, except that it is a supervised learning algorithm. LVQ is able to summarise or reduce large datasets to a smaller number of representative vectors suitable for classification or visualisation. It has similar advantages of SOM, such as the robustness with noise and missing information.

## 2.1.4 Case Base Competence Model

Most of the related work to building case base competence model arises from the area of case base maintenance (CBM). According to Leake and Wilson [Leak 1998], CBM implements policies of revising the organization and contents of the case bases to facilitate the future reasoning for a particular set of performance objectives. Case base competence (coverage) is one of the most crucial factors contributing to the performance of a CBR system. As pointed out in [Leak 2000], "competence is a sine qua non for performance: no CBR system is useful unless it can solve the problems that it confronts". Case base competence has been brought sharply into focus since many maintenance policies are linked directly with the heuristics of measuring case based competence. The competence models can therefore be used to guide the construction of a case base.

McKenna and Smyth [Smyt 1998] presented and evaluated a case base competence model. In their model, the cases in a given case base are divided into different competence groups. The concept of competence group is defined in such a way that there is no overlap of the case coverage for different competence groups. The coverage of each competence group can be obtained by considering the group size, the case density, and the coverage of each case. Here the two important concepts of case coverage and case reachability are defined to compute the coverage of a case. The overall case base competence can then be computed simply by summing up the coverage of each group. A novel application of this model was demonstrated as a guide to case-based designers during the case visualization and authoring process. This competence model always assumes the cases are uniformly distributed and that there is no interaction among different competence groups. In this research, we build a fuzzy integral-based competence model for a given case base, which takes into account the non-uniform case distribution and the possible interaction among the competence groups by fuzzy measures.

Based on their competence model, Smyth and McKenna [Smyt 1999c] proposed a novel retrieval technique by introducing the concept of "footprint" set, which greatly improves the efficiency of retrieving competence and quality cases. It divides the case base into two hierarchies, the first hierarchy is the footprint set which is a subset of cases that can cover the whole original case base. The cases in a footprint set are considered as the most represent cases. The second hierarchy is the whole case base which is divided into some competence groups by the footprint set. The retrieval process has two stages: retrieving cases from the footprint set first, and then from the competence group (i.e., the set of cases pointed by a foot print case). The footprint set is considered as a compact version of the original given case base.

Smyth and Keane [Smyt 1995] proposed a category-based approach to depict the case competence and this approach was used to build case deletion policy to reduce the size of a case base. They describe the case competence by classifying cases into four categories according to their importance with respect to their neighbors. Those cases which are isolated and cannot be replaced by their neighboring cases for problem solving are called pivotal cases, which are considered to be the most important cases. If many similar cases are closed together, and they can provide similar solutions to the incoming query problem, then some of these cases can be deleted without affecting the overall competence of the CBR system. These deleted cases are called auxiliary cases, which are considered to be totally redundant cases. Based on the case categories, the maintenance policy is to detect and preserve the pivotal cases and remove the auxiliary cases.

This category-based competence-preserving approach is coarse-grained competence assessment. For example, this approach assumes the pivotal cases always make the same competence contribution which is not necessarily true. Furthermore, it does not consider the coverage overlapping between different cases, which can reduce or enhance the relative competence of a case. Smyth and Keane [Smyt 1999d] defined a more fine-grained measure called relative coverage that estimates the unique competence contribution of an individual case. The relative coverage measure takes into account the local coverage of the case as well as the degree to which this coverage is affected by the nearby cases. The measure of the relative coverage can be used to rank cases. Based on this ranking, the competence-rich cases can be selected as the representative cases in the case base.

## 2.2 Preliminary Knowledge of Rough Set Theory

Rough sets play an important role in the developed soft computing based techniques in this research. The fast rough set-based feature reduction is built on the basis of the generalization of the concept of reduct. The reduced feature set can then affect the quality of the selected or generated case knowledge bases through case selection and case generation. For the convenience of the readers, this section gives an introduction of some basic concepts in rough set theory.

Rough set theory, proposed by Z. Pawlak in 1982 [Pawl 1982], provides a tool for knowledge discovery from data. The main idea is based on the indiscernibility relation that describes indistinguishable objects. Concepts are represented by their lower and upper approximations. The observation that one cannot distinguish objects on the basis of given information about them is the starting point of the rough set philosophy. Imperfect information causes indiscernibility of objects. The indiscernibility relation (i.e., an equivalence relation) induces an approximation space made of equivalence classes of indiscernible objects. A rough set is a pair of a lower and an upper approximation of a set in terms of these equivalence classes.

The knowledge representation approach provided by rough sets is free of redundancies which is so typical for real life data bases. The algorithm and the models of objects can be used to support decisions concerning new objects. Using rough sets, problems such as decision table optimization, rule generation in expert systems, symbolic learning from examples, and dissimilarity analysis can be efficiently addressed. Many real life applications of rough set theory have been implemented with success. For example, fault diagnosis [Tay 2003], image processing [Pal 2002], data mining [Chan 1998], web and text categorizations [Jens 2004], and market decision-making [Shen 2004]. Very promising results have been obtained while using rough sets in voice recognition, approximation classification and other areas.

Before we provide the definition of rough set, we present first some of the basic concepts and mathematical definitions related to rough set theory. For details, one can refer to [Pawl 1991].

## 2.2.1 Information Systems and Decision Systems

In applying rough sets, the data used is usually represented in a flat table as follows: columns represent the attributes, rows represent the objects, and every cell contains the attribute value for the corresponding objects and attributes. In rough set terminology, such tables are called information systems.

More formally, an information system is a triple of $IS = (\boldsymbol{U}, A, f)$, where $\boldsymbol{U}$ is a non-empty finite set of objects $\{x_1, x_2, \ldots, x_n\}$(called universe); $A$ is a non-empty finite set of attributes (features) $\{a_1, a_2, \ldots, a_m\}$. For every attribute $a \in A$, there is an attribute value set $V_a$ associated with it, i.e., $f_a$: $\boldsymbol{U} \to V_a$. An example of an information system is shown in Table 2.1.

Table 2.1 An example of information system

| ***U*** | ***a*** | ***b*** | ***c*** | ***d*** |
|---|---|---|---|---|
| 1 | 3.8-3.99 | Average | No | Poor |
| 2 | 4.0 | Average | Yes | Poor |
| 3 | 3,8-3.99 | Good | No | Extensive |
| 4 | Below 3.8 | Good | No | Extensive |
| 5 | 4.0 | Poor | Yes | Average |
| 6 | Below 3.8 | Average | No | Average |

Table 2.1 describes an information system containing several student records. $\boldsymbol{U}$ = {1, 2, …, 6}, and $A$ = {$a$, $b$, $c$, $d$}, which represent the High School GPA, Extracurricular activities, Alumni relatives and Honor awards. For attribute $b$, the attribute value of the first student record is represented by $f_b$, $f_b$ = Average.

In many real life classification problems, the outcome (or class label) is already known. This class label is referred as a decision attribute. An information system that includes the decision attribute is called a decision system, which is denoted as $DT = (\boldsymbol{U}, A \cup \{d^*\}, f)$, where $d^* \notin A$ is the decision attribute. The elements of $A$ are called conditional attributes. More generally, a decision system is represented by $DT = (\boldsymbol{U}, C, D, f)$, where $C$ is the set of condition attributes and $D$ is the set of decision attributes. Table 2.2 shows an example of a decision system.

Table 2.2 An example of decision system

| ***U*** | ***a*** | ***b*** | ***c*** | ***d*** | ***e*** |
|---|---|---|---|---|---|
| 1 | 3.8-3.99 | Average | No | Poor | Reject |
| 2 | 4.0 | Average | Yes | Poor | Accept |
| 3 | 3,8-3.99 | Good | No | Extensive | Accept |
| 4 | Below 3.8 | Good | No | Extensive | Reject |
| 5 | 4.0 | Poor | Yes | Average | Accept |
| 6 | Below 3.8 | Average | No | Average | Reject |

This table is the same as Table 2.1 except that it has the decision attribute *e – Decision* about the admission application of the students.

## 2.2.2 Indiscernibility Relation

Given an information system, each subset of attributes can be used to partition the objects into clusters. The objects in the same cluster have the same attribute values. These objects are indiscernible (or similar) based on the available information. An indiscernibility relation can be defined to describe this property. Before the definition of indiscernibility relation is given, the equivalence relation and equivalence class are firstly defined.

**Definition 2.1** (Equivalence relation): A binary relation $R \subseteq X \times X$ is called an equivalence relation, if it satisfies

(1) reflexivity, i.e., an object is in relation with itself, denoted as $x\ R\ x$;

(2) symmetry, i.e., if $x\ R\ y$ then $y\ R\ x$; and

(3) transitivity, i.e., if $x\ R\ y$ and $y\ R\ z$ then $x\ R\ z$.

**Definition 2.2** (Equivalence class): The equivalence class of an element $x \in X$, denoted as $[x]_R$, consists of all objects $y \in X$ such that $x\ R\ y$. That is, $[x]_R$ = {y | $x\ R\ y$ and $y \in X$ }. Each equivalence relation $R$ will induce a partition on $U$, which is denoted by $\boldsymbol{U/R}$.

Now let us provide the definition of the indiscernibility relation.

**Definition 2.3** (Indiscernibility relation with respect to a subset of attributes B): Given an information system $IS = (\boldsymbol{U}, A, f)$, with any $B \subseteq A$ there is an associated equivalence relation $I_B$,

$$I_B = \{(x, x) \in U \times U \mid \forall\, a \in B, f_a(x) = f_a(x)\}. \quad (2.1)$$

$I_B$ is called the $B$-indiscernibility relation.

If $(x, x') \in I_B$, the objects $x$ and $x'$ are indiscernible from each other by attributes from $B$. The family of all equivalence classes of $I_B$ (i.e., the partition determined by $B$) will be denoted by $U/I_B$, or simply $U/B$; an equivalence class of $I_B$ containing $x$ will be denoted by $[x]_B$. Now we use an example information system to illustrate the indiscernibility relation and the corresponding equivalence classes.

Table 2.3 A decision system to illustrate concept of indiscernibility relation

| ***Stores*** | ***E*** | ***Q*** | ***L*** | ***P*** |
|---|---|---|---|---|
| 1 | High | Good | No | Profit |
| 2 | Med. | Good | No | Loss |
| 3 | Med. | Good | No | Profit |
| 4 | Low | Avg. | No | Loss |
| 5 | Med. | Avg. | Yes | Loss |
| 6 | High | Avg. | Yes | Profit |

In the decision system given by Table 2.3, there are six objects (6 stores) which described by three attributes, $E$, $Q$, and $L$, and one decision attribute $P$ which denotes the profit status of the store.

Considering the attribute set {$E$, $Q$}, by Equation 1, we can get the {$E$, $Q$}-indiscernibility relation, $I_{\{E,Q\}}$, and the partition determined by it is as below:

$U/I_{\{E,Q\}} = \{\{1\},\{2,3\},\{4\},\{5\},\{6\}\}$.

Similarly for the other attributes sets, we can also obtain their corresponding indiscernibility relations and equivalent classes. Below are partitions determined by the indiscernibility relations $I_{\{Q,L\}}$, $I_{\{E\}}$ and $I_{\{E,Q,L\}}$, respectively.

$U/I_{\{Q,L\}} = \{\{1,2,3\},\{4\},\{5,6\}\}$,

$U/I_{\{E\}} = \{\{1,6\},\{2,3,5\},\{4\}\}$,

$U / I_{\{E,Q,L\}} = \{\{1\},\{2,3\},\{4\},\{5\},\{6\}\}$.

The partition induced by decision attribute set $\{P\}$ is $U / I_{\{P\}} = \{\{1, 3, 6\}, \{2, 4, 5\}\}$.
By the available information, the concept of "profit" corresponds to the set {1, 3, 6}, while the concept of "Loss" can be characterized by the set of {2, 4, 5}. Here the observation is that neither of the two concepts can be defined by crisp manner, i.e., they can not be represented by the equivalence classes induced by the equivalence relation on any subset of the conditional attributes. Set approximations and rough set are introduced to deal with this type of concepts.

### 2.2.3 Set Approximations

Before describing the concept of rough sets, let us first give the definition of set approximation.

**Definition 2.4** (Set approximations): Consider an information system $IS = (\boldsymbol{U}, A, f)$. With each subset $X \subseteq \boldsymbol{U}$ and $B \subseteq A$, $R = I_B$, we associate two crisp sets

$$\underline{R}X = \{x \mid [x]_B \subseteq X\}, \tag{2.2}$$

$$\overline{R}X = \{x \mid [x]_B \cap X \neq \Phi\}, \tag{2.3}$$

called the *R*-lower and the *R*-upper approximations of *X*, respectively.

The lower and upper set approximations, $\underline{R}X = \underline{I_B}X$ and $\overline{R}X = \overline{I_B}X$, are also represented by $\underline{B}X$ and $\overline{B}X$, respectively. The set $\underline{B}X$ (or $\overline{B}X$) consists of objects, which surely (or possibly) belong to $X$ with respect to the knowledge provided by *B*. The set $BN_B(X) = \overline{B}X - \underline{B}X$ is called the B-boundary region of *X*, which consists of those objects that cannot surely belong to *X*.

**Definition 2.5** (Rough set): A set is said to be rough if the boundary region is non-empty with respect to *B*. That is, if $BN_B(X) \neq \Phi$, the set *X* is called rough (i.e., inexact) with respect to *B*; and if $BN_B(X) = \Phi$, the set *X* is crisp (i.e., exact) with respect to *B*, in contrast.

The set $\boldsymbol{U} - \overline{B}X$ is called the *B*-outside region of *X*, and it consists of such objects that certainly cannot belong to *X* (on the basis of knowledge in *B*).

An example of set approximation is shown below:

Let $X = \{x \mid P(x) = Profit\}$. From Table 2.3, we then obtain *X* = {1, 3, 6}. Let us assume *B* = {*E*, *Q*, *L*}, then $[1]_B = \{1\}$, $[2]_B = [3]_B = \{2,3\}$, $[4]_B = \{4\}$, $[5]_B = \{5\}$, $[6]_B = \{6\}$.

By the definitions of the lower and upper approximations, we have

$\underline{B}X = \{1, 6\}$, $\overline{B}X = \{1, 2, 3, 6\}$, $BN_B X = \{2, 3\}$.

It can be seen that the boundary region is not empty, and this shows that the set *X* is rough. Objects in the store set {1, 6} surely belong to those making profit, and the store set {1, 2, 3, 6} includes objects possibly making profit, where the objects 2 and 3 included in the boundary region cannot be classified either as making a profit or having a loss. $\boldsymbol{U} - \overline{B}X$ = {4, 5} shows that the stores 4 and 5 certainly do not belong to those that are making profit. Here the approximations of *X* are shown in Fig. 2.1.

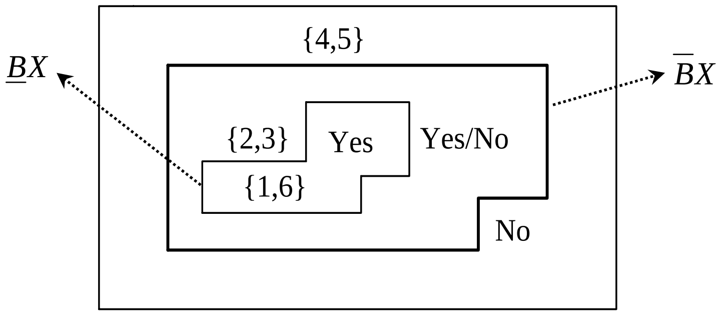

Fig. 2.1 Approximate the set of profit stores using three conditional attributes.

### 2.2.4 Reduct and Core

A fundamental problem occurred in information systems is whether the whole available information (including each attribute and record) is always necessary to support the decisions. This problem arises in many practical applications and will be referred as knowledge reduction. The concepts of reduct and core play basic role in rough set theory to address the issue of knowledge reduction.

**Definition 2.6** (Positive region): Let ***C*** and ***D*** denote the indiscernibility relations on *U* induced by the condition attribute set *C* and the decision attribute set *D*. The ***C***-positive region of ***D***, denoted as $\mathrm{POS}_C(D)$, is defined as

$$\mathrm{POS}_C(D) = \bigcup_{X \in U/D} \underline{C}X \tag{2.4}$$

To illustrate this concept, we use Table 2.3 as an example. Let $C = \{E, Q\}$ and $D = \{P\}$. According to definition 2.3,

$U/C = \{\{1\}, \{2, 3\}, \{4\}, \{5\}, \{6\}\}$, and

$U/D = \{\{1, 3, 6\}, \{2, 4, 5\}\}$.

For convenience, let $X_1 = \{1, 3, 6\}$, $X_2 = \{2, 4, 5\}$, then $U/D$ can be denoted by $\{X_1, X_2\}$.

Based on Definition 2.6, $\mathrm{POS}_C(D) = \bigcup_{X \in U/D} \underline{C}X = \underline{C}X_1 \cup \underline{C}X_2$.

Since $[1]_C = \{1\} \subseteq X_1$, $[2]_C = [3]_C = \{2, 3\} \not\subset X_1$, $[4]_C = \{4\} \not\subset X_1$, $[5]_C = \{5\} \not\subset X_1$, and $[6]_C = \{6\} \subseteq X_1$,

$\underline{C}X_1 = \{x \mid [x]_C \subseteq X_1\} = \{1, 6\}$.

Similarly, $\underline{C}X_2 = \{x \mid [x]_C \subseteq X_2\} = \{4, 5\}$.

Therefore, $\mathrm{POS}_C(D) = \{1, 6\} \cup \{4, 5\} = \{1, 4, 5, 6\}$.

**Definition 2.7** (Dispensable attributes and indispensable attributes): $c \in C$ is said to be dispensable in a decision system if

$\mathrm{POS}_C(D) = \mathrm{POS}_{C-\{c\}}(D)$,

Otherwise, $c$ is indispensable in $C$.

**Definition 2.8** (Reduct): The attribute set $R \subseteq C$ is called a reduct of $C$ if and only if each $r \in R$, $r$ is indispensable $\mathrm{POS}_R(D) = \mathrm{POS}_C(D)$.

Note that it is possible that there are more than one reduct of $C$. Let RED($C$) denote the set of all reducts of $C$.

**Definition 2.9** (Core): The set of all the condition attributes indispensable in a *DT* is denoted by CORE($C$),

$$\mathrm{CORE}(C) = \cap \mathrm{RED}(C). \tag{2.5}$$

The concept of core can be used as a basis for computation of all reducts, for it is included in every reduct and its computation is straightforward. On the other hand, the core can be interpreted as the set of the most characteristic part of knowledge, which cannot be eliminated when reducing the information.

To illustrate how to generate reduct and core, we also take the *DT* in Table 2.3 for example. Here $C = \{E, Q, L\}$, $D = \{P\}$.

$U/C = \{\{1\}, \{2, 3\}, \{4\}, \{5\}, \{6\}\}$,

$U/C$-$\{E\}$ = {{1, 2, 3}, {4}, {5, 6}},

U/C-{Q} = {{1}, {2, 3}, {4}, {5}, {6}},

$U/C$-$\{L\}$ = {{1}, {2, 3}, {4}, {5}, {6}},

$U/D$ = {{1, 3, 6}, {2, 4, 5}}.

Therefore, according to definition 2.6,

$$\mathrm{POS}_C(D) = \bigcup_{X \in U/D} \underline{C}X = \underline{C}\{1, 3, 6\} \cup \underline{C}\{2, 4, 5\}$$

$$= \{1, 4, 5, 6\}.$$

$\mathrm{POS}_{C\text{-}\{E\}}(D) = \{4\} \neq \mathrm{POS}_C(D)$, therefore, E is an indispensable attribute.

$\mathrm{POS}_{C\text{-}\{Q\}}(D) = \{1, 4, 5, 6\} = \mathrm{POS}_C(D)$, therefore, Q is dispensable.

Similarly, L is found to be dispensable.

The attribute E is then identified as the only indispensable attribute.

That is, CORE = {E}, and {E, Q} and {E, L} are the two reducts.

After generating the reducts, the original decision system can be reduced to two smaller *DT* as follows:

Table 2.4 Reduced decision tables

| ***Stores*** | ***E*** | ***Q*** | ***P*** | ***Stores*** | ***E*** | ***L*** | ***P*** |
|---|---|---|---|---|---|---|---|
| 1 | High | Good | Profit | 1 | High | No | Profit |
| 2 | Med. | Good | Loss | 2 | Med. | No | Loss |
| 3 | Med. | Good | Profit | 3 | Med. | No | Profit |
| 4 | Low | Avg. | Loss | 4 | Low | No | Loss |
| 5 | Med. | Avg. | Loss | 5 | Med. | Yes | Loss |
| 6 | High | Avg. | Profit | 6 | High | Yes | Profit |

### 2.2.5 Discernibility Matrix

The concept of discernibility matrix was proposed by A. Skowron in 1991, which enables simple computation of the core, reducts. In this section, the definition and its use to generate reducts and core are presented.

**Definition 2.10** (Discernibility Matrix): Let $IS = (U, A)$ be a information system with $n$ objects $\{x_1, x_2, \ldots, x_n\}$. The discernibility matrix of $IS$, denoted by $DM$ ($IS$) is a $n \times n$ matrix desined as
$(c_{ij}) = \{a \in A \mid a(x_i) \neq a(x_j)\}$ for $i, j = 1,2, \ldots, n$.

Thus entry $c_{ij}$ is the set of all attributes which discern objects $x_i$ and $x_j$.

The core can be defined now as the set of all single element entries of the discernibility matrix, i.e., $\text{CORE}(A) = \{ a \in A \mid c_{ij} = (a)$, for some $i, j\}$.

It can be easily seen that $B \subseteq A$ is the reduct of $A$, if $B$ is the minimal (with respect to inclusion) subset of $A$ such that
$B \cap c \neq \varnothing$ for any nonempty entry $c$ ($c \neq \varnothing$) in $DM$.

In other words, reduct is the minimal subset of attributes that discerns all objects discernible by the whole set of attributes.

**Example** Here we use the *IS* described by Table 2.3 to demonstrate how to compute the reducts and core through the discernibility matrix. Table 2.5 is the corresponding information system which is derived from the decision system in Table 2.3.

Table 2.5 An example *IS* to illustrate reduct computation based on discernibility matrix

| ***Stores*** | ***E*** | ***Q*** | ***L*** |
|---|---|---|---|
| 1 | High | Good | No |
| 2 | Med. | Good | No |
| 3 | Med. | Good | No |
| 4 | Low | Avg. | No |
| 5 | Med. | Avg. | Yes |
| 6 | High | Avg. | Yes |

According to the definition, *DM* of this table is as follows:

Table 2.6 Example of Discernibility Matrix

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | ▨ | | | | | |
| 2 | E | ▨ | | | | |
| 3 | E | ∅ | ▨ | | | |
| 4 | E, Q | E, Q | E, Q | ▨ | | |
| 5 | E, Q, L | Q, L | Q, L | E, L | ▨ | |
| 6 | Q, L | E, Q, L | E, Q, L | E, L | E | ▨ |

Note that the discernibility matrix is symmetric so we need only half of elements in the matrix.

From Table 2.6, CORE(A = {E, Q, L}) = {E}. {E, Q} and {E, L} is all possible reducts.

## 2.2.6 Dependency of Attributes

In data analysis, it is important to discover dependencies between the condition and decision attributes. Using these dependencies, one can identify and omit the unnecessary attributes (which are considered to be redundant information in the decision system) and the corresponding values for making decision or classification.

Intuitively, we say that a set of decision attributes *D* depends totally on a set of condition attributes *C*, denoted as $C \Rightarrow D$, if all the values of decision attributes are uniquely determined by values of the condition attributes. This implies, there exists a functional dependency between values of *C* and *D*. Note that the dependency can also be partial, which means only some values of *D* can be determined by values of *D*. Now we give its formal definition:

**Definition 2.11** Let *C* and *D* be subsets of *A*, $D \cap C \neq \Phi$ and $D \cup C = A$.
We say that *D* depends on *C* in a degree $k\ (0 \leq k \leq 1)$, denoted $C \Rightarrow_k D$, if

$$k = \gamma(C, D) = \sum_{X \in U/D} \frac{|\underline{C}X|}{|U|}, \tag{2.6}$$

where $|X|$ means the cardinality of *X*, ***U***/D denotes the partition determined by *D*, i.e., the family of all equivalence classes of $I_D$.

If *k* = 1 we say that *D* depends totally on *C*, and if *k* < 1, we say that *D* depends partially (with a degree *k*) on *C*. The degree *k*, called the degree of the dependency, means the ratio of all elements of the universe that can be properly classified into the partition ***U***/D employing attributes from *C*.

For example, in the decision system illustrated in Table 2.3, the attribute *P* depends on the set of attributes {*E*, *Q*, *L*} with a degree 2/3. This is explained below.

*C* = {*E*, *Q*, *L*}, *D* = {*P*}, ***U***/*D* = {{1, 3, 6}, {2, 4, 5}},
if $X = \{1, 3, 6\}$, $\underline{C}X = \{1, 6\}$;
if $X = \{2, 4, 5\}$, $\underline{C}X = \{4, 5\}$.

$$k = \gamma(C, D) = \sum_{X \in U/D} \frac{|\underline{C}X|}{|U|} = \frac{|\{1, 6\}|}{|\{1, 2, ..., 6\}|} + \frac{|\{4, 5\}|}{\{1, 2, ..., 6\}|} = 2/3 .$$

This means only four of the six objects, i.e., {1, 4, 5, 6} in ***U***, can be identified exactly, by using the attributes *E*, *Q* and *L*. Here, the decision attribute *Pf* partially depends on the condition attributes *E*, *Q*, and *L* with a degree 2/3.

## 2.3 Summary

In this chapter, we have briefly reviewed the related work of feature reduction, learning similarity measures, case selection and case generation, and case base competence model. The fundamental concepts and basic knowledge in rough set theory is then introduced for the convenience of readers.

# Chapter 3

# Fast Rough Set-Based Feature Reduction

Feature reduction (FR) is the first task of case knowledge extraction, its purpose is to remove the non-informative features and facilitate the task of case selection. This chapter presents a novel and fast approach for FR, which is developed based on the relative attribute dependency among features to compute the approximate reduct instead of crisp reduct. The approximate reduct is considered as a generalization of the crisp reduct, which can be found quickly. Some fundamental concepts, such as dispensable/ indispensable attributes, reduct and core, are also modified. The overall experimental results on four real life data sets show that the proposed FR method can preserve, and may also improve, the solution accuracy while at the same time reduce the dimensionality of the case bases. The developed FR method is compared with the widely used kernel PCA (KPCA), and their respective storage requirement, accuracy and training time are discussed.

## 3.1 Introduction

The purpose of FR is to identify the most significant attributes and eliminate the irrelevant ones to form a good feature subset for the case bases in CBR systems. It can be considered as a necessary pre-process for the task of CS, its performance is closely depended on the involved features and feature importance (weight). In the CS methods, the feature importance is crucial in computing the similarity, *k*-Nearest Neighbours, case coverage and case reachability. One solution is to obtain the feature weights by interviewing domain experts, which is labour intensive. Another solution is to obtain these weights using machine learning methods, such as decision tree generation [Shiu 2001a], or neural network training [Pal 2004]. However, these methods transform the feature weighting information into a set of rules or a trained neural network making them unsuitable for calculating similarity and adaptation on unseen cases. Other problems of using these machine learning methods include the difficulty of determining a feature evaluation function, and the requirement of much training effort due to the presence of non-informative features in the training process.

In this research, the feature importance in CS is addressed by rough set-based FR through reduct computation. The preserved features in the computed reducts are regarded as the most important ones, and the other features are regarded as irrelevant. The reduct computation does not require any domain knowledge, and the computation complexity is only linear with respect to the number of features and cases. If incorporating the FR in CS, the case representation should still be the same as that of the original case base. That is, each case is described by a set of features (subset of the original feature set) and a class label. This form of knowledge representation is easier to understand and more convenient for use in CBR reasoning.

Therefore, in this research, we introduce a new concept, the approximate reduct, which can quickly identify and remove the non-informative features. The proposed FR approach can be considered as a generalization of the original attribute dependency-based or discernibility function-based techniques. This generalization is achieved by introducing a consistency measurement among reducts. The approach reduces the computational complexity to $O(n \times m)$. Furthermore, the consistency measurement can be used to control the size of the feature set. Compared with the widely used KPCA, our method is shown to be much faster. Four different data sets are used in the experiments, which have the number of features ranging from 17 to 2018, the number of cases ranging from 40 to 8124.

The reminder of the chapter is organized as follows. Section 3.2 presents some definitions and theoretical results which are related to the rough set-based FR approach. Based on these concepts, Section 3.3 provides two fast FR algorithms. Section 3.4 demonstrates some experimental results with respect to the improvements of the storage requirement and problem-solving accuracy. Some comparisons are also made between the developed FR and KPCA. Section 3.5 provides the conclusions and discussions.

## 3.2 Relevant Concepts and Theoretical Results

In Chapter 2, we have introduced some basic concepts in rough set theory, including information systems and decision systems, indiscernibility relation, set approximations, dispensable and indispensable attribute, discernibility matrix, reduct and core. Based on these concepts, this section provide some further definitions and properties of rough sets which are the preliminary knowledge for developing the fast rough set-based FR method. In this chapter, the same notations are used as those in Chapter 2. An information system and the corresponding decision table are still denoted by $IS = (U, A, f)$ and $DT = (U, A \cup \{d\}, f)$, respectively. The conditional attribute set $C = A$, and the decision attribute set $D = \{d\}$.

The concept of reduct is defined on the basis of positive region (see Definition 2.6 in Chapter 2). Here we give an equivalent definition which is based on the definition of indiscernibility relation (see definition 2.3 in Chapter 2).

**Definition 3.1 (Reduct)**

A sub-attribute set $B \subseteq A$ is called a reduct of $A$ if it is a set of indispensable attributes in the information system $IS$ and $IND(B) = IND(A)$.

Traditionally, the reducts are obtained through the computation of discernibility matrix (see Definition 2.10 in Section 2.2.6). The discernibility matrix of an $IS$ completely depicts the identification capability of the system and all reducts of the system are therefore hidden in some discernibility function induced by the discernibility matrix [Skow 1992].

Based on these definitions, it requires a considerable effort for the discernibility function-based reduct generation methods to compute the discernibility matrix. To demonstrate the computation complexity of these methods, we describe a discernibility matrix-based algorithm which is often used to generate reducts for a given feature set.

First, the discernibility matrix *DM* of the decision table *DT* is generated in Step 1.
Since *DM* is symmetric and the elements $dm_{ii} = \varnothing$ for $i = 1, 2, \ldots, n$, it can be represented only by the elements in the lower or upper triangle of the matrix.

Second, one of the reduct of the feature set *A*, denoted by *REDU*, is generated in Step 2. It can be further divided into several sub-steps:

(1) The *CORE* of *DT* is obtained as such a set of $dm_{ij}$ that each $dm_{ij}$ contains a single attribute which can identify at least two cases, that is, $CORE = \{dm_{ij} \in DM \mid \text{card}(dm_{ij}) = 1\}$.
(2) *REDU* is initialized to *CORE*.
(3) One of the attributes in *A* is added iteratively to *REDU* until the intersection of *REDU* and each $dm_{ij}$ $(1 \le i, j \le N)$ is not empty.

The sub-step (1) is based on a proposition of the concept of core [Skow 1992]:
$CORE(A) = \{a \in A\colon dm_{ij} = \{a\} \text{ for some } i, j\}$.

Proof. Let $B = \{a \in A\colon dm_{ij} = \{a\} \text{ for some } i, j\}$. It needs to show that $CORE(A) = B$. The proof is divided into two parts:
($\subseteq$) Let $a \in CORE(A)$. Then $IND(A) \subseteq IND(A - \{a\})$, so there exist $x_i$ and $x_j$ which are indiscernible with respect to $A - \{a\}$ but discernible by a. Hence $dm_{ij} = \{a\}$.

($\supseteq$) If $a \in B$ then for some *i* and *j* we have $dm_{ij} = \{a\}$. Hence *a* is indispensable in *A*. □

The reduct computation in sub-step (3) can be explained as follows. Since any attribute in $dm_{ij}$ can distinguish cases *i* and *j*, the attributes in *REDU* can also discern the two cases if the intersection of *REDU* and $dm_{ij}$ is not empty. When the iterations of adding attributes to *REDU* stop, the discernibility power of the set of elements in *REDU* (i.e., a subset of the original attributes) is the same as that of the original set of conditional attributes *A*.

Here we should mention that, *REDU* is not necessarily the minimal set of attributes that preserves the identification capability of the original information system.

**Algorithm: Generate Reduct Based on Discernibility Matrix**

```
Step 1. Create the discernibility matrix $DM = [dm_{ij}]$, $i, j = 1, 2, \ldots, n$.

Step 2. Generate one reduct.
    Let $A$ denote the set of original attributes.

    CORE = $\{dm_{ij} \in DM \mid \text{card}(dm_{ij}) = 1\}$;
    REDU = CORE;
    $A = A - REDU$;

    While ($A \neq \emptyset$) do
        If ($REDU \cap dm_{ij} \in DM \neq \emptyset$ for every $i$ and $j$}, stop;
        Else
        {Randomly select one attribute $a \in A$
         Add $a$ to $REDU$: $REDU = REDU \cup \{a\}$;
         $A = A - \{a\}$;
        }
```

In this reduct generation algorithm, Step 1 requires O($n^2$) computations to create the discernibility matrix. This is because that *DM* has $n^2$ elements of the form $dm_{ij}$ and the number of steps for computing of any $dm_{ij}$ is bounded by a constant. In Step 2, *REDU* has *m* elements at maximum, and *DM* has $n^2$ elements. Therefore, Step 2 needs O ($n^2 \times m$) computations to obtain the intersection of REDU and each $dm_{ij}$. Thus, the complexity of this algorithm is O ($n^2 \times m$), which is rather high with large number of cases and attributes.

To address the problem of computational complexity, Han et al. [Han 2004] have developed a reduct computation approach based on the concept of relative attribute dependency. Given a subset of condition attributes, *B*, the relative attribute dependency is a ratio between the number of distinct rows in the decision table corresponding to *B* only and the number of distinct rows in the decision table corresponding to *B* together with the decision attributes, i.e., $B \cup \{d\}$. The larger the relative attribute dependency value (i.e., close to 1), the more useful is the subset of condition attributes *B* in discriminating the decision attribute values. If this value equals to 1, each distinct row in the decision table corresponding to *B* maps to a distinct decision attribute value.

Some further concepts [Han 2004] are defined as:

**Definition 3.2 (Projection)**

*Let $P \subseteq A \cup D$, where $D = \{d\}$. The projection of U on P, denoted by $\Pi_P(U)$, is a sub table of U and is constructed as follows:*

1) *remove attributes $A \cup D - P$; and*
2) *merge all indiscernible rows.*

**Definition 3.3 (Consistent Decision Table)**

*A decision table DT or U is consistent when $\forall x, y \in U$, if $f_D(x) \neq f_D(y)$, then $\exists a \in A$ such that $f_a(x) \neq f_a(y)$.*

Table 3.1 provides an example of consistent decision table. Here $U = \{c_1, c_2, \ldots, c_8\}$, $A = \{a, b, c, d\}$ and $D = \{e\}$. In Table 3.1, for every two objects in *U*, if they have the same attribute values for all the attributes, their decision attribute must be equal. In contrast, Table 3.2 shows an example of inconsistent table which derived from Table 3.1. It is obvious that $c_8$ and $c_9$ have the same condition attribute values, {1, 1, 1, 2}, but different decision attribute value, 1 for $c_8$ and 2 for $c_9$.

Table 3.1 A consistent decision table

| ***Id*** | ***a*** | ***b*** | ***c*** | ***d*** | ***e*** |
|---|---|---|---|---|---|
| $c_1$ | 1 | 1 | 2 | 1 | 2 |
| $c_2$ | 1 | 2 | 1 | 2 | 1 |
| $c_3$ | 2 | 2 | 2 | 1 | 2 |
| $c_4$ | 3 | 1 | 2 | 2 | 1 |
| $c_5$ | 3 | 2 | 2 | 1 | 1 |
| $c_6$ | 1 | 2 | 2 | 1 | 2 |
| $c_7$ | 3 | 2 | 1 | 2 | 1 |
| $c_8$ | 1 | 1 | 1 | 2 | 1 |

Table 3.2 An inconsistent decision table

| ***Id*** | ***a*** | ***b*** | ***c*** | ***d*** | ***e*** |
|---|---|---|---|---|---|
| $c_1$ | 1 | 1 | 2 | 1 | 2 |
| $c_2$ | 1 | 2 | 1 | 2 | 1 |
| $c_3$ | 2 | 2 | 2 | 1 | 2 |
| $c_4$ | 3 | 1 | 2 | 2 | 1 |
| $c_5$ | 3 | 2 | 2 | 1 | 1 |
| $c_6$ | 1 | 2 | 2 | 1 | 2 |
| $c_7$ | 3 | 2 | 1 | 2 | 1 |
| $c_8$ | 1 | 1 | 1 | 2 | 1 |
| $c_9$ | 1 | 1 | 1 | 2 | 2 |

Let $P = \{a, c, d\}$, according to Definition 3.1, the projection of $U$ on $P$, $\prod_P (U)$ can be described by Table 3.3 which is a sub-table of Table 3.1:

Table 3.3 An example of projection

| *Id* | *a* | *c* | *d* | *e* |
|---|---|---|---|---|
| $c_1$ | 1 | 2 | 1 | 2 |
| $c_2$ | 1 | 1 | 2 | 1 |
| $c_3$ | 2 | 2 | 1 | 2 |
| $c_4$ | 3 | 2 | 2 | 1 |
| $c_5$ | 3 | 2 | 1 | 1 |
| $c_6$ | 1 | 2 | 1 | 2 |
| $c_7$ | 3 | 1 | 2 | 1 |
| $c_8$ | 1 | 1 | 2 | 1 |

**Definition 3.4 (Relative Dependency Degree)**

*Let B $\subseteq$ A, A be the set of conditional attributes. D is the set of decision attributes. The relative dependency degree of B w.r.t. D is defined as* $\delta_B^D$, $\delta_B^D = \frac{|\Pi_B(U)|}{|\Pi_{B\cup D}(U)|}$, *where* $|\Pi_X(U)|$ *is the number of equivalence classes in U / IND(X).*

The relative dependency degree $\delta_B^D$ implies how well subset *B* discerns the objects in *U* relative to the original attribute set *A*. It can be computed by counting the number of equivalence classes induced by *B* and *B*∪*D*, i.e., the distinct rows in the projections of *U* on *B* and *B*∪*D*.

Take Table 3.3 for example to show the process of the computation of relative dependency degree $\delta_B^D$. Here *B* = {*a*, *c*, *d*}, *D* = {*e*}.

Since

$$U / IND(B) = \{\{c_1, c_6\}, \{c_2, c_8\}, \{c_3\}, \{c_4\}, \{c_5\}, \{c_7\}\}$$
$$= U / IND(B\cup D) = \{\{c_1, c_6\}, \{c_2, c_8\}, \{c_3\}, \{c_4\}, \{c_5\}, \{c_7\}\},$$

$\delta_B^D = \frac{|\Pi_B(U)|}{|\Pi_{B\cup D}(U)|} = \frac{6}{6} = 1$. We can say that the sub-attribute set *B* has preserved the discernibility ability of the original feature set *A*.

Based on the definition of the relative dependency degree, we define the dispensable and indispensable attributes as follows:

**Definition 3.5 (Dispensable and Indispensable Attributes)**

*An attribute* $a \in A$ *is said to be dispensable in A w.r.t. D, if* $\delta_{A-\{a\}}^D = \delta_A^D$ *; otherwise, a is indispensable in A w.r.t. D.*

According to Definitions 3.3-3.4, we can obtain Lemma 1.

**Lemma 1**

$\forall B \subseteq A$, $\Pi_{B\cup D}(U)$ is consistent if and only if $|\Pi_B(U)| = |\Pi_{B\cup D}(U)|$.

Proof. We need to show (1) if $\Pi_{B\cup D}(U)$ is consistent, then $|\Pi_B(U)| = |\Pi_{B\cup D}(U)|$; and (2) if $|\Pi_B(U)| = |\Pi_{B\cup D}(U)|$, then $\Pi_{B\cup D}(U)$ is consistent.

(1) We assume that $\Pi_{B\cup D}(U)$ is consistent. According to the definition of consistent decision table (definition 3.3), if *xIND*(*B*)*y*, ($x \in U$, $y \in U$), then *xIND*($B\cup D$)*y*. Therefore, the number of equivalence classes of $\Pi_{B\cup D}(U)$ and $\Pi_{B\cup D}(U)$ is equal, i. e., $|\Pi_B(U)| = |\Pi_{B\cup D}(U)|$.

(2) This part of proof is by contradiction. Suppose $|\Pi_B(U)| = |\Pi_{B\cup D}(U)|$ and $\Pi_{B\cup D}(U)$ is inconsistent. Then there exists at least two objects *x* and *y*, having the same condition attribute values but different decision attribute value. That is, *x* and *y* belong to one equivalence class with respect to *B*, and belong to two different equivalence classes with respect to $B\cup D$. We have $|\Pi_B(U)| < |\Pi_{B\cup D}(U)|$. This contradicts to $|\Pi_B(U)| = |\Pi_{B\cup D}(U)|$. Therefore, $\Pi_{B\cup D}(U)$ must be consistent when $|\Pi_B(U)| = |\Pi_{B\cup D}(U)|$.

It is easy to prove that, if *U* is consistent, then $\delta_A^D = \dfrac{|\Pi_A(U)|}{|\Pi_{A\cup D}(U)|} = 1$, i. e., $|\Pi_A(U)| = |\Pi_{A\cup D}(U)|$.

**Lemma 2**

If *U* is consistent, then $\forall B \subset A$, $POS_B(D) = POS_A(D)$ if and only if $|\Pi_B(U)| = |\Pi_{B\cup D}(U)|$.

Proof [Han 2004]. *U* is consistent indicates that $POS_A(D) = U$ and that $|\Pi_B(U)| = |\Pi_{B\cup D}(U)|$ means that $\Pi_{B\cup D}(U)$ is consistent according to Lemma 1. If can be easily inferred that $\Pi_{B\cup D}(U)$ is consistent if and only if $POS_B(D) = U$.

Based on definitions 3.3-3.5 and Lemmas 1-2, Theorem 1 can be induced.

**Theorem 1**

*If U is consistent, $B \subseteq A$ is a reduct of A w.r.t. D, if and only if $\delta_B^D = \delta_A^D = 1$ and for $\forall Q \subset B, \delta_Q^D \neq \delta_A^D$.*

Proof [Han 2004]. According to definition 3.4, $\delta_B^D = \delta_A^D = 1$ means that $|\Pi_B(U)| = |\Pi_{B\cup D}(U)|$, if and only if, by Lemma 2, $POS_B(D) = POS_A(D)$. Similarly, for $\forall Q \subset B, \delta_Q^D \neq \delta_A^D$ if and only if $POS_Q(D) \neq POS_A(D)$. By definition 2.6 in Chapter 2, the theorem holds.

In order to compute the reduct quickly, we use definitions 3.4-3.5 (relative dependency degree, dispensable and indispensable attributes) and theorem 1. Theorem 1 gives the necessary and sufficient conditions for reduct computation and implies that the reduct can be generated by only counting the distinct rows in some projections.

Here we use the example in Table 3.1 to illustrate Han's method. As mentioned previously, the consistent decision table consists of 8 cases, 5 attributes including 4 conditional attributes, $A = \{a, b, c, d\}$, and 1 decision attribute, $D = \{e\}$. We have the following computations:

Since $\delta_{A-\{a\}}^D = \dfrac{|\Pi_{\{b,c,d\}}(U)|}{|\Pi_{\{b,c,d,e\}}(U)|} = \dfrac{5}{6} < 1$, $\delta_{A-\{b\}}^D = \dfrac{6}{6} = 1$,

*b* is considered to be dispensable and therefore removed from *A*, i. e., $A = \{a, c, d\}$.

Next, because $\delta_{A-\{c\}}^{D} = \frac{|\mathit{\Pi}_{\{a,d\}}(U)|}{|\mathit{\Pi}_{\{a,d,e\}}(U)|} = \frac{5}{5} = 1$, *c* is then removed from *A*, *A* = {*a*, *d*}.

Since $\delta_{A-\{d\}}^{D} = \frac{3}{4} < 1$, *d* is preserved in *A*.

Finally, according to Theorem 1, the reduct is generated as {*a*, *d*}.

However, this method can not directly used on inconsistent decision table such as that shown in Table 3.2. For every subset of condition attributes $B \subseteq A$, we always have $\delta_{B}^{D} = \frac{|\mathit{\Pi}_{B}(U)|}{|\mathit{\Pi}_{B\cup D}(U)|}$<1. Therefore, the reduct cannot be found. In next section, we will introduce the concept of approximate reduct to overcome this problem.

## 3.3 Fast Rough Set-Based FR Algorithms

Based on the concepts and theoretical results in Section 3.2, this section presents the developed fast rough set-based FR method. Before the FR algorithms are given, the concept of approximate reduct is firstly defined and explained.

In Theorem 1, *U* is always assumed to be consistent, which is not necessarily true in real life applications. In this section, we relax this condition to find approximate reducts rather than the exact reducts. The use of a relative dependency degree in reduct computation is extended to inconsistent information systems. Some new concepts, such as the $\beta$-dispensable attribute, $\beta$-indispensable attribute, $\beta$-reduct (i.e., approximate reduct), and $\beta$-core are introduced to modify the traditional concepts in rough set theory. The parameter $\beta$ is used as the consistency measurement to evaluate the goodness of the subset of attributes currently under consideration. It can also determine the number of attributes which will be selected in the generated approximate reduct. These are explained as follows.

**Definition 3.6 ($\beta$-dispensable attribute and $\beta$-indispensable attribute)**

*If $a \in A$ is an attribute that satisfies $\delta^{D}_{A-\{a\}} \geq \beta \cdot \delta^{D}_{A}$, a is called a $\beta$-dispensable attribute in A. Otherwise, a is called a $\beta$-indispensable attribute.*

The parameter $\beta$, $\beta \in [0, 1]$, is called the consistency measurement.

For example, in Table 3.1, we have $\delta^{D}_{A-\{a\}} = \frac{5}{6} < 1$, $\delta^{D}_{A-\{d\}} = \frac{3}{4} < 1$. If $\beta$ is set as 0.75, then the attributes *a* and *d* are both considered as $\beta$-indispensable attribute. If $\beta$ is set as 0.8, then only the attribute *d* is $\beta$-indispensable attribute.

**Definition 3.7 ($\beta$-reduct/approximate reduct and $\beta$-core)**

*B is called a $\beta$-reduct or approximate reduct of conditional attribute set A if B is the minimal subset of A such that $\delta^{D}_{B} \geq \beta \cdot \delta^{D}_{A}$. The $\beta$-core of A is the set of $\beta$-indispensable attributes.*

The relationship between $\beta$-reduct and $\beta$-core is similar to the relationship between the traditional reduct and core, which is described in theorem 2.

**Theorem 2**

$\beta$-core can be computed as the intersection of all approximate reducts, i.e.,

$\beta$-core = $\cap_i \, reduct_i$, where $reduct_i$ is the *i*th approximate reduct.

Proof: The proof is divided into two parts.

(1) For every attribute $a \in \beta$-core, $a$ is a $\beta$-indispensable attribute, i.e., $\delta^{D}_{A-\{a\}} < \beta \cdot \delta^{D}_{A}$. According to definition 3.7, an approximate reduct implies that $a \in \cap_i \, reduct_i$. This can be proved using the method of contradiction as follows:

If $\exists i,\ a \notin reduct_i$, then $reduct_i \subseteq A - \{a\}$ and $\delta^D_{reduct_i} < \delta^D_{A-\{a\}} < \beta \cdot \delta^D_A$. This result contradicts the assumption that $reduct_i$ is an approximate reduct. Therefore, $a \in \cap_i\ reduct_i$, and then A $\subseteq$ B holds.

(2) Let an attribute $a \in \cap_i\ reduct_i$. If we assume $a \notin \beta$-core, that is, $a$ is a dispensable attribute, then $\exists\ i$, such that $a \notin reduct_i$. This is not possible since $a \in \cap_i\ reduct_i$. Therefore, $a \in \beta$-core.

This completes the proof. □

The consistency measurement $\beta$ represents how consistent the sub-decision table (with respect to the considered subset of attributes) is relative to the original decision table (with respect to the original attribute set). It also reflects the relationship of the approximate reduct and the exact reduct. The larger the value of $\beta$, the more similar is the approximate reduct to the exact reduct computed using the traditional discernibility function-based methods. If $\beta = 1$ (i.e., attains its maximum), the two reducts are equal (according to theorem 1). The reduct computation is implemented by counting the distinct rows in the sub-decision tables of some sub-attribute sets. $\beta$ controls the end condition of the algorithm and therefore controls the size of reduced feature set. Based on Definitions 3.6-3.7, the first rough set-based FR algorithm in our developed approach is given in Fig. 3.1.

```
Feature Reduction Algorithm 1

Input: U- the entire set of objects;
A – the entire condition attribute set;
D – the decision attribute set.
Output: R – the approximate reduct of A

Step 1 Initialize R = $\varnothing$ (empty set).

Step 2 Compute the approximate reduct.
      While (A is not empty)
         Compute $\delta_R^D$;
      If ( $\delta_R^D$ > $\beta$)
         Return R and stop;
    Otherwise
      R = R $\cup$ {q};
      A = A – R;

Step 3 Output R.
```

Fig. 3.1 Feature reduction algorithm 1

```
Feature Reduction Algorithm 2

The inputs and output are the same as that in algorithm 1.

Step 1 Initialize R = $\varnothing$;

Step 2 For each a $\in$ A
            Compute the significance of a;
            Add the most significant one, q, to R:
            R = R $\cup$ {q};
            A = A – {q};

Step 3 For current R
            Compute the relative dependency degree $\delta_R^D$;

Step 4 While (A is not empty)
          If $\delta_R^D > \beta$, return R and stop;
          Otherwise, go to step 2 and then step 3.
```

Fig. 3.2 Feature reduction algorithm 2

In some domains, the order for selecting attributes in the reduct must be considered carefully. For example, when dealing with text documents, there are hundreds or thousands of keywords which are all regarded as attributes. If the order is randomly selected or if one simply makes use of the order in which keywords appear in a text document, the most informative attributes may not be selected initially during reduct computation. Therefore, the end condition $\delta_R^D > \beta$ in algorithm 1 cannot be satisfied quickly. It should also be borne in mind that the final attribute set may consist of many non-informative features. This issue is addressed by computing the significance value of each attribute. These significance values are used to guide the attribute selection sequence. Details are given in algorithm 2 (see Fig. 3.2).

Notice that the significance of an attribute can be evaluated in many ways using different evaluation criteria such as information gain (IG), frequency of occurrence (often used in text documents), and dependency factors (in rough set-based methods). In this research, the text-based CBR classifiers are built using the frequency approach (see Section 3.4).

The computation complexities of the feature reduction algorithms 1 and 2 are O($n \times m$), where $m$ is the number of attributes in $A \cup D$, $n$ is the number of objects in $U$. In the FR algorithms, we consider $m$ subsets of attributes by adding one attribute in each iteration until $\delta_R^D > \beta$, and $n$ computations are required in each iteration to calculate the different rows.

## 3.4 Experimental Results

In this section, we test our proposed FR algorithms mainly on classification problems and provide their comparisons with KPCA. To demonstrate their effectiveness, we use two main evaluation criteria: storage requirement and classification accuracy. Storage requirement is used to describe the number of attributes which need to be stored. The classification accuracy is the percentage of the unseen cases which can be correctly classify. The experiments used four real life data sets:

(1) House-votes-84 database [Hett 1998].

This data set contains the voting records of members of the United States House of Representatives. It contains a total of 435 cases and 17 boolean valued features including 16 conditional attributes and 1 decision attribute.

(2) Text document sets (Texts 1-8).

It is composed of eight text document sets randomly sampled from Reuters21578 [Lewi 1999]. The number of documents ranges from 40 to 1578, and the number of distinct words ranges from 150 to 2018. Here, each distinct word is considered as a condition attribute. The topic of the documents is the class label.

(3) Mushroom Database [Hett 1998].

This data set consists of descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms. There are 8124 samples and 23 nominally valued features. The last feature is the decision attribute with four possible class labels.

(4) Multiple Features [Hett 1998].

This data set consists of numerical features of handwritten numbers from “0” to “9”, which extracted from a collection of Dutch utility maps. There are total 2000 samples, 649 attributes and 10 classes. Since KPCA can only handle numerical data, this data set is used to compare our developed FR and KPCA techniques.

## 3.4.1 Rough Set-Based Feature Reduction

In this section, we test and analyze the feature reduction capability of the rough set-based algorithms proposed in Section 3.3. The experimental results demonstrate not only a reduced storage requirement but also an improvement in the classification accuracy with fewer features in the generated reduct. Notice that, Storage = |Reduced feature set| / |Original feature set|, where |.| is the cardinality of set (.). The accuracy is the classification accuracy when using the reduced feature set.

(1) House-votes-84.

This data set is tested using four splits: randomly selecting 20%, 30%, 40%, and 50% of the original data, as the testing data; the corresponding left data are used as the training data. The four splits are denoted as Split 1-4. Table 3.4 shows the reduced storage requirement and the classification accuracy with different $\beta$ values.

Table 3.4 Storage and accuracy with different $\beta$ values on house-votes-84

| | Split 1 | | Split 2 | | Split 3 | | Split 4 | |
|---|---|---|---|---|---|---|---|---|
| $\beta$ | Storage (%) | Accuracy (%) | Storage (%) | Accuracy (%) | Storage (%) | Accuracy (%) | Storage (%) | Accuracy (%) |
| 1.00 | 100.00 | 93.10 | 100.00 | 92.31 | 100.00 | 93.68 | 100.00 | 94.01 |
| 0.90 | 43.75 | 96.55 | 43.75 | 94.62 | 43.75 | ***95.98*** | 43.75 | 95.85 |
| ***0.95*** | 56.25 | ***97.70*** | 50.00 | 94.62 | 50.00 | ***95.98*** | 56.25 | ***96.31*** |
| 0.96 | 63.50 | 94.25 | 66.25 | ***95.38*** | 62.50 | 94.83 | 62.50 | 94.47 |
| 0.97 | 68.75 | 94.25 | 62.50 | 94.62 | 62.50 | 94.83 | 68.75 | 92.63 |

Table 3.4 provides three observations: (i) the features could not be reduced with $\beta = 1$; (ii) the classification accuracy is improved after the rough set-based feature reduction with almost all of the used $\beta$ values; (iii) the accuracy attains most of its maximums for the four splits when $\beta = 0.95$.

Table 3.5 Storage and accuracy using $\beta = 0.95$

| Split | P0 (%) | P(FR) (%) | Storage |
|---|---|---|---|
| 1 | 93.10 | 97.70 | 56.25 |
| 2 | 92.31 | 94.62 | 50.00 |
| 3 | 93.68 | 95.98 | 50.00 |
| 4 | 94.01 | 96.31 | 56.25 |
| ***Avg.*** | ***93.28*** | ***96.15*** | ***53.13*** |

In Table 3.5, P0 represents the original accuracy with the whole data set, while P(FR) denotes the accuracy with the reduced feature set after applying the rough set-based feature reduction algorithm 1.

(2) Text data sets.

The fast rough set-based FR algorithm 2 was applied to the text data sets. The most distinct characteristic of the text domain is its high dimensionality. We randomly select 80% documents in each text data set as the training data and the remaining 20% is used as the testing data.

Initially, each word (term) occurred in the text data is considered as a feature, and therefore there are often hundreds or thousands of feature terms in a text dataset. Before the FR algorithm is performed, we pre-process the term feature set to filter the stop words or very low-frequent words. Stop words are extremely common words which appear in almost every document, such as “a”, “the”, and “with”. These words are often considered to contribute little useful information in classifying the text documents. On the other hand, low-frequency words - words which occur just once or twice - are filtered. The words which remain are considered to be the original feature terms.

To facilitate the rough-set based feature term reduction, each text document is represented using a term vector with respect to the acquired original feature terms. Assume there are $m$ terms in the set of original feature terms. A given document *DOC* can be described by an $m$-dimensional term vector $[t_1, t_2, \ldots, t_m]$, where $t_k$ is a Boolean variable which is given by

$$t_k = \begin{cases} 1 & if\ DOC\ contain\ term\ k \\ 0 & if\ DOC\ does\ not\ contain\ term\ k \end{cases} \qquad k = 1, 2, \ldots, m. \tag{3.1}$$

Each document is represented by an *m*-dimension vector. An example of a document vector is

$$D = [t_1, t_2, \ldots, t_m] \qquad t_k \in [0, 1], k = 1, 2, \ldots, m, \tag{3.2}$$

where $t_k$ is the normalized weight of feature term *k* in document *DOC*. $t_k$ is computed by two steps: weight computation and weight normalization.

Step 1 Weight computation. Compute the weight of each feature term in each document using term frequency-inverted document frequency (*tf-idf*).

$$w_k = -\log (N_k/N) f_k,\ k = 1, 2, \ldots, m,$$

where $w_k$ is the weight of term *k*;

$N_k$ is the number of documents containing term *k*;

*N* is the total number of documents;

$f_k$ is the frequency of term *k*.

Note that $w_j$ is the weight of the *k*th term in the whole set of text documents. Here, in order to reduce the computational load, the term weight for each term in each document is not computed.

Step 2 Weight Normalization. Let $w_{\max}$ denote the maximal weight. $w_k$ is normalized to be

$$w_k = w_k / w_{\max}.$$

That is, for each *k* in equation (2), $t_k = w_k$.

In the FR algorithm 2, the significance of each feature $t_k$ ($k = 1, 2, \ldots, m$) is evaluated by its term frequency-inverted document frequency, i.e., $w_k$, which is positively proportional to the frequency of occurrence of this feature and inverse proportional to the number of documents which contain this term.

The experimental results in Table 3.6 shows that the storage requirements for all the eight data sets fall significantly, and the accuracy using reduced feature set is preserved for Text2, Text3, and Text6 and even improves for Text1, and Texts4-5. For Texts7-8, the accuracy decreases a little due to the reduction of features. Since the accuracy attains its maximum when $\beta$=1, here $\beta$ is set to be 1.

Table 3.6 Reduced storage and improved accuracy when applying $\beta$= 1 to text data

| Text dataset | P0 (%) | P(FR) (%) | Storage (%) |
|---|---|---|---|
| Text1 | 62.50 | 75.00 | 12.50 |
| Text2 | 62.50 | 62.50 | 9.05 |
| Text3 | 33.33 | 33.33 | 28.48 |
| Text4 | 37.93 | 41.38 | 10.51 |
| Text5 | 56.25 | 75.00 | 9.59 |
| Text6 | 77.78 | 77.78 | 31.53 |
| Text7 | 51.19 | 50.00 | 7.81 |
| Text8 | 72.79 | 69.39 | 3.37 |
| ***Avg.*** | ***56.78*** | ***60.55*** | ***14.11*** |

(3) Mushroom data.

We randomly select 80% data as the training data and the left 20% as the testing data. It is shown that there are five features in the generated reduct. Therefore, the storage requirement with respect to the feature set is 22.73% of the original feature set. Here $\beta$=1. The classification accuracy is not affected after feature reduction. For this data set, P0 = P(FR) = 1.

*Summary:* After applying the fast rough set-based FR method to House-votes data and texts 1-8, the feature set is substantially reduced and the classification accuracy is preserved or even improved. Tables 3.4-3.6 show that the improvement in classification accuracy is 3.06% for House-votes-84 and 3.77% for the text data sets. The size of the feature set decreases from the original 100% to 53.13% for house-votes-84, 14.11% for text data sets, and 22.70% for mushroom data.

### 3.4.2 Comparisons between Rough Set-Based FR and KPCA

In this section, we make some comparisons to further demonstrate the effectiveness of our developed FR method.

The experimental setup is as follows.

(1) Data - Since KPCA can only handle numerical data, the data set of *Multiple Features* is used to conduct these experiments.
(2) Data splitting - We use the training/testing data sets in the original database of *Multiple Features*, which has 1000 training samples and 1000 testing samples.
(3) Data preprocessing - The numerical data needs to be discretized before the use of rough sets in the FR process.
(4) Performance Evaluation - Four main evaluation indices are used including training time, storage requirement and classification accuracy. Here the unit of time is second.

In order to compare the developed rough set-based FR and KPCA methods, fuzzy discretization [Pal 2004] is performed before applying rough sets for feature reduction. Each feature is described by fuzzy sets: "high", "medium" and "low", and three $\pi$-membership functions are used. Note that the discertization may cause some information loss, and therefore the classification accuracy may be affected. The impact of different fuzzification methods on the performance is not discussed in this research. For the KPCA method, we select polynomial kernels of degree 3 due to the expensive training with RBF kernels.

Table 3.7 shows the experimental results, where “T_train” is the training time; “T_trans” in KPCA is the required time for testing data transformation on the extracted components. The comparisons are made based on the same number of selected features. It is demonstrated that the KPCA achieve slightly higher classification accuracy, however, the training time and the transformation time are much more than the training time in rough set-based FR method.

Table 3.7 Rough set-based FR vs. KPCA feature extraction

| Rough set-based FR | | | | KPCA feature extraction | | | |
|---|---|---|---|---|---|---|---|
| $\beta$ values | Storage (%) | T_train | Accuracy (%) | Storage (%) | T_train | T_trans | Accuracy (%) |
| 0.80 | 1.15 | 5.00 | 81.30 | 1.15 | 17.23 | 1291.90 | 85.20 |
| 0.90 | 1.30 | 5.00 | 82.80 | 1.30 | 17.55 | 1170.10 | 89.40 |
| 0.95 | 1.44 | 5.00 | 84.20 | 1.44 | 17.30 | 1171.70 | 90.40 |
| 0.98 | 1.73 | 5.00 | 86.30 | 1.73 | 17.20 | 1166.10 | 89.10 |
| 0.99 | 1.87 | 7.00 | 91.30 | 1.87 | 16.89 | 1167.50 | 92.20 |
| Avg. | | 6.33 | 85.20 | Avg. | 17.23 | 1193.46 | 89.30 |

## 3.5 Discussions

Although our experimental results are very promising, we need to point out that there are still some limitations of the developed FR and CS approaches, which may need to be tackled in our future work: (1) The determination of the parameters, i.e., $\beta$, is empirical and heuristic based during the testing and their best values are dataset dependent. The characteristic of these parameters is: the smaller the values of them, the more the reduction of features. Therefore, we do not consider the parameter values smaller than 0.5 because it may cause much information loss. (2) The fast rough set-based FR method works better with symbolic data. The numerical data needs to be discretized before applying FR process.

## 3.6 Summary

In this chapter, we describe a fast rough set-based FR approach for case knowledge extraction. The concept of a reduct is generalized to an approximate reduct which can be obtained quickly. In some situations, the crisp reduct is the best subset of features in terms of the classification accuracy, e.g., when $\beta$ =1 for the Text data (Table 3.6). Although the generated approximate reduct is equivalent to the crisp reduct which can be obtained by the traditional discernibility function-based methods, the computational complexity has been reduced using our FR approach. In some other situations, the crisp reduct is not the optimal subset of features, e.g., $\beta < 1$ for the House-votes-84 (Tables 3.4-3.5) and the Multiple Features database (Table 3.7). In this chapter, the $\beta$ value is determined to optimize the classification accuracy, it can also be determined according to other performance criteria or user requirements such as the size of case bases or the case retrieval time.

Some related concepts are also modified. Through the computation of approximate reduct, the original feature set is reduced and a new case base with fewer features is generated. It is shown that, compared with using the original case base, higher classification accuracy and less storage space requirement could be obtained. Comparisons are also made between the proposed FR and KPCA. FR shows a much better performance in terms of the training time.

# Chapter 4
# Learning Similarity Measure of Nominal Features

This chapter tackles the second task for case knowledge extraction as mentioned in Chapter 1, i.e., learning similarity measures of nominal features. The purpose of this task is to discover the relationship among different feature values to emphasize the importance of critical features. Similar to the previously task of FR in Chapter 3, learning similarity measures can reduce the effect of non-informative features, and therefore enhance the quality of cases selected through the process of CS. The problem-solving accuracy which based on the k-nearest neighbour principle can also be improved.

Nominal feature is one type of symbolic features, its feature values are completely unordered. The most often used similarity metrics for symbolic features is the Hamming metric and its variances which always assume that, if two nominal feature's values are equal, the similarity is defined as one; otherwise, the similarity is defined as zero. This similarity computation is coarse-grained and may affect the quality of the retrieved cases and also the problem-solving accuracy. In this chapter, we extend the similarity values from {0, 1} to [0, 1] using a GA-based supervised method for the learning of similarities of nominal feature values.

This chapter is organized as follows. Section 4.1 introduces some often used similarity measures for different types of features. Section 4.2 presents an example to show the importance of learning the similarity measures for different nominal feature values. Section 4.3 describes the GA algorithm which is applied to determine the similarity measure for classification problems. Section 4.4 demonstrates some experimental results which include the learned similarity values, the improved accuracy, and the implied feature importance. Finally, Section 4.5 concludes this chapter.

## 4.1 Introduction

CBR systems [Kolo 1993][Pal 2004] have been successfully applied to various domains, among which those used in classification problems are called CBR classifiers. When a new problem is presented to a CBR classifier, the most similar case or cases will be identified and retrieved based on the given similarity measure. The solutions of these retrieved cases are then reused to solve the new problem. Since the basic assumption in CBR is that similar problems should have similar solutions, the similarity measure plays a critical role in case matching and retrieval. In this chapter, we develop a GA-based method to learning the similarity measures of nominal features for CBR classifiers.

There are two main categories of features: numerical features and symbolic features. Symbolic features can be further divided into nominal, ordinal, and combinational features. The feature values of a nominal feature are completely unordered such as “red”, “green” and “blue” for the feature of color. The ordinal features have discrete and ordered values such as “1”, “2” and “3” for the feature of job rank. The feature values of the combinational features consist of both unordered and ordered discrete values.

Obviously, the ordered values shall provide more information than the unordered ones. For example, based on the ordered feature values, we can identify whether a given pair of cases has the same feature values or not, in addition, we could also find out the ordered relation between these two feature values. However, based on the completely unordered feature values, there are only two possible relationships for a given pair of cases: either they are the same or they are different. From the point of view of information system, these unordered domain values of nominal features lead to coarse information granules, and may cause difficulty in determining an accurate similarity measure in the case matching and retrieval.

There are various forms of distance metrics and similarity measures for different types of features. The most often used metrics are Euclidean Distance, Hamming distance, and Cosine coefficients. They are briefly described below.

Euclidean distance is the most common type of distance metric which is based on the location of objects in Euclidean space. The distance is calculated as the square root of the sum of the squares of the arithmetical differences between the corresponding coordinates of two objects. Let $d_E(x, y)$ represent the Euclidean distance between two cases *x* and *y*, a similarity measure of *x* and *y* can be defined as: $Sim_E(x, y) = 1 / 1 + \varepsilon \cdot d_E(x, y)$, where $\varepsilon$ is a positive constant. The higher the value of $d_E(x, y)$, the lower the similarity between cases *x* and *y*.

In information theory, the Hamming distance is defined as the number of positions in two strings of equal length for which the corresponding elements are different. For example, if *x* = (111000) and *y* = (110100), then $d_H(x, y) = 2$. In the context of CBR, the Hamming distance of two cases *x* and *y*, $d_H(x, y)$, is the number of features which have different feature values. The smaller the value of $d_H(x, y)$, the more similar of *x* to *y*. Based on the Hamming distance, the corresponding similarity measure can be defined as $Sim_E(x, y) = 1 / 1 + \varepsilon \cdot d_E(x, y)$, which is similar to that based on Euclidean distance.

In the field of information retrieval (IR), the most important task is to identify and retrieve relevant documents for a given query document. The relevant degree of one document to a query is measured by the distance and similarity metrics. A variety of these metrics have been proposed for the text documents, some of them are Dice, Jaccard, and Cosine coefficients [Ande 1973]. Due to the simplicity and normalization, Cosine coefficients, which computed based on the term frequency and inverse document frequency, are the most widely used similarity measures. In the field of CBR, there are many text-based case bases, where these distance and similarity metrics can be used.

In this research, the domain of the similarity values will be extended from {0, 1} to [0, 1] to obtain a fine-grained measure of nominal features. The information hidden in the classification labels can be incorporated to determine the similarity values of different nominal feature values.

A GA-based supervised learning approach is developed to obtain the similarity measures of the nominal feature values for a given classification problem. Here we assume that there are only limited feature values in the domain of each nominal feature. Theoretically, for a given nominal feature, the similarity of each pair of feature values is required to be computed to determine the similarity measure of this feature. That is, if there are $n$ elements in the domain of a feature, $n \cdot (n-1)/2$ similarity values need to be learned, which may require substantial computational effort. In practice, it is not necessary to determine so many similarity values. In the given classification problem, if two different nominal feature values certainly lead to different class labels, the similarity between these two nominal values is assumed as zero. The GA-based method is then used to learn the similarity values of other nominal feature values. The learned similarity values are expected to improve the classification accuracy and can be used to analyze the importance of each feature in the given CBR classifier.

## 4.2 Significance of Learning Nominal Feature Similarity: An Illustrative Example

In this section, an example is presented to explain the significance of learning similarity measures for the nominal features. First, we explain the problems of using {0, 1} as the domain of the similarity measures, then we demonstrate the advantages of learning similarity measures. The benefits from the learned similarity measures, including the improvement of both classification accuracy and the understanding of the data sets, are discussed.

## 4.2.1 Problem Statement

Table 4.1 describes a case base with nominal features. The classification problem is to predict which continent a person comes from according to his hair color and complexion. The features “Hair_Color” (***H***) and “Complexion” (***C***) are conditional features; and the last feature, “Place”, is the class label. $D_H = \{Bl, Br, G, R\}$ and $D_C = \{Ye, Bl, W\}$ are used to represent he domains of the two conditional features of ***H*** and ***C***, respectively.

Table 4.1 A case base with nominal features

| ID | Hair_Color | Complexion | Place |
|---|---|---|---|
| 1 | *Bl* | *Ye* | *Asia* |
| 2 | *Br* | *Bl* | *Asia* |
| 3 | *G* | *Ye* | *Asia* |
| 4 | *G* | *W* | *Europe* |
| 5 | *Br* | *W* | *Europe* |
| 6 | *R* | *W* | *Europe* |

Table 4.2 Testing cases

| ID | Hair_Color | Complexion | Place |
|---|---|---|---|
| 1 | *R* | *Bl* | *Asia* |
| 2 | *R* | *Ye* | *Asia* |
| 3 | *Bl* | *W* | *Europe* |

Traditionally, the similarity between two different nominal feature values is defined as zero. For the case base showed in Table 4.1, we have

$sim(Ye, Bl) = sim(Ye, W) = sim(Bl, W) = 0$ with respect to the feature ***C***, and
$sim(Bl, Br) = sim(Bl, G) = sim(Bl, R) = sim(Br, G) = sim(Br, R) = sim(G, R) = 0$ with respect to the feature ***H***,

where $sim(\bullet_1, \bullet_2)$ is the similarity value of $\bullet_1$ and $\bullet_2$.

However, consider feature ***C***, it can be implied from the class labels of the cases in Table 4.1 that *Ye* and *Bl* should be more similar than *Ye* and *W* does. This is based on the observation that either a person has the complexion of *Ye* or *Bl*, he comes from Asia; in contrast, if the person has *W* complexion, he surely comes from Europe. If we still assume that the similarity of *Ye* and *Bl* is the same as that of *Ye* and *W* (i.e., equal to zero), problems will arise when classifying new problems (unseen cases).

Assume such a problem *p* is presented to the CBR classifier as: *p* = {***H*** = *R*, ***C*** = *Bl*}. Based on the similarity metric in [Gowd 1992], cases 2 and 6 are retrieved with the maximum similarities, *sim*(*e*, 2) = *sim*(*e*, 6) = 0.5. Therefore, *e* cannot be specifically classified to “Asia” or “Europe”.

## 4.2.2 Learning Similarity Measures for Nominal Features

To address the problem mentioned in Section 4.2.1, we can adapt the similarity measure of *Ye*, *Bl*, and *W* for the feature ***C*** by making use of the hidden information in the class labels. For example, if we redefine that *sim*(*Ye*, *Bl*) = 1 > 0, then the retrieved cases will be 1, 2, 3, and 6. Based on the majority voting rule, the class label of the unseen case *e* is determined as “Asia”. This modified similarity measure has been extended from {0, 1} to [0, 1], which is shown to be much better compared with the traditional similarity measures.

The determined similarity measure can also be used to analyze the goodness of each nominal feature in the given classification task. In this research, the contribution of a feature in a CBR classifier is evaluated by the inconsistent degree caused when using the feature for case retrieval. The smaller the degree, the more useful is the feature. The inconsistency arises when cases have the same feature values but belong to different classes. In this sense, the best two situations (when the inconsistency degree = 0) of a nominal feature *f* are (i) all cases with different feature values are certainly classified to different classes; (ii) all cases with different feature values belong to the same classes. In the first situation, we assume that all the similarities of different feature values of *f* are equal to zero, while in the second situation, they are equal to one. Therefore, a feature $f_1$ is said to be more useful than another feature $f_2$ when its learned similarity values are closer to zero or one than those of $f_2$.

We define a distance *d* between the similarity values of a feature *f* and the set {0, 1} to describe how close the similarity values to zero or one. The distance *d* is denoted as

$$d(\{s_1, s_2, \ldots, s_L\}, \{0,1\}) = \sum_{i=1}^{L} \min\{s_i, 1-s_i\}, \tag{4.1}$$

where $\{s_1, s_2, \ldots, s_L\}$ is the learned set of similarity values of *f*, *d* is also denoted as $d(f, \{0, 1\})$ in the following sections. Using this definition of distance, the feature importance can be analyzed based on the learned similarity measure of the nominal features.

Since the learning process of determining the similarity measures is supervised and it aims to reduce the inconsistency contained in the case base, the classification accuracy can be improved using the learned similarity values. On the other hand, the importance of each feature is reflected in its learned similarity measures and some specific domain knowledge can also be obtained through the similarity value of each pair of nominal feature values.

For each nominal feature, the number of similarity values to be determined depends on the number of different feature values in the domain of this nominal feature. Theoretically, there are $4\times(4-1)/2 + 3\times(3-1)/2 = 9$ similarities to be learned for the case base in Table 4.1. In fact, we do not need to determine all of these similarity values. As mentioned in Section 4.1, if two different feature values certainly lead to different class labels, the similarity between the two nominal values is set as zero. For example, consider the feature ***H***. *sim*(*Bl*, *R*) is zero because case 1 (with *Bl*) and case 6 (with *R*) belong to different classes; for the feature ***C***, only the similarity of *Ye* and *Bl* should be modified from the original similarity value zero. The similarities of *Ye* and *W*, *Bl* and *W* are still zero because the cases with *Ye* or *Bl* surely belong to different classes from that of the cases with *W*. Therefore, there are totally 6 but not 9 similarity values that need to be determined.

## 4.3 Using GA to Learn Similarity Measure for Nominal Features

The genetic algorithms (GA), proposed by John Holland in the early 1970s, are adaptive and robust search algorithm inspired by natural evolution. They are viewed as randomized, yet structured optimization techniques. GAs are performed iteratively on a set of coded solutions, called a population, with three operators: selection, crossover, and mutation. To produce the optimal solution of a problem, a GA starts from a set of assumed solutions (i. e., chromosomes) and evolves better sets of solutions over a sequence of iterations. In each iteration, the objective function (i.e., fitness) determines the suitability of each solution, based on which some of the solutions (i.e., parent chromosomes) are selected for reproduction.

In this section, we describe a GA algorithm for learning the similarity values of nominal features falling in [0, 1] instead of {0, 1}. This similarity learning method is supervised and the testing classification accuracy is directly used as the fitness function.

Let there be a case base consisting of $N$ cases $e_1$, $e_2$, …, $e_N$, $m$ nominal features $f_1, f_2, \cdots, f_m$ . The domain of each feature has limited elements represented by $D_i = \{v_{i1}, v_{i2}, ..., v_{m,l_i}\}$ , $i$ = 1, 2, …, $m$; $v_{ij}$ is a nominal value, $l_i$ is the number of different values in the domain of the $i$-th nominal feature. As we have mentioned in Section 4.1, $L_i$ = $l_i \cdot (l_i - 1)/2$ similarity values should be learned at most for the $i$-th feature. In the following, we discuss the encoding rule, fitness function, and the constructed GA algorithm in detail.

Encoding rule:

Each chromosome is encoded as a string consisting of *m* parts corresponding to the *m* features. A chromosome *c* takes the form shown in Fig. 4.1. For the *i*-th part, there are $L_i$ genes represented by $L_i$ decimals: $s_{ip} \in [0, 1], (1 \le p \le L_i)$, representing the similarity measure for the *i*-th feature. The initial values of $s_{ij}$ $(j = 1, 2, ..., L_i)$ is randomly generated for $i = 1, 2, ..., m$.

| $s_{11}$ | $s_{12}$ | $\cdots$ | $s_{1L_1}$ | ▨ | $s_{21}$ | $s_{22}$ | $\cdots$ | $s_{2L_2}$ | ▨ | $\cdots$ | ▨ | $s_{m1}$ | $s_{m2}$ | $\cdots$ | $s_{mL_m}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.25 | 0.10 | … | 0.83 | ▨ | 0.62 | 0.58 | … | 0.35 | ▨ | … | ▨ | 0.74 | 0.40 | … | 0.56 |

Fig. 4.1 A chromosome *c*

Fitness Function:

In the GA-based learning process, the fitness function of a chromosome *c* is the corresponding classification accuracy using the similarity values indicated in *c*. Based on case retrieval, the class label of an unseen case can be determined by the majority of its *k* nearest neighbours. The classification accuracy is the ratio of the number of correctly classified cases, $N_{Corr}^{c}$, over the whole number of unseen cases, $N_{Total}^{c}$. The fitness function is then defined as: $fitness(c) = N_{Corr}^{c} / N_{Total}^{c}$.

The GA Algorithm:

a) Initialize the population of the chromosomes. A population set is represented by $\{ c_1, c_2, ..., c_P \}$, where *P* is the size of the population. Each chromosome is encoded as in Fig. 4.1. Each gene is a randomly initialized to be a decimal in [0, 1], representing the similarity value between two nominal values in the domain of each nominal feature.

b) Selection and crossover. Here the selection probability is set as 1 and the whole set of population is considered to be the mating pool. These settings let the modal to be closer to a random search. In each generation, two chromosomes are randomly selected to perform crossover. The cutting point for crossover is randomly generated and the genes in the two chromosomes that lie behind the cutting point are exchanged to produce an offspring.

c) Mutation. Let the mutation probability be $p_{muta}$. Randomly select one gene $g$ (with value $v_g$) in the newly generated offspring string, and convert the value $v_g$ to $(1-v_g)$. If $v_g$ represents the degree of how similar of two feature values, then $(1- v_g)$ represents their degree of dissimilarity.

d) End condition. Repeat (a)-(c) until the number of generations attains a predefined threshold.

Here we provide some discussions about parameter control in GA. Crossover probability is how often crossover will be performed. If there is no crossover, offspring are exact copies of parents. If crossover probability is 100%, then all offspring are made by crossover. Crossover is made in hope that new chromosomes will contain good parts of old chromosomes and therefore the new chromosomes will be better. The crossover operator can be divided into two types: 1-point and multi-point crossover. 1-point is used in the traditional GA, where two mating chromosomes are each cut once at corresponding points, and the segments following the cuts are exchanged. In a 2-point or multi-point crossover, chromosomes are seen as loops formed by joining the ends together, rather than as linear strings. Researchers now agree that 2-point crossover produces better results than 1-point crossover [Beas 1993]. However, if a strict interpretation of the schema theorem is imposed then operators which use many crossover points should be avoided because they can cause extreme disruption to schema [Darr 1993].

Mutation probability is how often parts of chromosome will be mutated. If there is no mutation, offspring are generated immediately after crossover (or directly copied) without any change. If mutation probability is 100%, whole chromosome is changed. Mutation generally prevents the GA from falling into local extremes. Mutation should not occur very often, because then GA will in fact change to random search. Recommendations are often results of empiric studies of GAs that were often performed on binary encoding only. Crossover rate should be high generally, about 80%-95%. On the other side, mutation rate should be very low. Best rates seem to be about 0.5%-1%.

## 4.4 Simulation Results and Analysis

Two examples are used in the simulations to show the effectiveness of learning similarity measure using the GA-based learning approach. The used data sets include the example case base in Table 4.1 and the Balloons database from the UCI repository [Hett 1998]. The results demonstrate that the similarity measure can enhance both the classification accuracy and the understanding of the databases.

### 4.4.1 Example 1

The cases in Table 4.1 are used as training data, and those in Table 4.2 are used as testing cases. There are totally 6 similarity values which need to be determined: $sim(Bl, Br)$, $sim(Bl, G)$, $sim(Br, G)$, $sim(G, R)$ and $sim(Br, R)$ for the feature $\boldsymbol{H}$; $sim(Ye, Bl)$ for the feature $\boldsymbol{C}$. Therefore, each chromosome has two parts, the first of which consists of 5 genes and the second part has only one gene. Fig. 4.2 shows an example of randomly initialized chromosome:

| $sim(Bl, Br)$ | $sim(Bl, G)$ | $sim(Br, G)$ | $sim(G, R)$ | $sim(Br, R)$ |  | $sim(Ye, Bl)$ |
|---|---|---|---|---|---|---|
| 0.88 | 0.68 | 0.36 | 0.43 | 0.39 |  | 0.05 |

Fig. 4.2 An initialized chromosome

Here the size of population $P$ = 10, the terminal number of generations is from 100 to 20000, and the mutation probability $P_m$ = 0.05.

Table 4.3 shows the results of the learned similarity measure for the nominal features. For the feature ***H***, *Bl* and *Br* is the most similar feature values, and then comes the pair of *G* and *R*; and for feature ***C***, *sim*(*Ye*, *Bl*) = 0.51 is much greater than *sim*(*Ye*, *W*) = *sim*(*Bl*, *W*) = 0. With these similarity values, all the testing cases in Table 4.2 can be correctly classified by the training cases, i.e., the classification accuracy is 1. It is obvious that the learned similarity measure is superior to the traditional similarity measure which cannot specifically classify the testing cases 1 and 2.

Table 4.3 Learned similarity measure of nominal features (testing accuracy = 1)

| *Number of Generations* | *sim*(*Bl*, *Br*) | *sim*(*Bl*, *G*) | *sim*(*Br*, *G*) | *sim*(*G*, *R*) | *sim*(*Br*, *R*) | *sim*(*Ye*, *Bl*) |
|---|---|---|---|---|---|---|
| 100 | 0.38 | 0.07 | 0.29 | 0.74 | 0.19 | 0.45 |
| 500 | 0.69 | 0.72 | 0.53 | 0.45 | 0.50 | 0.55 |
| 1,000 | 0.66 | 0.55 | 0.50 | 0.44 | 0.46 | 0.54 |
| 5,000 | 0.72 | 0.49 | 0.49 | 0.52 | 0.47 | 0.51 |
| 10,000 | 0.76 | 0.50 | 0.50 | 0.50 | 0.51 | 0.50 |
| 20,000 | 0.84 | 0.49 | 0.49 | 0.51 | 0.50 | 0.50 |
| Avg. | 0.68 | 0.47 | 0.47 | 0.53 | 0.44 | 0.51 |
| *d*(*, {0, 1}) | 0.36 | | | 0.16 | | |

Here *d*(*, {0, 1}) in Table 4.3 denotes the distance of the learned similarities to the set of {0, 1} defined by Equation 1, where * means the feature ***H*** or ***C***. The distances are computed as:

$d(\boldsymbol{H}, \{0, 1\}) = ( (1\text{-}0.68) + 0.47 + 0.47 + (1\text{-}0.53) + 0.44))/(4\times3/2) = 0.36$; and

$d(\boldsymbol{C}, \{0, 1\}) = (1\text{-}0.51)/(3\times2/2) = 0.16$.

Therefore, the feature ***C*** is more important than the feature ***H*** for classifying unseen cases.

## 4.4.2 Example 2

The Balloons Database is used as the second example to demonstrate the effectiveness of the proposed method for learning similarity measures. This data set consists of 16 cases and 4 nominal features (three conditional features and one class label). There are two nominal values in the domain of each conditional feature. Some example cases are shown in Table 4.4. It is found that there are 4 similarity values which need to be learned: *sim*(*Yellow*, *Purple*) (Color), *sim*(*Small*, *Large*) (Size), *sim*(*Stretch*, *Dip*) (Act), and *sim*(*Adult*, *Child*) (Age).

Six cases are firstly selected as the training data and the remaining 10 cases are used as testing cases. The original classification accuracy based on the majority voting principle is 0.70. In the learning process of the GA algorithm, the mutation probability is also set as 0.05 as in Section 4.4.1.

Table 4.4 Example cases in Balloons database

| ID | Color | Size | Act | Age | Inflated |
|---|---|---|---|---|---|
| 1 | *Yellow* | *Small* | *Stretch* | *Adult* | *T* |
| 2 | *Purple* | *Large* | *Dip* | *Child* | *F* |

Table 4.5 Learned similarity values on Balloons Database (Original accuracy = 0.70)

| *Number of Generations* | *sim*(*Yellow, Purple*) | *sim*(*Small, Large*) | *sim*(*Stretch, Dip*) | *sim*(*Adult, Child*) | *Accuracy* |
|---|---|---|---|---|---|
| 100 | 0.55 | 0.43 | 0.69 | 0.28 | 1.0 |
| 500 | 0.56 | 0.86 | 0.83 | 0.38 | 0.9 |
| 1,000 | 0.57 | 0.55 | 0.79 | 0.24 | 0.9 |
| 5,000 | 0.63 | 0.51 | 0.29 | 0.03 | 1.0 |
| 10,000 | 0.65 | 0.31 | 0.84 | 0.25 | 1.0 |
| 20,000 | 0.70 | 0.33 | 0.79 | 0.23 | 1.0 |
| *Avg.* | 0.61 | 0.50 | 0.71 | 0.24 | 0.97 |
| *d*(*, {0, 1}) | 0.39 | 0.36 | 0.23 | 0.23 | - |

Note: * denotes the features, Color, Size, Act, and Age, respectively.

Table 4.5 shows the learned similarity values for the four nominal features. With these similarity values, the accuracy increases from the original 0.70 to 0.97. The distances of similarity values to {0, 1} for "Act" and "Age" are the smallest compared with that of other features. Therefore, the features "Act" and "Age" are the most critical features to make the classification decisions. In fact, only using these two features, all the testing cases can be correctly classified based on the majority voting principle. In contrast, with the other two features “Color” and “Size”, five out of ten cases are classified to the wrong classes.

## 4.5 Summary

In this chapter, we presented a GA-based approach to learn the similarity measures of nominal features. Two examples are used to illustrate the effectiveness of the developed learning method. The simulation results show that the testing accuracy increases and the importance of each feature can be analyzed through the learned similarity values. To summarize, the main contributions are as follows: (i) the similarity between the nominal features has been extended from {0, 1} to [0, 1], which can make the best out of the available information; (ii) since the use of learned similarity values can reduce the effect of non-informative features, this GA-based method is an alternative way to improve the classification accuracy; (iii) based on the learned similarity values, we can further analyze the importance of each nominal feature which can provide potential useful information to enhance the understanding of the data sets. The limitation of the GA-based learning approach is that it requires high computational complexity with large number of features.

# Chapter 5
# Case Selection Methods for Case Knowledge Extraction

In the previous chapters, we have addressed feature reduction and learning similarity measures for nominal features, which will facilitate the task of case selection (CS) for case knowledge extraction. In this chapter, we construct and compare different case selection methods based on the similarity measure and the concepts of case coverage and case reachability. The process of FR is incorporated in the CS process, which can reduce the training burden as well as enhance the performance of CS. The overall experimental results demonstrating on four real life data sets show that the combined FR and CS methods can preserve, and may also improve, the solution accuracy while at the same time substantially reducing the storage space. The case retrieval time is also greatly reduced because the CBR system contains a smaller amount of cases with fewer features. The developed FR and CS combination method is also compared with the kernel PCA and SVMs techniques. Their storage requirement, classification accuracy, and classification speed are presented and analyzed.

This chapter is organized as follows. Section 5.1 introduces some background and related work of CS. Section 5.2 presents four CS algorithms and their rationales. Section 5.3 explains the importance of FR in CS and the steps for combining them. Section 5.4 presents and analyses experimental results on both the individual and synergistic performance of the FR and CS methods. Some comparisons are also made between the developed FR and CS methods and the combination of KPCA and SVMs techniques. Section 5.5 provides the conclusions and discussions.

## 5.1 Introduction

In this chapter, the task of CS is discussed in the context of CBR classifiers which can be defined as CBR systems that are built for the classification problem - to determine whether or not an object is a member of a class, or which of several classes it may belong to. To build a CBR classifier, the cases stored in the case base are used as training data and the unseen cases are used as testing data. In this chapter, through combining the FR and CS processes, we present a novel and fast approach to extract case knowledge for building both efficient and competent CBR classifiers.

In Chapter 3, the task of FR is implemented by the rough set-based approach which is fast and effective in reducing the non-informative features. Like FR, CS is economical. The main objective of the CS process developed in this research is to extract case knowledge through identifying and removing redundant and noisy cases. In the context of CBR, a redundant case can be defined as follows. If two cases are the same (i.e., case duplication) or if one case subsumes another case, one of the cases duplicated or subsumed cases are considered to be redundant. They can be removed from the case base without affecting the overall problem-solving ability of the CBR system. The meaning of subsumption is as follows: Given two cases $e_p$ and $e_q$, when case $e_p$ subsumes case $e_q$, case $e_p$ can be used to solve more problems than $e_q$. In this case, $e_q$ is said to be redundant. On the other hand, the definition of noisy cases is very much depended on how we interpret the data distribution regions, and their association with the class labels. According to Brighton and Mellish [Brig 2002], there are two broad categories of class structures: the classes are defined by (1) homogenous regions; or (2) heterogeneous regions. In this research, we only consider the first category of data distribution. Based on the assumption that similar problems should have similar solutions, we define noisy cases as those that are very similar in their problem specifications yet propose different (or conflicting) solutions.

CS schemes are traditionally based on the $k$-NN principle, e.g., the Condensed Nearest Neighbor Rule (CNN) [Hart 1968] and the Wilson Editing method [Wils 1972]. There are several variations of the CNN and Wilson Editing method [Gate 1972][Ritt 1975][Tome 1976]. Based on the assumption that similar problems should have similar solutions, these methods examine the *k*-nearest neighbours of each case and then identify and remove noisy cases. In this chapter this group of methods are referred to as *k*-NN based CS methods.

Some CS strategies are derived from the area of case base maintenance (CBM), which includes policies of revising the organization and content of the case bases to facilitate the future reasoning of CBR systems. The concepts of case coverage and reachability [Smyt 1999a] are used to reduce redundant cases and thus build the case knowledge bases. As mentioned in Chapter 2, *coverage* of a case is the set of target problems (i.e., cases) that this case can be used to solve. The *reachability* of a target problem (i.e., a case) is the set of all cases that can be used to solve the target problem. The larger the coverage and the smaller the reachability of a case, the more important of this case in the CBR system. Thus, these two concepts can be used to identify redundant cases through examining the problem-solving ability of each case. Some such algorithms are developed in [Smyt 1999a][Smyt 1999b][Raci 1997][Cao 2003]. This research constructs and compares different case selection approaches based on the similarity measure and the concepts of case coverage and reachability, which are closely related to the *k*-NN based methods.

Case generation (CG) is an alternative approach of CS for reducing the size of the case base. New cases (or called prototypes) can be generated instead of selecting a subset of cases from the original case base. New cases, thus generated, has lower dimension than that in the original case base, for example, the fuzzy-rough method in [Pal 2004] generated cases of variable dimensions of lower size. On the other hand, the support vectors produced by SVM [Vapn 1998] or SVM ensemble [Vapn 1999][Kim 1997] can be also considered as cases selected as a subset of the original case base.

We have explained in Chapter 3 (see Section 3.1) that the process of FR plays an important role in CS and CG methods. This is because the feature importance (weight) is very closely related to the computations of case similarity, *k*-NNs, case-coverage and case-reachability. The non-informative features will mislead the results of these computations and therefore further affect the quality of selected cases.

In this chapter, we address this problem by combining the rough set-based FR (see Chapter 3) with the CS process. The feature importance is addressed by the reduct generation, whose computation complexity is only linear with respect to the number of attributes and cases. The features in the produced reduct are considered to be critical and those are removed are considered to be irrelevant. Different from some machine learning methods (e.g., neural networks) for feature weighting, after incorporating the FR in CS, the case representation should still be the same as that of the original case base. That is, each case is described by a set of features (subset of the original feature set) and a class label. This form of knowledge representation is easier to understand and more convenient for use in CBR systems.

In order to find the "best" sub-set of features (i.e., the set of features which can achieve the highest classification accuracy) that could be used by the CS process, we generate different approximate reducts in the proposed FR approach in Chapter 3 by fine-tuning the value of the consistency measurement of the sub-feature set. This allows the size of the approximate reduct to be controlled, and the "best" sub-set of features to be obtained.

To summarize, in this chapter, we construct and compare four different similarity measure-based case selection methods. Then FR is combined with CS to extract case knowledge for the CBR classifiers, which reduces both the burden of training and the need to acquire domain knowledge. The Wilson Editing method and SVM ensembles are implemented in the conducted experiments to provide comparisons with the proposed approach. The experimental results show that our proposed FR and CS methods, used individually or in combination, can preserve and even improve the classification accuracy while at the same time reduce the storage space. Furthermore, the combination of the FR and CS method is much faster than the combination of KPCA and SVMs techniques. The same data sets as that in Section 3.4 are used in the experiments, which have the number of features ranging from 17 to 2018, the number of cases ranging from 40 to 8124.

## 5.2 Case Selection Approach

In this section, we present four CS algorithms that are based on the similarity measure but that use of the case similarity in different ways. Algorithm 1 first selects cases having a large coverage and then, if the two cases have a similar coverage, selects the one with the smaller reachability set. CS algorithm 2 directly selects cases according to measurements of case similarity. The CS algorithms 3-4 are formed by incorporating the *k*-NN principle into CS algorithm 1 and CS algorithm 2, respectively.

Each of the four CS approaches has its own rationale. For CS algorithm 1, the similarity concept is used to compute a case's coverage and reachability values which can be interpreted as a measurement of its significance with respect to all other cases. A case is considered to be important if it "covers" many similar cases (with a similarity value greater than a threshold $\alpha$) all belonging to the same class. Here $\alpha$ is the similarity threshold between a particular case and its nearest boundary case. Since the cluster centres (cases) often have large coverage sets and the boundary cases have small coverage sets, this CS algorithm tends to select the cluster centres and remove the boundary cases.

CS algorithm 2 assumes that redundant cases can be found in densely populated clusters in the case base, with the similarity measure being used to describe the local density around a case. The more densely populated the cluster, the more redundant cases should be removed. A threshold can then be set up to determine the number of cases which should be deleted. Assume $e_p$ is a case which has been already selected. A case $e_q$ is considered to be redundant and should be removed if the similarity of $e_p$ and $e_q$ is greater than the given threshold and the classification label of $e_p$ is the same as that of $e_q$. As they tend to have different class labels from their neighbour cases, boundary cases will not be removed. Therefore, a number of representative interior cases and the boundary cases are preserved. This algorithm is fast, and it is suitable for case bases with high densities.

It is observed that, however, both CS algorithm 1 and 2 are vulnerable to noisy cases. The noisy cases mislead the computations of case coverage and reachability in the first CS algorithm, and they are often recognized as boundary cases which play important role in the second CS algorithm. In order to solve this problem, the *k*-NN principle is incorporated into the CS algorithms 1 and 2 to first detect and remove noisy cases, thereby forming algorithms 3 and 4. Here, the similarity concept is used to compute the *k*-nearest neighbors of each case. Based on the assumption that similar cases should have similar solutions, noisy cases are defined as cases having different class labels from the majority voting of their *k*-nearest neighbors. After the noisy cases are removed, the CS algorithms 1 and 2 are applied to remove the redundant cases. In this way, both noisy and redundant cases can be deleted from the case base.

Before providing a detailed description of the four CS algorithms, we shall define some related concepts. Assume there is a case base *CB*, the condition attribute set is *A*, the decision attribute set is *D*. *Coverage* of a case is the set of target problems (i.e., cases) that this case can be used to solve; while *reachability* of a target problem (i.e., a case) is the set of all cases that can be used to solve the target problem.

**Definition 5.1** The **coverage set** of a case *e* is defined as

$CoverageSet(e) = \{e' \mid e' \in CB, e' \text{ can be solved by } e\}$.

**Definition 5.2** The **reachability set** of a case e is defined as

$ReachabilitySet(e) = \{e' \mid e' \in CB, e \text{ can be solved by } e'\}$.

Notice that, in different situations, the meaning that a case can "solves" another case is different. In this chapter, we redefine the two concepts more explicitly as follows.

**Definition 5.3** Coverage Set of a case e is redefined as

$Cover(e) = \{e' \mid e' \in CB, sim(e, e') > \alpha, d(e) = d(e')\}$,

*where $\alpha$ is the similarity computed between case e and its nearest boundary case (the cases which have different class label of e); d is the decision attribute in D.*

Here the coverage set of a case *e* is the set of cases which fall in the disc centred at *e* with radius $\alpha$. We assume there are only one decision attribute *d*. It is straightforward to extend the definition to a situation with multiple decision attributes.

**Definition 5.4** Reachability Set of a case can be derived from the definition 5.3:

*Reach(e) = {e'| e'* $\in$ *CB, e can be covered by e'}.*

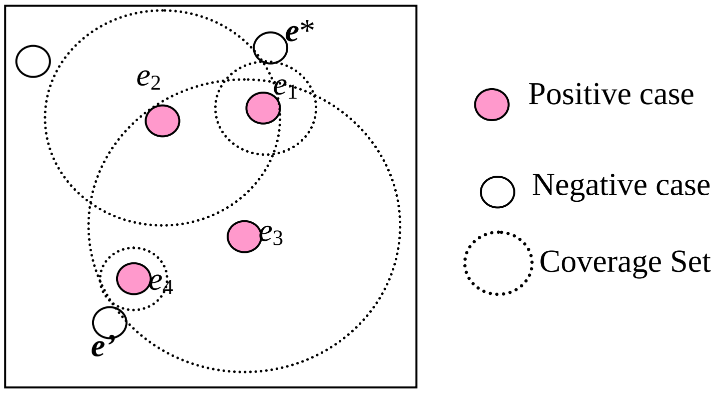

Fig. 5.1 The CoverageSet and ReachabilitySet

These definitions are illustrated in Fig. 5.1, where $\boldsymbol{e}^*$ is the nearest boundary case of cases $e_1$ and $e_2$; ***e'*** is the boundary case of $e_3$ and $e_4$. The dotted circle centred at a case represents the coverage set of this case. According to definitions 5.3-5.4, we have

*Cover*($e_1$) = {$e_1$}, *Cover*($e_2$) = {$e_1$, $e_2$}, *Cover*($e_3$) = {$e_1$, $e_2$, $e_3$, $e_4$}, *Cover*($e_4$) = {$e_4$}; and *Reach*($e_1$) = {$e_1$, $e_2$, $e_3$}, *Reach*($e_2$) = {$e_2$, $e_3$}, *Reach*($e_3$) = {$e_3$}, *Reach*($e_4$) = {$e_3$, $e_4$}.

The implication of the concepts of case coverage and reachability is that the larger the coverage set of a case, the more significant the case because it can correctly classify more cases based on the *k*-NN principle. In contrast, the larger the reachability set of a case, the less important the case in the case base because it can be reached by more existing cases.

In the example shown in Fig. 5.1, case $e_3$ is the most important case due to its largest coverage set and then follows case $e_2$. Notice that case $e_1$ and $e_4$ have the same size of coverage set but the reachability set of $e_4$ is smaller, $e_4$ is considered to be more critical. One focus of this research is the preservation of the competence (the number of cases the case base can cover) of the case bases. We attempt to build an algorithm (see Fig. 5.2) for selecting a subset of cases that would preserve the overall competence as compared to the original entire case base.

Since the algorithm involves the computation of coverage set and reachability set for each case in the original case base, the computation complexity of this algorithm is $O(m\times n^2)$, where $m$ is the number of condition attributes in $A$; $n$ is the number of cases in the case base. This algorithm must make three passes of the case base: the first to compute the similarity; the second to search the boundary case with the largest similarity $\alpha$ for each case; and the third pass is to find the nearest neighbors with a similarity with the current case that is larger than $\alpha$. Case selection algorithm 2, shown in Fig. 5.3, addresses this problem, requiring only one pass of the case base to compute the similarity between each two cases.

Algorithm 2 is totally similarity-based. If the similarity between a case $e^*$ and the current case $e$ is larger than a given threshold $\eta$ and they are with the same class label, $e^*$ will be considered as redundant and eliminated from the case base. This algorithm is suitable to the case bases with high density cases, while the CS algorithm 1 can be used on both sparse and dense cases. Notice that, the larger the parameter $\eta$, the more cases are selected by this algorithm. The value of $\eta$ can be determined either by the predefined size of the selected case base, or by the required classification accuracy.

```
Case Selection Algorithm 1

Input: CB – the entire case base;
          A – the entire condition attribute set;
         D – the decision attribute set.
Output: S – the selected subset of cases.

Step 1 Initialize $S = \varnothing$ (empty set).

Step 2 For every case e, $e \in CB$,
        Compute the coverage set and reachability set of e.

Step 3 Select the case which has the maximum coverage set.
        Ties are broken by selecting the case with smallest
        reachability set.

Step 4 The process stop when the selected cases cover the
           whole original case base CB.
```

Fig. 5.2 Case Selection Algorithm 1

```
Case Selection Algorithm 2

Input: CB – the entire case base;
          A – the entire attribute set;
         D – the decision attribute set.
Output: S – the selected subset of cases.

Step 1 Initialize resulted subset case base S = CB.

Step 2 For each case e in CB,
              Compute the similarity between e and all the cases
             in CB.

              If $sim(e, e') > \eta$, and $d(e') = d(e)$,
                  remove e' from S, $S = S - \{e'\}$;

Step 3 Output S.
```

Fig. 5.3 Case Selection Algorithm 2

Based on the concepts of coverage and reachability, case selection algorithm 1 could remove not only the redundant cases but also the noisy cases due to the small coverage sets of the noisy cases. However, the effectiveness of CS is still degraded by the existence of noisy cases. Cases located near the noisy cases tend to have smaller coverage sets than do other cases (see Fig. 5.4). As a result, cases close to noisy cases would be selected less often, which may lead to the loss of important information. As shown in Fig. 5.5, case selection algorithm 2 tends to eliminate redundant cases but was not able to effectively deal with noisy cases. A noisy case $e^*$ may be regarded as a boundary case. Since its class label could not be predicted by the cases which satisfy $sim(e, e^*) > \eta$, it could not be removed. This would result in the preservation of noisy cases (see Fig. 5.5) and the selection of an unsatisfactory case base.

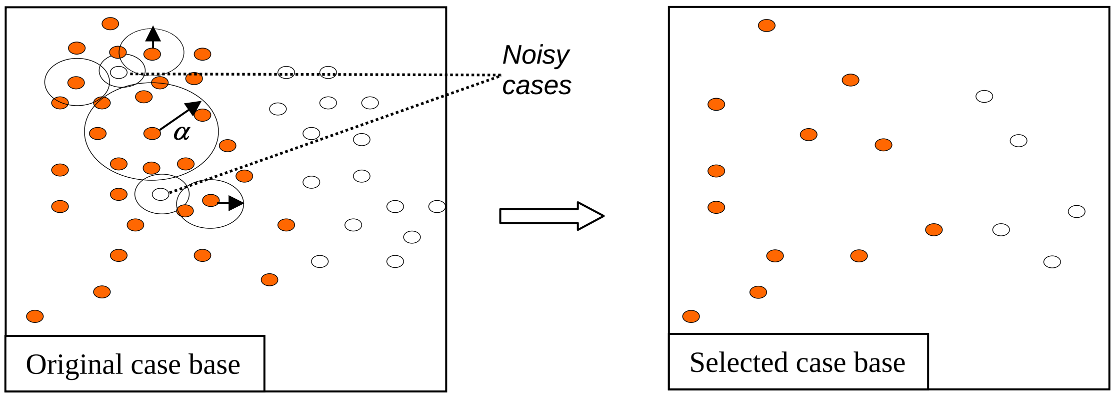

Fig. 5.4 Case selection algorithm 1 with noisy cases

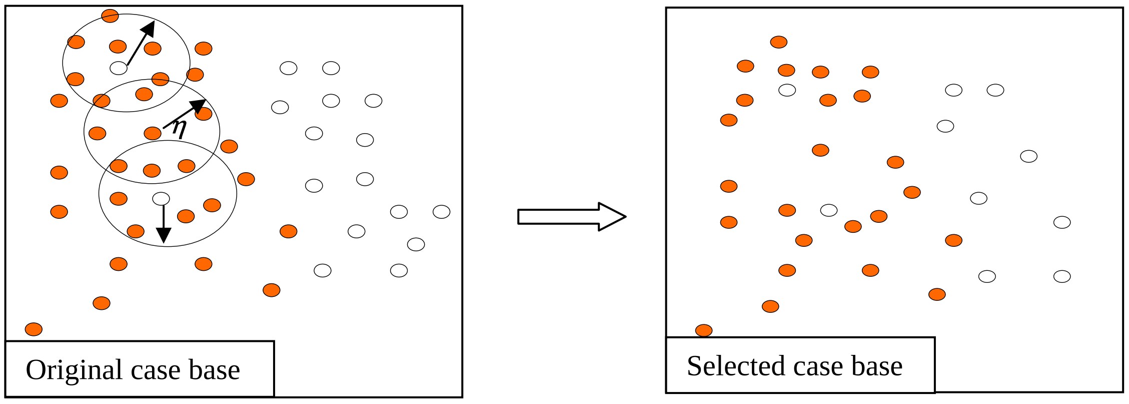

Fig. 5.5 Case selection algorithm 2 with noisy cases

To tackle the mentioned problems with case selection algorithm 1 and algorithm 2, the *k*-NN principle is incorporated to delete both the noisy cases and the redundant cases. Based on the similarity computation between cases, the *k*-NN principle is firstly used to find out the noisy cases. A case is said to be a noisy case if it cannot correctly classified by the majority of its *k*-nearest neighbors. Notice that, when the value *k* increases, the possibility of a case being a noise decreases, and vice versa. In this section, *k* is equal to the small odd number, 3. After the noisy case removal, the case selection algorithms 1 and 2 are then applied to further eliminate the redundant cases. The CS methods which incorporate the *k*-NN principle in the CS algorithms 1 and 2 are given as case selection algorithms 3 and 4 (see Fig. 5.6).

*Case Selection Algorithm 3*

*(1) Eliminate noisy cases using k-NN principle based on similarity computation.*
*(2) Remove redundant cases with case selection algorithm 1.*

*Case Selection Algorithm 4*

*(1) Eliminate noisy cases using k-NN principle based on similarity computation.*
*(2) Remove redundant cases with case selection algorithm 2.*

Fig. 5.6 Case Selection Algorithms 3 and 4

## 5.3 Combining Feature Reduction and Case Selection

In this section, we further explain the importance of combining FR and CS and gives two combination algorithms.

In most existing CS methods, as a first step, one computes the similarity between cases using all features involved and then the similarities are used to compute *k*-nearest neighbours, case coverage sets and reachability sets. The feature importance can be determined using three different methods: all the feature weights are equal; or the feature weights are determined in advance with domain knowledge; or the feature weights are learned by training some models. Each method, however, has some limitations which offer challenges to both FR and CS.

When all the feature weights are equal, and feature importance is consequently not considered, the computed similarities may be misleading. This will result in wrongly-computed *k*-nearest neighbours, case coverage and reachability sets. This will in turn directly affect the quality of the cases selected using our proposed CS algorithms.

The second and third methods are also problematic to determine feature importance. When the feature weights must be determined in advance using required domain knowledge, the knowledge is obtained either by interviewing experts - which is labour intensive - or is extracted from the cases - which adds to the burden of training. Similarly, when feature weights must be learnt using models such as neural networks or decision trees, the burden of training is again not trivial, and even after training these models, the case representation is then in the form of a trained neural network or a number of rules, which is not convenient for directly retrieving similar cases from a case base for the unseen cases.

We address these problems by combining the fast rough set-based FR approach with the CS algorithms. Feature importance is taken into account through reduct generation. The features in the reduct are regarded as the most important while other features are considered to be irrelevant. Reduct computation does not require any domain knowledge and the computational complexity is only linear with respect to the number of attributes and cases. After combining the FR method and CS algorithms, the case representation is still the same as that of the original case base. This form of knowledge representation is easier to understand and more convenient for retrieving unseen cases. Furthermore, since only the features in the reduct are involved in the computations in the CS algorithms, the running time for case selection is also reduced.

For the CBR classifiers, there are three main benefits from combining FR with CS: (1) classification accuracy can be preserved or even improved by removing non-informative features and redundant and noisy cases; (2) storage requirements are reduced by deleting irrelevant features and redundant cases; (3) the classification decision response time can be reduced because fewer features and cases will be examined when an unseen case occurs.

In this work, we propose two ways to combine FR and CS based on different definitions of a "best" sub-feature set (approximate reduct) $R^*$. The first method - called an "open loop" - applies the FR and CS sequentially and only once. The best approximate reduct is identified after applying FR alone. In construct, the second method can be regarded as a "close loop", which integrates FR and CS in an interactive manner, determining the best approximate reduct after applying both FR and CS approaches. The interaction of FR and CS is reflected in the identification of the suitable $\beta$ value.

In the first, "open loop", method, the "best" approximate reduct $R^*$ is defined as the approximate reduct which can achieve the highest accuracy after applying only the FR process. Such a best approximate reduct can be generated by iteratively tuning the value of the consistency measurement $\beta$ (see Section 3.3). For example, we start from the exact reduct with $\beta = 1$, and in each iteration reduce $\beta$ using a given parameter $\lambda = 0.01$. When the classification accuracy attains its maximum after applying FR alone, the approximate reduct is selected as $R^*$. In the following CS process, $R^*$ is used to detect redundant and noisy cases.

In the second, "close loop", method, the "best" sub-feature set is defined as the approximate reduct which can achieve the highest accuracy after applying both FR and CS. $R^*$ is determined much as in the first method. The value of consistency measurement $\beta$ is modified with step length $\lambda$ until it attains its maximum classification accuracy. Theoretically speaking, the "best" approximation reduct found using the second method is not necessarily the same as that found using the first method. The two combination methods are described as follows (see Figs 5.7-5.8):

```
RFRCS1 (Rough set-based Feature Reduction and Case Selection method 1):

Step 1 Initialize $P = \emptyset$, $Accr = \emptyset$, and $\beta = 1$. (P will store the reduced case bases
        after FR; Accr will store the corresponding classification accuracies using
        these reduced case bases.)

Step 2 While ($\beta > 0$)
            Implement Feature Reduction Algorithm; (Output the generated
            approximate reduct R)
            $P \leftarrow P \cup \Pi_R (U)$;
            Implement unseen case classification using $\Pi_R (U)$; (Output the current
            accuracy, a)
            $Accr \leftarrow Accr \cup \{a\}$;
            $\beta = \beta - \lambda$;

Step 3 Find $a^*$, $a^* = \max\{a \in Accr\}$; and find the corresponding $R^*$.

Step 4 Output the reduced final case base corresponding to $R^*$, denoted by $CB^* =
        \Pi_{R^*}(U)$.

Step 5 Let $CB^*$ be the input original case base. Apply the case selection algorithms
        1-4.
```

Fig. 5.7 The RFRCS1 Algorithm

To demonstrate their effectiveness, we use three main evaluation indices: storage requirement, classification accuracy, and classification speed. For FR and CS processes, storage requirement has different meanings. The storage in FR is the percentage of preserved features after reducing features; in CS, storage is the percentage of selected cases. The classification accuracy is the percentage of the unseen cases which can be correctly classify. The classification speed is used in Section 5.4.2 to examine the efficiency of the built classifier using the FR and CS combinations. In Section 5.4.3, all these evaluation indices are considered in the comparisons between our approach with the KPCA and SVMs. The experiments used four real life data sets, House-votes-84 database, Text document sets, Mushroom database, and Multiple Features, which are the same as those used in Section 3.4.1.

## 5.4.1 Case Selection

The CS algorithms developed in Section 5.2 are applied to the real life data sets and compared with the traditional Wilson Editing. Note that the generation method of the training data and testing data is the same as that in Section 3.4.1 in Chapter 3:

(1) For the *House-votes-84* data, four splits are generated by randomly selecting 20%, 30%, 40%, and 50% as the testing data; the corresponding left data are used as the training data.
(2) For the *Text data* and *Mushroom data,* we randomly select 80% documents in each text data set as the training data and the remaining 20% is used as the testing data.
(3) For the *Multiple Features* data, we use the training/ testing data sets in the original database, which has 1000 training samples and 1000 testing samples.

Tables 5.1 and 5.2 demonstrate the reduced storage and improved accuracy when using different CS algorithms. P(W), P(1), P(2), and P(4) represent the classification accuracy using Wilson Editing, case selection algorithms 1, 2, and 4, respectively. *Notice that the results of algorithm 3 are very similar to those of algorithm 1. Due to space limitations, they are not included in Tables 5.1-5.2 and related results in the following sections.* Here "storage" means the proportion of cases which are selected in the final case base. In case selection algorithm 2, the parameter $\eta = 0.99$.

Table 5.1 Case selection using the house-votes-84 data set

| Split | P0 (%) | P (W) (%) | Storage | P(1) (%) | Storage | P(2) (%) | Storage | P(4) (%) | Storage |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 93.10 | 95.40 | 92.82 | 93.10 | 74.43 | 93.10 | 81.03 | 95.40 | 73.85 |
| 2 | 92.31 | 94.62 | 93.11 | 92.31 | 84.26 | 92.31 | 81.97 | 94.62 | 75.08 |
| 3 | 93.68 | 95.40 | 92.34 | 94.83 | 67.43 | 93.68 | 85.44 | 95.40 | 77.78 |
| 4 | 94.01 | 95.39 | 91.74 | 94.93 | 83.9 | 94.01 | 85.78 | 95.39 | 77.52 |
| Avg. | 93.28 | ***95.20*** | ***92.50*** | 93.79 | 77.40 | 93.28 | 83.56 | ***95.20*** | ***76.06*** |

Table 5.2 Case selection using text data sets

| Data | P0 (%) | P(W) (%) | Storage | P(1) (%) | Storage | P(2) (%) | Storage | P(4) (%) | Storage |
|---|---|---|---|---|---|---|---|---|---|
| Text1 | 62.50 | 75.00 | 40.43 | 62.50 | 87.23 | 62.50 | 89.36 | 75.00 | 38.30 |
| Text2 | 62.50 | 75.00 | 17.65 | 62.50 | 54.90 | 75.00 | 41.18 | 75.00 | 17.65 |
| Text3 | 33.33 | 66.67 | 75.34 | 33.33 | 90.41 | 33.33 | 72.60 | 66.67 | 67.12 |
| Text4 | 37.93 | 31.03 | 82.86 | 34.48 | 82.86 | 37.93 | 74.29 | 31.03 | 80.00 |
| Text5 | 56.25 | 56.25 | 73.68 | 56.25 | 94.74 | 56.25 | 71.05 | 56.25 | 71.05 |
| Text6 | 77.78 | 33.33 | 8.99 | 77.78 | 14.82 | 77.78 | 10.09 | 33.33 | 6.56 |
| Text7 | 51.19 | 40.48 | 22.97 | 44.05 | 54.05 | 51.19 | 43.24 | 40.48 | 22.97 |
| Text8 | 72.79 | 71.09 | 32.20 | 72.11 | 44.00 | 68.37 | 25.20 | 72.45 | 18.40 |
| Avg. | 56.78 | 56.11 | 44.27 | 55.38 | 65.38 | ***57.79*** | ***53.38*** | ***56.28*** | ***40.26*** |

Table 5.1 (the house-votes data) shows that after case selection, all the CS algorithms were able to reduce cases while preserve or even improve classification accuracy. The Wilson Editing and case selection algorithm 4 attain greatest accuracy; while algorithm 4 has more powerful capability to reduce useless cases than other algorithms do. Table 5.2

shows the results for the text data sets. Algorithm 2 is most accurate. Algorithm 4 produced the smallest reduced case base with respect to the number of cases.

To summarize, both Tables 5.1 and 5.2 show results for algorithm 4 that are satisfactory in terms of both classification accuracy and storage requirements after the case selection.

### 5.4.2 Combining FR and CS

In this section, we discuss some experiments using RFRCS1 and RFRCS2 (see Section 5.3) that were conducted to show the positive impact of the proposed rough set-based FR method on the CS algorithms. The two main evaluation measurements are still storage and accuracy. Comparisons are made based on the *k*-NN classifier, using different CS algorithms in combination with FR. Here *k* is set to a small odd number, 3.

In this section, let P(F+W), denote the classification accuracy of the combination of the rough set-based FR and Wilson Editing; and P(F+1), P(F+2), P(F+3), P(F+4) that of case selection algorithms 1 to 4. The final reduced case base is the case base containing the reduced feature set and the selected cases. The results of P(3) and P(F+3) are similar to those of P(1) and P(F+1) and are therefore not shown. Since the combination method RFRCS2 requires a greater computational effort, in this section we mainly conduct the experiments using the algorithm RFRCS1.

#### *RFRCS1: Storage requirement and classification accuracy*

(1) House-votes-84

Table 5.3 shows the results when using RFRCS1. On this data set, RFRCS1 incorporates the proposed fast rough set-based FR approach into the CS algorithms. Obviously, the combined algorithms are more accurate and require less storage space than the approaches that make use of individual CS algorithms alone. Here $\beta = 0.95$.

```
RFRCS2:

Step 1 Initialize FCB = $\varnothing$, Accr = $\varnothing$ and $\beta = 1$. (FCB will store the final
        reduced case bases after both FR and CS.)

Step 2 While ($\beta > 0$)
            Implement Feature Reduction Algorithm; (Output the reduced
            feature set R)

            Implement Case Selection Algorithms 1-4, where
            the original input case base CB = $\Pi_R(U)$; (Output final
            reduced case base FCB.)

            Implement unseen case classification using FCB; (Output the
            current accuracy, a)
            $Accr \leftarrow Accr \cup \{a\}$;
            $\beta = \beta - \lambda$;

Step 3 Find $a^*$, $a^* = \max\{a \in Accr\}$; and find the corresponding $R^*$.

Step 4 Output the reduced final case base corresponding to $R^*$,
        denoted by $FCB^* = \Pi_{R^*}(U)$.
```

Fig. 5.8 The RFRCS2 Algorithm

Obviously, the second combination method, RFRCS2, requires more computational effort because the "best" approximate reduct depends on both FR and CS processes. For this reason, we mainly use the RFRCS1 method to test the performance of FR-CS combinations.

## 5.4 Experimental results

In this section, we test our proposed CS algorithms, the combinations of the rough set-based FR and CS, and provide comparisons with KPCA and SVMs techniques.

Table 5.3 Applying RFRCS1 to house-votes-84 ($\beta = 0.95$)

| Split | P(W) (%) | P(F+W) (%) | P(1) (%) | P(F+1) (%) | P(2) (%) | P(F+2) (%) | P(4) (%) | P(F+4) (%) |
|---|---|---|---|---|---|---|---|---|
| 1 | 95.40 | 97.70 | 93.10 | 94.25 | 93.10 | 97.70 | 95.40 | 97.70 |
| 2 | 94.62 | 96.15 | 92.31 | 94.62 | 92.31 | 94.62 | 94.62 | 96.15 |
| 3 | 95.40 | 97.13 | 94.83 | 95.98 | 93.68 | 95.98 | 95.40 | 97.13 |
| 4 | 95.39 | 96.77 | 94.93 | 95.39 | 94.01 | 96.31 | 95.39 | 96.77 |
| Avg. | 95.20 | **96.94** | 93.79 | 95.06 | 93.28 | 96.15 | 95.20 | **96.94** |
| | +1.74 | | +1.27 | | +2.87 | | +1.74 | |

Algorithms (F+W) and (F+4) are shown to be most accurate. The (F+4) algorithm also has the best classification accuracy and the most reduced storage requirement. This is because the algorithm 4 is able to reduce the number of cases more effectively than algorithm that uses Wilson Editing (Table 5.1). We can conclude that the fast rough set-based FR approach using case selection algorithm 4 is superior to other FR and CS algorithms used either individually or in combination.

Table 5.4 Applying RFRCS1 to text data sets ($\beta = 1$)

| Data | P(W) (%) | P(F+W) (%) | P(1) (%) | P(F+1) (%) | P(2) (%) | P(F+2) (%) | P(4) (%) | P(F+4) (%) |
|---|---|---|---|---|---|---|---|---|
| Text1 | 75.00 | 87.50 | 62.50 | 75.00 | 62.50 | 75.00 | 75.00 | 87.50 |
| Text2 | 75.00 | 75.00 | 62.50 | 62.50 | 75.00 | 62.50 | 75.00 | 75.00 |
| Text3 | 66.67 | 66.67 | 33.33 | 33.33 | 33.33 | 33.33 | 66.67 | 66.67 |
| Text4 | 31.03 | 44.83 | 34.48 | 44.83 | 37.93 | 41.38 | 31.03 | 44.83 |
| Text5 | 56.25 | 68.75 | 56.25 | 75.00 | 56.25 | 75.00 | 56.25 | 68.75 |
| Text6 | 33.33 | 44.44 | 77.78 | 77.78 | 77.78 | 77.78 | 33.33 | 44.44 |
| Text7 | 40.48 | 41.67 | 44.05 | 45.24 | 51.19 | 53.57 | 40.48 | 41.67 |
| Text8 | 71.09 | 70.75 | 72.11 | 68.71 | 68.37 | 63.27 | 72.45 | 70.41 |
| Avg. | 56.11 | **62.45** | 55.38 | 60.30 | 57.79 | 60.23 | 56.28 | **62.41** |
| | +6.34 | | +4.92 | | +2.44 | | +6.13 | |

(2) Text data sets

This section examines the impact of FR on CS using the text data sets. Table 5.4 displays the text data set results. They are similar to those for the house-votes data set, except that the improvement in accuracy is much greater after incorporating FR to CS.

(3) Mushroom data

Table 5.5 shows the experimental results after applying RFRCS1 to the mushroom data set. Only the results of the case selection algorithm 1 and its combination with FR are contained in the table. This is because that, except the case selection algorithm 1, none of other algorithms were able to remove cases from the original case base. This is because that the Mushroom data is sparse and the CS algorithms 2 and 4 are suitable to the highly dense data. The classification accuracy of FR approach was the same as the original accuracy using the entire case base, 1. Therefore, P0 = P(FR) = P(W) = P(2) = P(4) = 1. Table 5.5 shows the impact of feature reduction on case selection algorithm 1. On average, the classification accuracy after applying the combination of FR and the CS algorithm 1 increases by 9.3% from P(1) = 89.6% to P(F+1) = 98.9%. Here, the storage is with respect to cases instead of that of features. It is the percentage of cases which need to be stored in the final reduced case base after applying the algorithm 1. For the FR in algorithm (F+1), $\beta$ is set to be 1. There are five features in the generated reduct so the storage requirement with respect to the feature set is 22.7% of the original feature set.

Table 5.5 Applying RFRCS1 to mushroom data

| Split | P(1) (%) | P(F+1) (%) | Storage (%) |
|---|---|---|---|
| 1 | 87.00 | 100.00 | 25.50 |
| 2 | 89.33 | 98.67 | 7.43 |
| 3 | 90.50 | 99.50 | 11.33 |
| 4 | 91.60 | 97.60 | 10.00 |
| Avg. | 89.61 | 98.94 | 13.57 |
| | +9.33% | | 13.57 |

To conclude, when the combination method RFRCS1 is applied, the results of almost all of the data sets and of all of the proposed CS algorithms are positive. The classification accuracy and storage space requirement show a notable improvement: when the CS algorithms are applied to the house-votes-84 data, the average increase in accuracy is 1.91% (Table 5.3). Applied to the text data sets, it is 4.95% (Table 5.4), and applied to the mushroom data set it is 9.33% (Table 5.5). These improvements in accuracy are respectively achieved only with 51.13% (Table 5.4 in Chapter 3), 14.9% (Table 5.5 in Chapter 3), 22.73% (Section 3.4.1) of the original features for the three data sets. In almost all the testing, the combination of proposed rough set based FR approach with the CS algorithm 4, denoted by (F+4), is the most promising algorithm in terms of both classification accuracy and storage requirement.

## *RFRCS1: Classification Efficiency*

This section describes some experiments carried out to determine the efficiency of case retrieval or unseen case classification after reducing both features and cases. The testing is also based on the algorithm RFRCS1.

Table 5.6 Speed of case classification using RFRCS1

| Data sets | T_FR | T_CS | T0 | T | $T_s$ |
|---|---|---|---|---|---|
| House-votes-84 | 0.343 | 0.004 | 0.10 | 0.07 | 0.03 |
| Text data | 5.597 | 0.020 | 1.58 | 0.02 | 1.56 |
| Mushroom data | 0.600 | 0.008 | 1.14 | 0.93 | 0.21 |

Table 5.6 shows the average T_FR, T_CS, T0, T and $T_s$ using the three data sets. T_FR and T_CS are the average time cost in the FR process and the case selection method 4, respectively. T0 is the average time needed to classify one unseen case using the entire original data sets. T is the average time needed to classify one unseen case using the reduced data set RFRCS1. $T_s$ = T0 – T. Since there are much fewer features and cases in the reduced data sets than those in the entire data sets, the case retrieval time of CBR

classifiers using the reduced data sets will be much less than the time using the entire data sets. For this reason, $T_s$ describes the amount of time that is saved for an unseen case classification due to this data compression. T_FR, T_CS, T0, T, and $T_s$ are represented in seconds. The efficiency of case classification is improved using the reduced data sets. Although the average saved time of identifying only one unseen case is not notable, it could be significant using all the testing cases. For example, for the house-votes-84 data, the total saved time for predicting the class labels for all the 217 testing cases is $(0.03\times217) \approx 6.51$ seconds.

### *RFRCS2: Storage requirement and classification accuracy*

Previously, we performed some experiments using the FR and CS combination method RFRCS1. The "best" $\beta$ and approximate reduct were determined through only in the FR process. Here the RFRCS2 is applied to real life data, where the most suitable $\beta$ is obtained when the final accuracy attains its maximum after both FR and CS. The experimental results show that the best $\beta$ values found in RFRCS2 are not necessarily the same as those in RFRCS1. Compared with RFRCS1, using this kind of combination of FR with CS, the classification accuracy is shown to be further improved and/or the storage space could be further reduced.

(1) house-votes-84

The most suitable $\beta$ found for this data set is 0.90, but not 0.95 in RFRCS1. There are seven features (43.75% of the original features) in the corresponding approximate reduct. Compared with RFRCS1, the classification accuracy is preserved using the reduced case base while the number of features is further reduced. Fig. 5.9 shows the relationship between P(F+4) and $\beta$ values. When $\beta \in [0.90, 0.95]$, the accuracy attains its maximum, 0.977. Since the smaller the $\beta$ value, the fewer the features in the corresponding approximate reduct, $\beta$ is set to be 0.90 so that the accuracy can be preserved whereas the number of features is minimum. The similar results could be achieved using other CS algorithms. Table 5.7 shows the results in detail. Here "Storage" means the storage requirement with respect to the features instead of the cases; "Max accuracy" is the highest classification accuracy obtained by combinations of FR with different CS algorithms. As can be seen in Table 5.7 using RFRCS2, the maximum accuracies have been preserved and the feature storage requirement decreases by 9.38% from 53.13 (using RFRCS1) to 43.75 (using RFRCS2).

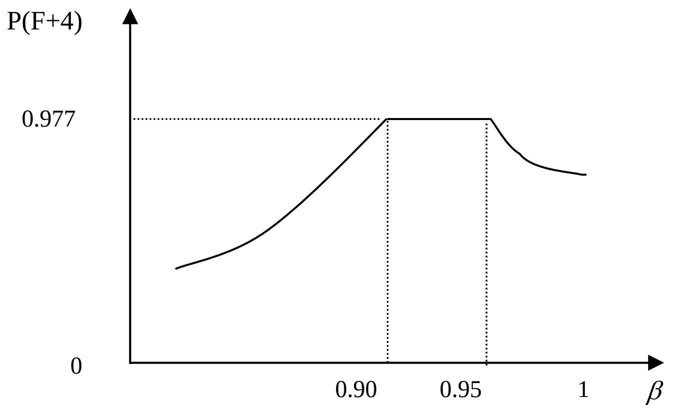

Fig. 5.9 P(F+4) vs. $\beta$ values

Table 5.7 Applying RFRCS2 to House-votes data ($\beta = 0.90$)

| Splits | Storage (%) (RFRCS1) | Storage (%) (RFRCS2) | Max. Accuracy (%) |
|---|---|---|---|
| 1 | 56.25 | 43.75 | 97.70 |
| 2 | 50.00 | 43.75 | 96.15 |
| 3 | 50.00 | 43.75 | 97.13 |
| 4 | 56.25 | 43.75 | 96.77 |
| Avg. | 53.13 | 43.75 | 96.94 |

(2) Text data sets

In the experiments using *RFRCS1*, $\beta$ is set to be 1 for the text data sets. This is because when $\beta = 1$, the classification accuracy attains its maximum with the case base after only applying a FR process to reduce the number of features.

Table 5.8 RFRCS2 with various $\beta$ values on Text data sets

| Text data | $\beta$ | Storage (%) (RFRCS2) | Storage (%) (RFRCS1) | Max. Accuracy (%) | Max.Accuracy (%) (RFRCS1) |
|---|---|---|---|---|---|
| Text1 | 0.85 | 6.25 | 12.50 | 100.00 | 87.50 |
| Text2 | 0.70 | 5.71 | 9.05 | 87.50 | 75.00 |
| Text3 | 0.80 | 9.93 | 28.48 | 66.67 | 66.67 |
| Text4 | 1.00 | 10.51 | 10.51 | 44.83 | 44.83 |
| Text5 | 0.85 | 3.62 | 9.59 | 81.25 | 75.00 |
| Text6 | 1.00 | 31.53 | 31.53 | 77.78 | 77.78 |
| Text7 | 1.00 | 7.81 | 7.81 | 53.57 | 53.57 |
| Text8 | 1.00 | 3.37 | 3.37 | 72.10 | 62.45 |
| Avg. | 0.90 | 9.84 | 14.11 | 72.96 | 67.85 |
| Comparisons | | - 4.27 | | +5.11 | |

In this section, using the combination method RFRCS2, various best $\beta$ values are found for different text data sets. Table 5.8 demonstrates that with these different $\beta$ values, the average required storage space could be further reduced from 14.11% to 9.84% of the original features, the average maximum accuracy increases by 5.11% from 67.85% to 72.96%.

(3) Mushroom data.

After applying RFRCS2, the best $\beta$ in the mushroom data set is the same as in RFRCS1, i.e., $\beta$ = 1. Therefore, the results of RFRCS2 with respect to the storage requirement, classification accuracy are the same as with RFRCS1.

*Discussions*: Using the combination method RFRCS2, the best $\beta$ values which achieve the maximum accuracies after applying both FR and CS can be found for the real life data sets. Compared with RFRCS1, more computational efforts are required because the CS process is involved in tuning the $\beta$ values. The accuracy is shown to be preserved (for

House-votes-84 and Mushroom data) and even improved (for the text data sets) and the storage requirement of the feature set is further reduced (for House-votes-84 and text data sets). The users can choose either RFRCS1 or RFRCS2 to construct the final case base for the CBR classifier. For some large case bases, users can select RFRCS1 which has less computational load with still satisfactory accuracy.

### 5.4.3 Comparisons: Rough Set-Based FR and CS vs. KPCA and SVMs

In Section 3.4.2 in Chapter 3, the fast rough set-based FR method is compared with the widely used KPCA. In this section, we make some comparisons to further demonstrate the effectiveness of our developed FR and CS methods. FRCS1 is compared with the combination of KPCA and SVM ensembles.

The experimental setup is the same as that in Section 3.4.2 as follows. (1) Data - Since KPCA can only handle numerical data, the data set of *Multiple Features* is used to conduct these experiments. (2) Data splitting - We use the training/testing data sets in the original database of *Multiple Features*, which has 1000 training samples and 1000 testing samples. (3) Data preprocessing - The numerical data needs to be discretized before the use of rough sets in the FR process. (4) Performance Evaluation - Four main evaluation indices are used including training time, retrieval time, storage requirement and classification accuracy. Here the unit of time is second.

In this section, RFRCS1 (see Section 5.3) is compared with the combined KPCA and SVM ensembles. In RFRCS1, the CS algorithm 4 is used for case selection after the FR process. From Table 5.6 in Chapter 3, we notice that when 1.87% features (i.e., 13 features) are selected, the accuracy attains its maximum. Therefore, here $\beta$ value is set as 0.99 and the number of eigenvectors extracted by KPCA is determined as 13. On the other hand, after the feature extraction of KPCA, a transformed data set is obtained which has lower dimensionality. This new reduced data set is then used in constructing the multiple SVM classifiers. Here we use one-against-all method to deal with the multiple class problem, and the final results are combined based on the majority voting rule. Since the data set contains 10 classes, 10 SVMs are constructed and trained in the experiments.

Tables 5.9-5.10 show the results of RFRCS1 and the combination of KPCA and SVMs. Here the "Storage" is the percentage of selected cases instead of features; "T_train" is the processing time of the CS method. In RFRCS1, with the increase of $\eta$ value, the storage, training time, retrieval time and the accuracy also increase. The combination of KPCA and 10 SVMs totally extracts 408 support vectors (prototypical cases), and the classification accuracy is 93.8%, which is slightly higher than the maximum accuracy obtained by rough set-based method, 91.5%. However, the total training time of the 10 SVMs exceeds 6 hours. In contrast, the combination of rough set-based FR and CS approach requires much less processing time and can still achieve satisfactory classification accuracy. Since SVMs has reduced more storage than FRCS1 does, generally, the combination of KPCA and SVMs is more efficient in the end. However, this depends on the frequency of updating the case bases and retraining of SVMs. The higher the frequency, the larger the possibility of that our FRCS methods outperform the KPCA and SVMs.

Table 5.9 RFRCS1 ($\beta = 0.99$)

| $\eta$ | Storage (%) | T_train | T_retrieval | Accuracy (%) |
|---|---|---|---|---|
| 0.55 | 10.10 | 60 | 6 | 65.90 |
| 0.60 | 24.10 | 160 | 21 | 78.80 |
| 0.65 | 46.90 | 383 | 68 | 86.50 |
| 0.70 | 76.20 | 666 | 168 | 90.30 |
| 0.75 | 94.00 | 858 | 250 | 91.40 |
| 0.80 | 99.10 | 914 | 277 | 91.50 |
| 0.85 | 99.60 | 919 | 278 | 91.50 |
| >0.90 | 99.80 | 918 | 278 | 91.50 |
| Avg. | 68.73 | 609.75 | 168.25 | 85.93 |

Table 5.10 KPCA and 10 SVMs (Accuracy = 93.80%)

| SVM_NO | T_train | Vec_NUM |
|---|---|---|
| 1 | 2236 | 33 |
| 2 | 2193 | 44 |
| 3 | 2193 | 34 |
| 4 | 2227 | 42 |
| 5 | 2225 | 36 |
| 6 | 2400 | 50 |
| 7 | 2554 | 40 |
| 8 | 2285 | 37 |
| 9 | 2477 | 43 |
| 10 | 2509 | 49 |
| SUM | 23299 | 408 |

SVM_NO: the ID number of the trained SVM classifier; Vec_NUM: the number of generated support vectors. The retrieval time is not listed in Table 11 because it is trivial compared with the training time.

In RFRCS1, we use *k*-NN principle to classify unseen cases, where the *k* value may affect the classification accuracy. Here we report the results of some testing using different *k* values in RFRCS1 on the data set of *Multiple Features*. Let $k = 1, 3, \ldots, \sqrt{n}$, where *n* is the number of training samples. It is demonstrated in Fig. 5.10 that, the accuracy attains its maximum when *k* = 3. Therefore, we set *k* = 3 in the experiments in previous sections.

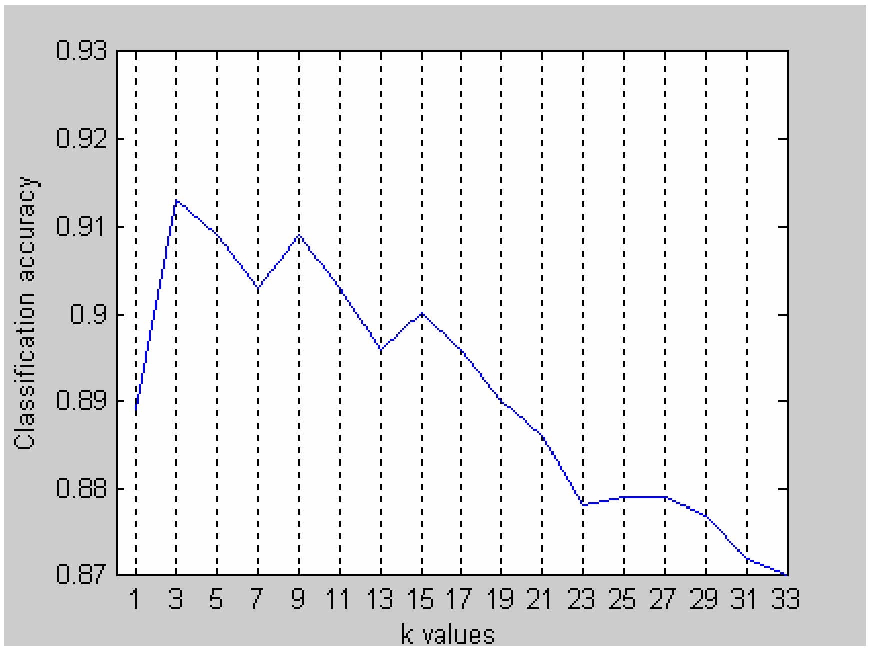

Fig. 5.10 The effect of k value on accuracy

Table 5.11 Comparisons between FR and CS approach and KPCA and SVMs

| Comparisons | | | FR and CS approach | KPCA and SVMs |
|---|---|---|---|---|
| Qual. Indices | Rationale | | Attribute dependency | Data variance |
| | Training Type | | Supervised | Unsupervised |
| | Data Type | Sym. | Yes | No |
| | | Num. | Yes (need discretization) | Yes |
| | Reduced feature set | | A subset of original features | A transformed feature set |
| Quan. Indices | Training time | | 609.75 | 23299 |
| | Accuracy | | 91.50 | 93.80 |

Qual. = Qualitative; Quan. = Quantitative; Sym. = Symbolic; Num. = Numerical

*Discussions*: Table 5.11 shows the comprehensive comparisons between the developed rough set-based FR and CS approach and the combination KPCA and SVMs. They are based on different rationale and can be used to different data types; the rough set-based FR is supervised learning while KPCA is unsupervised; the rough set-based FR generates a subset of the original features and KPCA extracts a set of transformed features; the combination of FR and CS is fast while KPCA and SVMs achieve a slightly higher accuracy. Users can choose either of the combination methods based on the used data and their requirements on efficiency and accuracy.

## 5.5 Discussion

It should be pointed out that, we have only considered the case bases which consist of homogeneous data regions, in which the noisy cases are defined as the cases that cannot be correctly classified by their k-nearest neighbors. Therefore, our CS approach cannot be directly used on case bases containing heterogeneous data regions, which may result that some useful cases are misclassified as noisy cases.

## 5.6 Summary

In this chapter, we describe four similarity-based CS algorithms and their combinations with the fast rough set-based FR in Chapter 3. The developed CS algorithms can remove not only the redundant cases but also the noisy cases. It can be shown that, compared with using the original case base, higher classification accuracy and less storage space requirement could be obtained with each individual CS algorithm. By combining the FR and CS processes, we could further enhance the accuracy and reduce the storage. Two methods of combination, RFRCS1 and RFRCS2, are developed based on different definitions of the "best" value of the consistency measurement.

The experimental results show that the case selection algorithm 4 using the proposed rough set-based FR algorithm, denoted by (F+4), is the most promising one which has the highest accuracy and the least storage requirement. The enhanced efficiency using the reduced data sets is also demonstrated through the experimental results in Section 5.4.2. Comparisons are also made between RFRCS1 and RFRCS2 in this section. RFRCS2 shows higher accuracy and lower storage load but requires more computational efforts. In section 5.4.3, some comparisons are also made between the RFRCS1 and the combination of KPCA and SVMs. These two combination methods have different characteristics, based on which users can select one of them to reduce both the dimensionality and size of data.

# Chapter 6

# Rough LVQ-Based Case Generation

Case generation (CG) is an alternative approach of CS for case knowledge extraction through reducing the size of case bases. Different from CS in Chapter 5, CG produces a new set of prototypical cases instead of selecting a subset of cases in the original case base. These generated prototypical cases are considered to be the most representative cases which can cover the whole case base.

In this chapter, we develop a case generation approach which integrates fuzzy sets, rough sets and learning vector quantization (LVQ). If the feature values of the cases are numerical, fuzzy sets are firstly used to discretize the feature spaces. Secondly, the fast rough set-based FR is incorporated to identify the significant features. Finally, the representative cases (prototypes) are then generated through LVQ learning process on the case bases after FR. As a result, a few of prototypes are generated as the representative cases of the original case base. These prototypes can be considered as the extracted case knowledge which can improve the problem-solving efficiency and enhance the understanding of the case base. Three real life data are used in the experiments to demonstrate the effectiveness of this case generation approach. Several evaluation indices, such as classification accuracy, the storage space, case retrieval time and clustering performance in terms of intra-similarity and inter-similarity, are used in these testing.

## 6.1 Introduction

Similar to the task of CS, CG is to extract the most representative cases from a given case base, which can build a new case base with smaller number of cases (i.e., a case knowledge base). The cases generated by CG process are not necessarily the data points of the given case base. The case representation form of these produced prototypical cases may be different from that of the original cases. For example, the support vectors which generated by SVMs and the rules learned by the decision trees, where some features may not exist in the newly extracted prototypical cases. Some other related work to CG includes [Chan 1974] [Domi 1995] [Salz 1991] which generates cases through merging the cases in the same class or modifying the original cases. A rough-fuzzy CG technique is proposed in [Pal 2004] which identifies the cluster granules as the newly generated cases.

In this chapter, we discuss the CG techniques in the context of CBR classifiers which have been defined in Chapter 5. The purpose is to extract case knowledge to build both compact and competent CBR classifiers. Generally speaking, with more cases, the case bases will be more competent and therefore higher problem-solving accuracy can be obtained. On the other hand, it is obvious that the larger the size of a case base, the lower the case retrieval speed. It is difficult to achieve the optimal classification accuracy and case retrieval time simultaneously. If the size of a case base is reduced, the competence of the case base may be hurt because of the removal of some important cases. In this chapter, we attempt to make a trade-off by developing a rough learning vector quantization (LVQ)-based case generation approach. A few of prototypes are generated to represent the entire case base without much loss of the competence of the original case base.

As a necessary pre-processing of LVQ-based case generation, the fast rough set-based FR method developed in Chapter 3 is used to select the relevant features and eliminate the irrelevant ones, which can modify the similarity among cases and achieve better clustering performance. In Chapter 3, the proposed FR method has been proved to be able to find approximate reducts quickly and effectively. The features in the resulted approximate reduct are considered to be important for the CBR classifiers. In this chapter, before applying FR, fuzzy sets are used to discretize the numerical attribute values to generate indiscernibility relation and equivalence classes of the given case base. Triangular membership functions are applied in the discretization of feature spaces.

Learning vector quantization is then applied to extract the prototypical cases to represent the entire case base. LVQ is a competitive algorithm, which is considered to be a supervised version of the Self-Organizing Map (SOM) algorithm. The SOM algorithm [Koho 1988][Koho 1998] constructs a stable topology preserving mapping from the high-dimensional space onto map units in such a way that relative distances between data points are preserved. Mangiameli et al. [Mang 1996] demonstrated that SOM is a better clustering algorithm than hierarchical clustering with regard to clustering data with overlapped dispersion, irrelevant variables, outliers or different sized populations. Their study also proved that SOM is insensitive to learning rates which vary in the self-organizing process, and the clusters resulted from SOM are robust. Pal et al. used SOM to extract prototypical cases in [Pal 2004] and reported a compact representation of data.

Kohonen pointed out in [Koho 1988], “the SOM has not been meant for statistical pattern recognition; it is a clustering, visualization, and abstraction method. Anybody wishing to implement decision and classification processes should use Learning Vector Quantization (LVQ) instead of SOM”. Since we focus on the classification problems in this chapter, LVQ is used to generate prototypical cases.

After the CG process, the original case base can be reduced to a few prototypes which can be directly used to predict the class label of the unseen cases. These prototypes can be regarded as the specific domain knowledge which is extracted from the case base. This will speed up the case retrieval and make the case base be more easily understood. On the other hand, since the most representative cases are generated, case base competence can be also preserved. Therefore, using our developed rough LVQ-based case generation approach, the retrieval speed, clustering performance, and the understanding of the case base are all improved without decreasing the classification accuracy.

The reminder of this chapter is organized as follows. In Section 6.2, fuzzy sets are applied to discretize the continuous-valued attributes of the cases. Three triangular membership functions are used to describe each attribute. In Section 6.3, the fast rough set-based FR method in Chapter 3 is firstly used to reduce the irrelevant features, which is considered to be the necessary pre-processing of the following LVQ learning process. Then the supervised learning process of LVQ is presented to generate prototypical cases for the given case base. To validate the developed rough LVQ case generation approach, section 6.4 presents the experimental results on three real life data. The classification accuracy, case retrieval speed, intra- similarity and inter- similarity are used as the indices to evaluate the performance of our approach. Comparisons are made among the developed rough LVQ approach, LVQ, SOM, and Random case generation methods.

## 6.2 Fuzzy Discretization of Feature Space

The rough set-based FR methods developed in Chapter 3 are all built on the basis of indiscernibility relation. If the attribute values are continuous, the feature space needs to be discretized to define the indiscernibility relations and equivalence classes on different subset of attribute sets.

In this chapter, fuzzy sets are used for the discretization by partitioning each attribute into three levels: “low” (L), “medium” (M), and “high” (H). Finer partitions may lead to better accuracy at the cost of higher computational load. The use of fuzzy sets has several advantages over the traditional “hard” discertizations, such as handling the overlapped clusters and linguistic representation of data [Pal 2004].

Triangular membership functions are used to define the fuzzy sets: *L*, *M* and *H*. There are three parameters $C_L$, $C_M$, and $C_H$ for each attribute which should be determined beforehand. They are considered as the centers of the three fuzzy sets. Here the center of fuzzy set *M* for a given attribute *a* is the average value of all the values occurring in the domain of *a*.

Assume $V_a$ is the domain of attribute *a*, then $C_M = \frac{\sum_{y \in V_a} y}{|V_a|}$, where $|*|$ is the cardinality of set $*$. $C_L$ and $C_M$ are computed as

$C_L = (C_M - Min_a)/2$,

$C_H = (Max_a - C_M)/2$,

where $Min_a = \min\{y \mid y \in V_a\}$ and $Max_a = \max\{y \mid y \in V_a\}$.

The membership functions are illustrated in Fig. 6.1.

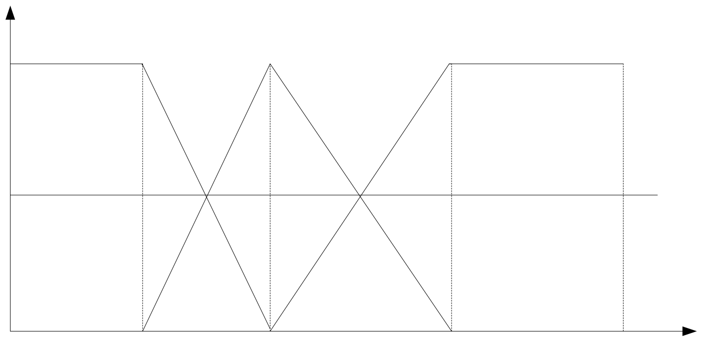

Fig. 6.1 Membership functions of *L*, *M* and *H* for attribute *a*

More formally, the membership functions for a given attribute *a* can be formulated as:

$$\mu_L(x) = \begin{cases} 1, & Min_a \le x \le C_L \\ \dfrac{C_M - x}{C_M - C_L}, & C_L < x \le C_M \\ 0, & x > C_M \end{cases},$$

$$\mu_M(x) = \begin{cases} 0, & x \le C_L \\ \dfrac{x - C_L}{C_M - C_L}, & C_L < x \le C_M \\ \dfrac{C_H - x}{C_H - C_M}, & C_M < x \le C_H \\ 1, & x > C_H \end{cases}$$

$$\mu_H(x) = \begin{cases} 0, & x \le C_M \\ \dfrac{x - C_M}{C_H - C_M}, & C_M < x \le C_H \\ 1, & x > C_H \end{cases},$$

1

where $\mu_*(x)$ is the membership value of case *x* to fuzzy set *.

0 5

## 6.3 Rough LVQ-Based Case Generation

In this section, we present the case generation method based rough LVQ learning process. Firstly, Section 6.3.1 briefly introduces the characteristics of LVQ. Section 6.3.2 explains the reason of incorporating the rough set-based FR and describes the rough LVQ-based CG algorithm.

### 6.3.1 Learning Vector Quantization

LVQ derives from the Self-organizing map (SOM) which is an unsupervised learning and robust to handle noisy and outlier data. The SOM can serve as a clustering tool of high-dimensional data. For classification problems, supervised learning LVQ should be superior to SOM since the information of classification results is incorporated to guide the learning process. LVQ is more robust to redundant features and cases, and more insensitive to the learning rate. As Kohonen pointed out in [Koho 1988], LVQ in stead of SOM should be used in decision and classification processes. This is the reason that LVQ is applied in case selection for building compact case base for CBR classifiers.

The basic idea of LVQ (see Fig. 6.2) is the same as that of SOM, which is simple yet effective. It defines a mapping from high-dimensional input data space onto a regular two-dimensional array of nodes called competitive layer. Every node *i* of the competitive layer is associated with an *m*-dimensional vector $v_i = [v_{i1}, v_{i2}, \ldots, v_{im}]$, where *m* denotes the dimension of the cases called reference vectors. The basic assumption here is that the nodes near to the same input vector should locate near to each other. Given an input vector, the most similar node in the competitive layer can be found as the winning node. Other nearby nodes for the input vector can be also found through similarity computation. Based on the mentioned assumption, the winning node and those nearby nodes should locate near to the input vector. The class information is also incorporated in the learning process. At each learning step, if the winning node and those nearby nodes are in the same class of input vector, the distances among these nodes are reduced; otherwise, these nodes are kept intact. This is different from the unsupervised learning process of SOM, where the winning node and those in its neighbourhood will move towards each other even they are not in the same class. The amount of decrease in distance is determined by the given learning rate. As a result, after the learning with the reference vectors, LVQ converges to a stable structure and the final weight vectors are the cluster centres. These weight vectors are considered as the generated prototypes which can represent the entire case base.

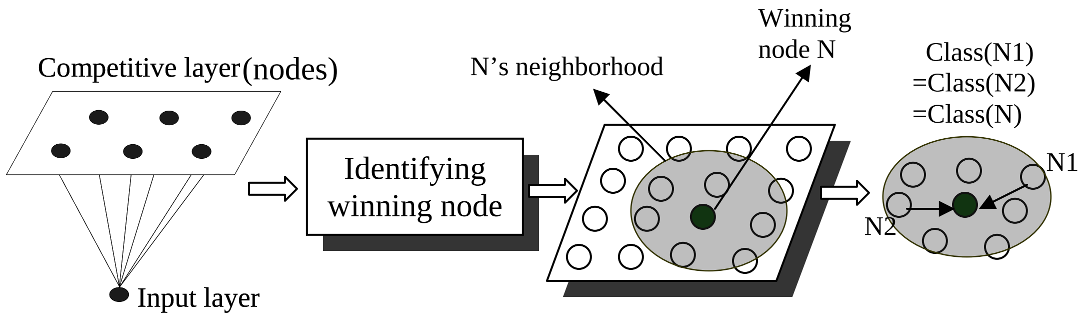

Fig. 6.2 Outline of Learning Vector Quantization (LVQ)

## 6.3.2 Rough LVQ Algorithm

Although LVQ has similar advantages of SOM, such as the robustness with noise and missing information, it does not mean that the data pre-processing is not required before the learning process. Since the basic assumption of LVQ is that similar feature values should lead to similar classification results, the similarity computation is critical in the learning process. Feature selection is one of the most important preparations for LVQ which can achieve better clustering and similarity computation results.

Different subset of features will result different data distribution and clusters. Take the Iris data [Hett 1998] for example. Fig 6.3 and Fig. 6.4 shows the two dimensional Iris data on two different subset of features: {PW, PL} and {SW, SL}.

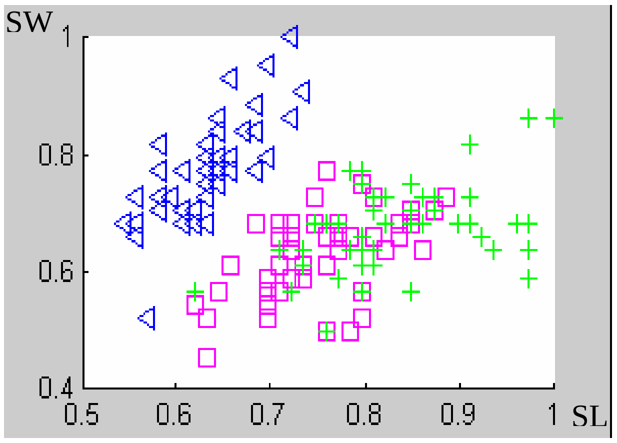

Fig. 6.3 Iris data on SL and SW

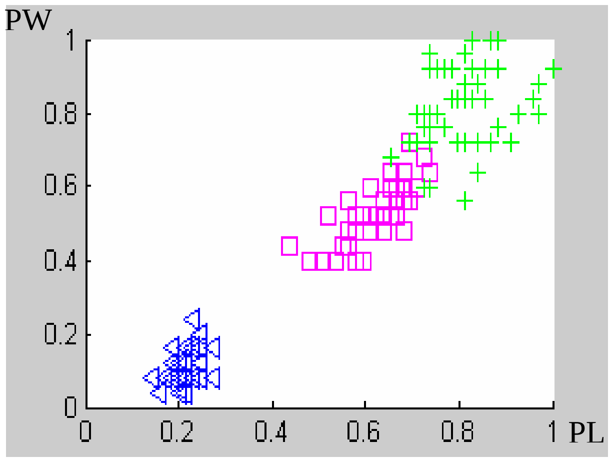

Fig. 6.4 Iris data on PL and PW

Based on the two subsets of features, LVQ is applied to learn three prototypes for the Iris data. The generated representative cases are shown in Tables 6.1-6.2 as follows:

Table 6.1 Prototypes extracted using PL and PW

| Prototypes | SL | SW | PL | PW | Class label |
|---|---|---|---|---|---|
| P1 | 0.619 | 0.777 | 0.224 | 0.099 | 1 |
| P2 | 0.685 | 0.613 | 0.589 | 0.528 | 2 |
| P3 | 0.766 | 0.587 | 0.737 | 0.779 | 3 |
| Classification accuracy using P1, P2, and P3: 0.98 | | | | | |

Table 6.2 Prototypes extracted using SL and SW

| Prototypes | SL | SW | PL | PW | Class label |
|---|---|---|---|---|---|
| P1 | 0.649 | 0.842 | 0.211 | 0.094 | 1 |
| P2 | 0.712 | 0.550 | 0.572 | 0.212 | 2 |
| P3 | 0.980 | 0.840 | 1.096 | 1.566 | 3 |
| Classification accuracy using P1, P2, and P3: 0.80 | | | | | |

It shows that different subset of attributes can affect the LVQ learning process and different prototypes are generated. According to the classification accuracy, the feature set of {PL, PW} is better than {SL, SW}.

In this chapter, the feature selection is addressed using the approximate reduct-based FR method which developed in Chapter 3. LVQ is then applied to generate representative cases for the entire case base. Here the learning rate $\alpha$ is given in advance, and only the distance between the winning node and the given input vector is updated in each learning step. The number of weight vectors is determined as the number of classes in the given case base. The learning process is ended with a fixed number of iterations *T*, say, 5000 in this chapter. Assume the given case base has *n* cases which represented by *m* features, and there are *c* classes. *R* is the approximate reduct computed by the feature selection process. The LVQ algorithm is given as follows:

**LVQ-based Case Generation Algorithm**

Step 1 Initialize *c* weight vectors $[v_1, v_2, \ldots, v_c]$ by randomly selecting one case from each class.

Step 2 Generate prototypes through LVQ.

```
    $t \leftarrow 1$;
    While ($t \leq T$)
        for $k$ = 1 to $n$
            $x \in U, x_k \leftarrow x, U \leftarrow U - \{x_k\}$;
        1. Compute the distances $D = \{\|x_k - v_{i,t-1}\|_R : 1 \le i \le c\}$;
        2. Select $v_{win,t-1} = \arg\{v_{i,t-1} : \|x_k - v_{i,t-1}\|_R = \min\{d \in D\}\}$;
        3. If $Class(v_{win,t-1}) = Class(x_k)$
               Update $v_{win,t} = v_{i,t-1} + \alpha(x_k - v_{win,t-1})$;

    4. Output $V = [v_{1,T-1}, v_{2,T-1}, ..., v_{c,T-1}]$.
```

The output vectors are not the data points in the given case base, but modified during the learning process based on the provided information by the data. They are considered to be the generated prototypes which represent the entire case base. Each prototype can be used to describe the corresponding class and regarded as the cluster center.

## 6.4 Experimental Results

To illustrate the effectiveness of the developed rough LVQ case selection method, we describe here some results on three real life data from UCI Machine Learning Repository [Hett 1998]. These databases are: Iris data, Glass data, and Pima data, whose characteristics are listed in Table 6.3. In all the experiments, 80% cases in each database are randomly selected for training and the remaining 20% cases are used for testing.

Table 6.3 The characteristics of three UCI databases

| Data set | Number of cases | Number of features | Category of features |
|---|---|---|---|
| Iris | 150 | 4 | Numerical |
| Glass | 214 | 10 | Numerical |
| Pima | 768 | 8 | Numerical |

In this section, four indices are used to evaluate the rough LVQ case generation method. The *classification accuracy* is one of the important factors to be considered for building classifiers. On the other hand, the efficiency of CBR classifiers in terms of *case retrieval time* should not be neglected. The *storage space* and clustering performance (in terms of *intra-similarity* and *inter-similarity*) are also tested in this section. Based on these evaluation indices, comparisons are made between our developed method and others such as basic SOM, basic LVQ and Random case selection methods.

As mentioned in Section 6.3, the rough set-based FR is firstly used to find the approximate reduct of the given case bases. In the experiments of this section, the parameter $\beta$ is determined during the testing through populating the points in the interval [0.5, 1]. Initially, $\beta$ is set to be 0.5. In each step, the $\beta$ value increase at a constant rate 0.01 and this value is used in the feature selection process and being tested. The steps stop when $\beta$ attains 1. The $\beta$ value which can achieve the highest classification accuracy is selected as the suitable $\beta$. Based on the generated subset of features, the LVQ learning is then applied for extracting representative cases as the prototypes of the entire case base. The learning rates for the three data sets are: $\alpha$ = 0.8 (Iris data), $\alpha$ = 0.8 (Glass data) and $\alpha$ = 0.5 (Pima data). In the following sections, each evaluation index is tested to show the effectiveness of the rough LVQ case generation approach.

### 6.4.1 Classification Accuracy

In this section, the results of classification accuracy for the three databases and four CG methods are demonstrated and analyzed. The used accuracies here are defined as:

$$Accuracy_{Test} = \frac{|\{x, x\ can\ be\ correctly\ classfied, x \in Testdata\}|}{|Testdata|},$$

$$Accuracy_{All} = \frac{|\{x, x\ can\ be\ correctly\ classfied, x \in Entiredata\}|}{|Entiredata|},$$

where $|*|$ is the cardinality of set $*$; *Testdata* is the set of cases for testing; *Entiredata* is the set of cases in the whole data set. To be more specifically, "*x* can be correctly classified" means that *x* can be correctly classified by the extracted prototypes.

If the training cases are used for classify the testing cases, the classification accuracies on the three databases are: 0.980 (Iris), 0.977 (Glass), 0.662 (Pima). These accuracy values are called the original classification accuracies. The experimental results of using the generated prototypes are demonstrated in Table 6.4. It is observed that after the case generation, the original accuracies are preserved and even improved. The rough LVQ method can achieve the highest classification accuracy in most of the testing. The basic LVQ method performs better than the other methods: Random and SOM.

Table 6.4 Comparisons of classification accuracy
using different case generation methods

| | Iris data | | Glass data | | Pima data | |
|---|---|---|---|---|---|---|
| Methods | $Accuracy_{Test}$ | $Accuracy_{All}$ | $Accuracy_{Test}$ | $Accuracy_{All}$ | $Accuracy_{Test}$ | $Accuracy_{All}$ |
| Random | 0.760 | 0.746 | 0.860 | 0.864 | 0.597 | 0.660 |
| SOM | 0.920 | 0.953 | 0.930 | 0.925 | 0.688 | 0.730 |
| LVQ | 0.980 | 0.953 | 0.930 | **0.935** | 0.708 | **0.743** |
| Rough LVQ | **1.000** | **0.960** | **0.930** | **0.935** | **0.714** | 0.740 |

## 6.4.2 Reduced Storage Space of Rough LVQ-Based Method

Due to both the feature selection and case selection processes, the storage space with respect to the features and cases is reduced substantially. Subsequently, the average case retrieval time will decrease. These results are shown in Table 6.5, where

$$Reduce\ features = (1 - \frac{|Selected\ \ features|}{|Original\ \ features|}) \times 100\%,$$

$$Reduced\ cases = (1 - \frac{|\Pr ototypes|}{|Entiredata|}) \times 100\%,$$

$$Saved\ time\ of\ case\ retrieval = (t_{train} - t_p),$$

where $|*|$ is the number of elements in set $*$; *Selected features* is the set of features that are selected by the rough set-based method; *Original features* is the set of features in the original database; P*rototypes* is the set of extracted representative cases; $t_{train}$ is the case retrieval time to classify the testing cases using the training cases; $t_p$ is the case retrieval time to classify the testing cases using the extracted prototypes; *Testdata* is the same as that in Section 6.4.1. The unit of time is second.

Table 6.5 Reduced storage and saved case retrieval time

| Data set | Reduced features | Reduced cases | Saved time of Case retrieval |
|---|---|---|---|
| Iris | 50% | 97.0% | 0.600 sec |
| Glass | 60% | 98.8% | 0.989 sec |
| Pima | 50% | 99.6% | 0.924 sec |

From Table 6.5, the storage requirements of features and cases are reduced dramatically. For example, the percentage of reduced features is 60% for Glass data, and the percentage of reduced cases is 99.6% for Pima data. The case retrieval time also decreases because that there are much fewer features and cases after applying the rough LVQ-based case selection method.

### 6.4.3 Intra-similarity and Inter-similarity

Intra-similarity and inter-similarity are two important indices to reflect the clustering performance. They are used in this section to prove that the developed rough LVQ-based approach can achieve better clustering than using random selected prototypes.

Since the similarity between two cases is inverse proportional to the distance between them, we use inter-distance and intra-distance to describe the inter-similarity and intra-similarity. These distances can be directly computed based on the numerical feature values. Assume there are $K$ classes for a given case base, $C_1$, $C_2$, …, $C_K$. The intra-distance and inter-distance of the case base are defined as:

$$\textit{Intra-Distance} = \sum\nolimits_{x,y \in C_i} d(x, y) \ ,$$

$$\textit{Inter-Distance} = \sum\nolimits_{x \in C_i, y \in C_j} d(x, y), i, j = 1, 2, ..., K, i \neq j \ .$$

$$\textit{Ratio} = \textit{Inter-Distance} / \textit{Intra-distance}.$$

The lower the intra-distance and the higher the inter-distance, the better is the clustering performance. Therefore, it is obvious that the higher the ration between the inter-distance and the intra-distance, the better is the clustering performance.

The results are shown in Table 6.6. Rough LVQ method demonstrates higher Ratio values and therefore achieves better clustering result.

Table 6.6 Inter-distance and inter-distance:
Comparisons between the Random and Rough LVQ methods

| Data set | Methods | Inter-Distance | Intra-Distance | Ratio |
|---|---|---|---|---|
| Iris | Random | 1284.52 | 102.13 | 12.577 |
| | Rough LVQ | 1155.39 | 51.99 | 22.223 |
| Glass | Random | 8640.20 | 4567.84 | 1.892 |
| | Rough LVQ | 7847.37 | 3238.99 | 2.423 |
| Pima | Random | 56462.83 | 54529.05 | 1.035 |
| | Rough LVQ | 28011.95 | 25163.45 | 1.113 |

## 6.5 Summary

In this chapter, a rough LVQ approach is developed to address the case generation for building compact and competent CBR classifiers. Firstly, the rough set-based FR method is used to select features for LVQ learning. As mentioned in Chapter 3, this method is built on the concept of approximate reduct instead of exact reduct. The approximate reduct can be found quickly and effectively. LVQ is then used to extract the prototypes to represent the entire case base. These prototypes are not the data points in the original case base, but are modified during the LVQ learning process. They are considered as the most representative cases for the given case base, and used to classify the unseen cases. Through the experimental results, using much fewer features (e.g., 40% of the original features for Glass data), the classification accuracies for the three real life data are higher using our method than those using methods of Random, basic SOM and LVQ. The case retrieval time for predicting class labels of unseen cases is also reduced. Furthermore, higher intra-similarity and lower inter-similarity are achieved using the rough LVQ approach than that using the random method.

# Chapter 7 Fuzzy Integral-Based Case Base Competence Model

In the previous chapters, we have presented our developed techniques for the tasks of FR, learning similarity measures, CS and CG to extract case knowledge for building both compact and competent CBR systems. In this chapter, we build a case base competence model to evaluate the coverage of a given case base or knowledge base, where the fuzzy integral technique is applied for modeling the competence of a given CBR system.

Case base competence has been brought sharply into focus in the area of case base maintenance (CBM) since many CBM policies are directly linked with the heuristics of measuring case base competence to guide the maintenance procedures [Smyt 1995, 1998a, 1998b][Yang 2001][Leak 2000]. However, most of the current competence heuristics only provide coarse-grained estimates of competence. For example, Smyth et al. [Smyt 1999a, 1999b, 2000a, 2000b, 2000c, 2001a, 2001b] employed a case deletion policy guided by a category-based competence model, where the cases are classified to only four basic competence categories. Zhu and Yang [Zhu 1999] provided a case addition policy based on the concept of case neighborhood, which is a coarse approximation of case coverage. Modeling case base competence becomes a crucial issue in the field of CBM.

Smyth and McKenna proposed a competence model based on the concept of competence group, which is defined in such a way that there are no overlaps among case coverage in different competence groups. The group size and density have been considered in the definition of the group coverage. Then the overall case base competence can be computed by simply summing up the coverage of each group. However, the distribution of each group is not taken into account in this model. More specifically, it always assumes that the distribution of cases in each group is uniform, which leads sometimes to over or under estimation of case base competence.

This problem is addressed by adopting a fuzzy integral based competence model to compute the competence of a given CBR system more accurately. Consider a competence group. We first repartition it to ensure that the distribution of cases in each newly obtained group is nearly uniform. Since there are overlaps among the coverage of different new groups, fuzzy measures (non-additive set functions) can be used to describe these overlaps because of their non-additive characteristics. Fuzzy integrals are the corresponding integrals with respect to these fuzzy measures. As the most important tools of aggregations in information fusion, fuzzy integrals are appropriate here for computing the overall case base competence of a CBR system. A kind of fuzzy measure, $\lambda$-fuzzy measure, and a corresponding fuzzy integral, Choquet integral [Wang 1992][Pap 1996][Wang 2000a], are adopted in this approach.

For the convenience of readers, we first give a brief description of fuzzy measures and fuzzy integrals.

## 7.1 Concepts of Fuzzy Measures and Fuzzy Integrals

The traditional tool of aggregation for information fusion is the weighted average method, which is essentially a linear integral. It is based on the assumption that the information sources involved are non-interactive and, hence, their weighted effects are viewed as additive. This assumption is not realistic in many applications. To describe the interaction among various information sources in such cases, a new mathematical tool, namely, fuzzy measures or non-additive set functions, can be used instead. In other words, a nonlinear integral, such as the Choquet integral with respect to the non-additive set functions, can be used instead of the classical weighted average for information fusion.

More formally, fuzzy measures and fuzzy integrals are defined as follows:

**Definition 7.1 (Fuzzy measure)**

Let $X$ be a nonempty set and P($X$) be the power set of $X$. We use the symbol $\mu'$ to denote a non-negative set function defined on P(X) with the properties $\mu'(\Phi)=0$. If $\mu'(X)=1$, $\mu'$ is said to be regular. It is a generalization of classic measure. When $X$ is finite, $\mu'$ is usually called a fuzzy measure if it satisfies monotonicity, i.e.,

$$A \subseteq B \Rightarrow \mu'(A) \le \mu'(B) \text{ for } A, B \in P(X). \tag{7.1}$$

For a non-negative set function $\mu'$, there are some associated concepts:

**Definition 7.2 (Additive, super-additive, sub-additive set function)**

For $A$, $B \in P(X)$, $\mu'$ is said to be additive if

$$\mu'(A \cup B) = \mu'(A) + \mu'(B), \tag{7.2}$$

$\mu'$ is said to be sub-additive if

$$\mu'(A \cup B) \le \mu'(A) + \mu'(B), \tag{7.3}$$

$\mu'$ is said to be super-additive if

$$\mu'(A \cup B) \le \mu'(A) + \mu'(B). \tag{7.4}$$

If we regard $\mu'(A)$ and $\mu'(B)$ as the importance of subsets $A$ and $B$, respectively, then the additivity of the set function means there is no interaction between $A$ and $B$, that is, the joint importance of $A$ and $B$ is just the sum of their respective importance. Super-additivity means the joint importance of $A$ and $B$ is greater than or equal to the sum of their respective importance, which indicates that the two sets are enhancing each other. Sub-additivity means the joint importance of the two sets $A$ and $B$ is less than or equal to the sum of their respective importance, which indicates that the two sets are resisting each other.

Due to non-additivity of the fuzzy measures, some new types of integrals (known as fuzzy integrals) such as Choquet integral, Sugeno integral, and N-integral, are used. Here we only give the definition of Choquet integral which is used in this section:

**Definition 7.3 (Choquet integral)**

Let $X = \{x_1, x_2, \ldots, x_n\}$, $\mu'$ be a fuzzy measure defined on the power set of $X$, and $f$ be a function from $X$ to [0, 1]. The Choquet integral of $f$ with respect to $\mu'$ is defined by

$$(C)\int f \, d\mu' = \sum_{i=1}^{n} (f(x_i) - f(x_{i-1}))\mu'(A_i) \tag{7.5}$$

where we assume without loss of generality that $0 = f(x_0) \leq f(x_1) \leq \cdots \leq f(x_n)$ and $A_i = \{x_i, x_{i+1}, \cdots, x_n\}$.

Now we give an example [Wang 1999] to illustrate how fuzzy measure and fuzzy integral describe the interactions of different objects.

*Example*:

Let there be three workers $a$, $b$, and $c$ working for $f(a) = 10$, $f(b) = 15$, and $f(c) = 7$ days respectively to manufacture a kind of products. Without a manager, they begin to work from the same day. Their efficiencies of working alone are 5, 6, and 8 products per day respectively. Their joint efficiencies are not the simple sum of the corresponding efficiencies given above, but are listed as follows:

| Workers | Products/ day |
|---|---|
| $\{a, b\}$ | 14 |
| $\{a, c\}$ | 7 |
| $\{b, c\}$ | 16 |
| $\{a, b, c\}$ | 18 |

These efficiencies can be regarded as a fuzzy measure, $\mu'$, defined on the power set of $X = \{a,b,c\}$ with $\mu'(\phi) = 0$ (the meaning is that there is no product if no worker is there).

Here inequality $\mu'(\{a,b\}) > \mu'(\{a\}) + \mu'(\{b\})$ means that *a* and *b* have good cooperation, while inequality $\mu'(\{a,c\}) < \mu'(\{a\}) + \mu'(\{c\})$ means that *a* and *c* have bad relationship and are not suitable for working together. Here $\mu'$ can be considered as an efficiency measure.

In such a simple manner, during the first 7 days, all workers work together with efficiency $\mu'(\{a,b,c\})$, and the number of products is $f(c) \cdot \mu'(\{a,b,c\}) = 7 \times 18 = 126$, during the next $f(a) - f(c)$ days, workers a and b work together with efficiency $\mu'(\{a,b\})$, and the number of products is $[f(a) - f(c)] \cdot \mu'(\{a,b\}) = 3 \times 14 = 42$; during the last $f(b) - f(a)$ days, only b works with efficiency $\mu'(\{b\})$, and the number of products is $[f(b) - f(a)] \cdot \mu'(\{b\}) = 5 \times 6 = 30$. Function $f$ defined on $X = \{a,b,c\}$ is called an information function. Thus, the value of the Choquet integral of *f* with respect to $\mu'$,

$$(C) = \int f d\mu' = f(c) \cdot \mu'(\{a,b,c\}) + [f(a) - f(c)] \cdot \mu'(\{a,b\}) + [f(b) - f(a)] \cdot \mu'(b) = 198$$

is just the total number of products manufactured by these workers during these days.

Note that the meaning of fuzzy measures and fuzzy integrals is problem dependent. In this section, they are used to describe the coverage contributions of cases in a case base.

Before the fuzzy integral based competence model is explained, the closely related (non-fuzzy) competence model of Smyth [Smyt 1998a] is described for convenience, along with its limitations.

## 7.2 Case Base Competence

Smyth and McKenna [Smyt 1995, 1998a, 1998b] explained the concept of case-base competence, and subsequently different concepts such as *coverage* and *reachability* for measuring the problem solving ability of case bases were developed. Some statistical properties of a case base, e.g., the size and density of cases, are used as input parameters for modeling the case-base competence. For convenience, their model is called the S-K model in this research, which will be briefly reviewed in Section 7.2.1.

### 7.2.1 The S-K Competence Model

The main idea of the S-K model is to introduce the concept of competence group as the fundamental computing unit of case base competence. It is defined in such a way that different groups have no interaction (overlap) with each other [Smyt 1998b]. Based on this concept, the competence of each group is computed by considering the size and group density, which is given as

**Definition 7.4 (Group competence)**

The competence of a group of cases ($G$) (i.e., *group coverage of G*) depends on the number of cases in the group and its density. This is defined as

$$GroupCoverage(G) = 1 + |G| \cdot (1 - GroupDensity(G)) \,. \tag{7.6}$$

Here *GroupDensity* is defined as the average *CaseDensity* of the group, i.e.,

$$GroupDensity(G) = \sum_{e \in G} CaseDensity(e, G) / |G| , \tag{7.7}$$

where

$$CaseDensity(e, G) = \sum_{e^* \in G-\{e\}} SM(e, e^*)/(|G|-1), \tag{7.8}$$

and |*G*| is the size of the competence group *G*, i.e. the number of cases in the group *G*.

Different ways of computing the *Similarity* between two cases *e* and *e** depend on the problems at hand.

For a given case base *CB*, with competence groups $\{G_1, G_2, ..., G_n\}$, the total coverage or the case-base competence is defined by Equation (9) as

$$Coverage(CB) = \sum_{G_i \in G} GroupCoverage(G_i) \tag{7.9}$$

## 7.2.2 The Problem of the S-K Competence Model

In this Section, we analyze the problems arising from the S-K competence model, which includes the ignorance of the case distribution and overlaps among group competence.

From Equation (7.9), it is seen that the definition of case-base competence only took the concepts of group size and group density into account. However, the distribution of cases in a case base is also an important factor which influences the case-base competence. For example, consider Fig. 7.1 where the cases in (b) are uniformly distributed, whereas those in (a) & (c) are not. It is appropriate and necessary to incorporate this while computing the case-base competence of a case base.

Suppose that in some problem domain, we have a group of non-uniformly distributed cases as depicted in Fig. 7.1(a), it can be shown that the S-K model is not a good predictor of the group competence because this model assumes that the cases are distributed uniformly such as those shown in Fig. 7.1(b). Assuming that $Size(G) = Size(G')$, i.e. $|G| = |G'|$, and $GroupDensity(G) = GroupDensity(G')$. Then, from Equation (7.9), we have

$GroupCoverage(G) = GroupCoverage(G')$, where $G$ is an arbitrary competence group in a case-base, so similar results can be obtained between two case-bases, in which one has its cases non-uniformly distributed and the other uniformly distributed.

However, from Figs 7.1(a) and 7.1(b), it is obvious that the coverage of the two competence groups cannot possibly be the same. There are coverage holes, i.e., the regions that cannot be covered by the case base, in Fig. 7.1(a) compared with that of Fig. 7.1(b). If we calculate the competence of the groups in Fig. 7.1(a) using the S-K model, then the actual competence will be over-exaggerated. It is because, the S-K model only considers the group density, but ignores their distribution. There are possibly many ways of case distributions, therefore a more accurate way of modeling of case-base competence is required.

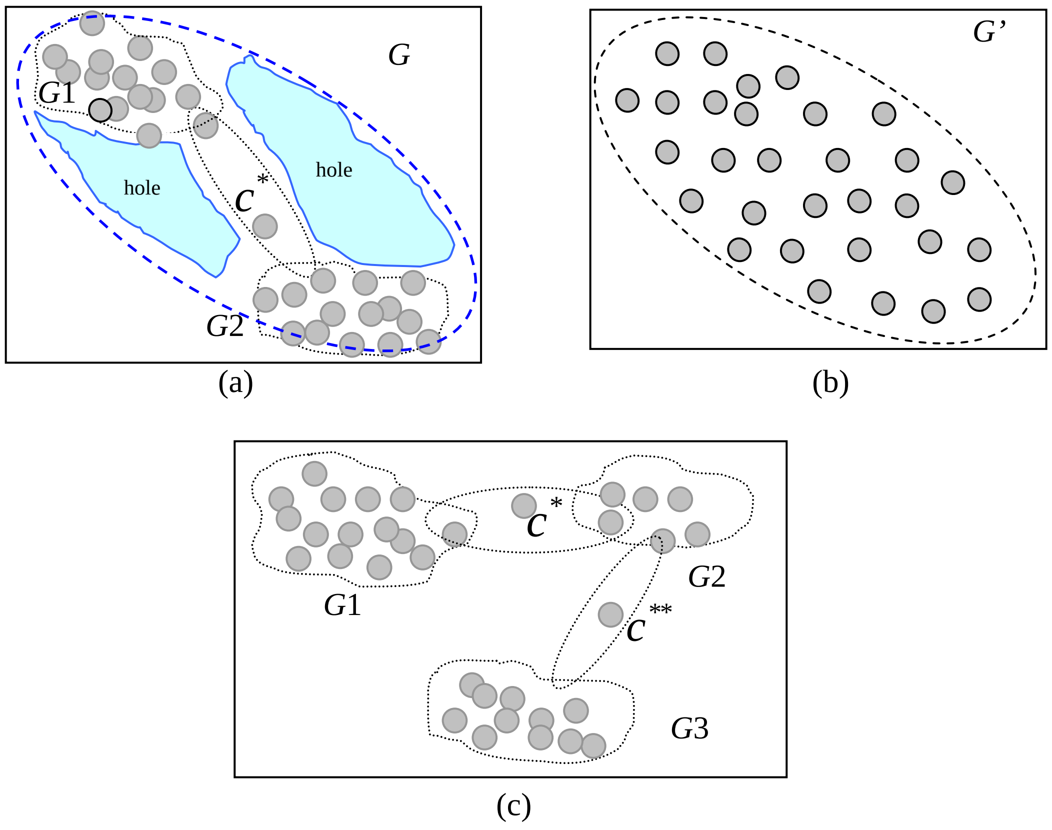

Fig. 7.1 Examples of uniform and non-uniform case distributions

Moreover, this model assumes that there is no overlap among different competence groups of cases (e.g., features interaction [Wang 2000b] is a common cause of overlaps). Therefore, by simply taking the sum of group competences as the overall case base competence, without considering the overlapping effects, the resulting group competence may be over or under-exaggerated. This group overlap problem has been tackled by Shiu et al. [Shiu 2001b] using fuzzy integral. Details are explained in the following.

## 7.3 Fuzzy Integral Based Competence Model

Consider both Figs 7.1(a) and (c), where we can easily see that cases like $c^{*}$ and $c^{**}$ play an important role in affecting the overall competence distribution in the group. Therefore, it is important to detect such cases (which are called weak-links in the following discussion) for possible identification of smaller competence groups, such as $G1$, $G2$, $G3$ in Fig. 7.1(c), those are having more evenly distributed cases. These smaller groups' competence can then be computed using Equations (7.6)-(7.8). It is worth noting that, the competence of weak links can be considered to be their respective individual coverage, which reflects the relation among the several new groups. A new way of computing the group competence based on this principle is described as follows.

### 7.3.1 Competence Error

In general, competence groups, such as $G1$ and $G2$ in Fig. 7.1(a), are not necessarily having the same strictly uniform distribution, and the weak link case $c^{*}$ is not necessarily a pivotal case (a case that cannot be solved by any other cases). To deal with this situation, $GroupDensity(G1)$ (which is assumed to be equal to $GroupDensity(G2)$) can be replaced by the average group density of group $G1$ and $G2$, which can be denoted by $\overline{GroupDensity(G_i)},\ i \in \{1,\ 2\}$. Let $[\overline{GroupDensity(G_i)} - GroupDensity(G)]$ be denoted by $\Delta GroupDensity$. A concept, called quasi-uniform distribution, can be used to describe the case base distributions which are close to uniform distribution. As mentioned, the other assumption that $c^{*}$ is a pivotal case in the example is not necessarily true in many cases. To address this problem, just consider the individual competence of $c^*$ as its relative coverage, which is defined as

**Definition 7.5 (Relative coverage)**

The relative coverage of a case *e* is denoted by

$$RelativeCoverage(e) = \sum_{e' \in CoverageSet(e)} \frac{1}{|ReachabilitySet(e')|}. \tag{7.10}$$

Then define,

$$\begin{aligned} &CompetenceError(c^*) \\ &= |G \, \Delta GroupDensity - \overline{GroupDensity(G_i)} - RelativeCoverage(c^*) \\ &\geq |G| \Delta GroupDensity - (RelativeCoverage(c^*) + 1) \end{aligned} \tag{7.11}$$

Since $RelativeCoverage(c^*)$ is small, we can see that it is $\Delta GroupDensity$ which mainly leads to the competence error.

## 7.3.2 Weak Links Detection

In order to tackle the problem of non-uniformly distributed cases, it is necessary, as mentioned before, to identify first the weak links in each competence group. The definition of weak link and several other concepts which are more directly related to the competence of the group in question are stated below:

**Definition 7.6 (weak link)**

Let $G = \{G_1, G_2, \cdots, G_n\}$ be a set of competence groups in a case base *CB*. $c^* \in G$ is called a *weak link* if $CompetenceError\,(c^*) \geq \alpha$,

where $\alpha$ is a parameter that is defined by the user depending on the requirement.

If $\exists c^* \in G$, $c^*$ is a *weak-link*, then the competence group $G$ is called a *non-uniform distributed* competence group. Otherwise, if $\forall e \in G$, *CompetenceError* $(e) \le \alpha$ , $G$ is called *a quasi-uniform distributed* competence group.

Here a recursive method is explained to detect the weak links in a given competence group *G*, as follows:

***Weak-link Detection Algorithm*:**

```
1. $W\text{-}SET \leftarrow \{\ \}$, $G\text{-}SET \leftarrow \{\ \}$, $i = |G|$;

2. If ( $i \ne 0$ )
 {Consider each given competence group G in the S-K competence model, compute
   CompetenceError(e), $\forall e \in G$;  $i = i - 1$;}

3. If there is no weak link, add G to G–SET, end;

4. If there is a weak link $c^*$, identify the competence groups $G_1, G_2, \cdots, G_n, (n \ge 1)$ in G–
   $\{c^*\}$ using the S-K competence model; add $c^*$ to the set of weak-links W-SET.

5. For $(1 \le i \le n)$ { $G \leftarrow G_i$ ; repeat Steps 1 to 4}.
```

Thus, we can obtain the set of weak links *W–SET* in a given competence group *G* and the set of new competence groups *G–SET*.

## 7.3.3 Overall Coverage of Competence Group Using Fuzzy Integral

After detecting the weak links in a competence group *G* and cutting them off, let *n* new competence groups $G_1, G_2, \cdots, G_n (n \ge 1)$ be produced. According to the definition of a weak link, each newly produced group is sure to be quasi-uniformly distributed. The next task is to compute the overall coverage or competence of *G*. In the example described in

Fig. 7.1(a), the overall competence of $G$ can be simply calculated by the sum of the competence of $G_i$ $(1 \le i \le n)$ and the relative coverage of $c^*$, but this method is not representative. There could be more complicated situations, as illustrated in Fig. 7.1(c), where it is difficult to clearly identify the contribution of each weak link. For example, in Fig. 7.1(c), $c^*$ has much more influence on the coverage of $G$ than $c^{**}$ has, which reflects different relations among new competence groups. Therefore, a powerful tool, called fuzzy integral (or non-linear integral) with respect to a fuzzy measure (a non-additive set function), is applied to describe this complex relationship.

*Determining the $\lambda$-fuzzy measure $\mu'$:*

When the fuzzy integral is used to compute the overall coverage of the original competence group $G$, it is necessary to determine first the importance measure $\mu'$ of the $n$ small competence groups $G_i$ $(1 \le i \le n)$ both individually and in all possible combinations, the total number being $(2^n - 1)$. For cases in Fig. 7.1(c), there will be seven values of such measure, e.g.,

$$\mu'(G_1), \mu'(G_2), \mu'(G_3), \mu'(G_1 \cup G_2), \mu'(G_1 \cup G_3), \mu'(G_2 \cup G_3), \mu'(G_1 \cup G_2 \cup G_3),$$

where $\mu'$ values of the unions of small competence groups can be computed by the $\lambda$-*fuzzy* measure [Wang 1992], which takes the following form:

$$\mu'(A \cup B) = \mu'(A) + \mu'(B) + \lambda\mu'(A) \cdot \mu'(B), \quad \lambda \in (-1, \infty) \tag{7.12}$$

If $\lambda \le 0$, $\mu'$ is a sub-additive measure; if $\lambda \ge 0$, $\mu'$ is a super-additive measure; if and only if $\lambda = 0$, $\mu'$ is additive. So the focus of determining the $\lambda$-*fuzzy* measure $\mu'$ falls on the determination of the importance of each single group and $\lambda$. Note that $\lambda \ge 0$ in this example.

Given $\mu'(G_i) = 1\,(1 \le i \le n)$, here the main problem in computing $\mu'$ is therefore to determine the parameter $\lambda$. It is obvious that the properties of the weak links between two groups are important for determining $\lambda$. In this model, coverage of a group refers to the area of the target problem space covered by the group. In this sense, the value of $\lambda$ is closely related to the coverage of weak links and the density of their coverage sets.

Consider two arbitrary new groups $G_i$ and $G_j$. Let the W-SET between them be $C^* = \{c_1^*, \cdots, c_h^*\}$. The $Coverage(C^*)$ and $Density(C^*)$ are defined as follows:

$$Coverage(C^*) = \sum_{i=1}^{h} \mathrm{Re}\,lativeCoverage(c_i^*)\,, \tag{7.13}$$

$$Density(C^*) = \sum_{i=1}^{h} GroupDensity(Cov(c_i^*))/h\,, \tag{7.14}$$

where $Cov(c_i^*)$ is the coverage set of the *i*th weak link $c_i^*$ between $G_i$ and $G_j$.

The coverage contribution of $G_i \cup G_j$ must be directly proportional to $Coverage(C^*)$ and inversely proportional to $Density(C^*)$. With these assumptions, the parameter $\lambda$ is defined as

$$\lambda = Coverage(C^*) \cdot (1 - Density(C^*))\,, \tag{7.15}$$

The $\lambda$-*fuzzy* measure $\mu'$ of $(G_i \cup G_j)$ can then be determined with Equation (7.12).

*Using the Choquet integral to compute competence (coverage)*:

Due to the non-additivity property of the set function $\mu'$, some new types of integrals (known as non-linear integrals) are used to compute the overall coverage of the original competence group $G$ based on the $\mu'$ measures of the $n$ constituting competence groups and their unions. A common type of nonlinear integrals with respect to non-negative monotone set functions is the Choquet integral [Pap 1996]. Its use in computing the competence of the group *G* is described below.

Let the competence group $G = \{G_1, G_2, \ldots, G_n\}$ be finite, where $G_1, G_2, \cdots, G_n$ are the new small competence groups as defined earlier. Let $G^* = G \cup C^*$, where $C^*$ represents a weak link. Let $f_i = GroupCoverage(G_i)$, and the importance measure $\mu'$ satisfy:

$\mu'(G_i) = 1\, (1 \le i \le n)$;

$\mu'(A \cup B) = \mu'(A) + \mu'(B) + \lambda \cdot \mu'(A) \cdot \mu'(B)\, (\lambda \ge 0)$,

where $\lambda$ is determined by Equation (7.15).

The process of calculating the value of the Choquet integral is as follows:

(1) Rearrange $\{ f_1, f_2, \cdots, f_n \}$ into a non-decreasing order such that

$f_1^* \le f_2^* \le \cdots \le f_n^*$,

where $(f_1^*, f_2^*, \cdots, f_n^*)$ is a permutation of $(f_1, f_2, \cdots, f_n)$;

(2) Compute

$$Coverage(G) = \int f d\mu' = \sum_{j=1}^{n} [f_j^* - f_{j-1}^*] \cdot \mu'(\{G_j^*, G_{j+1}^*, \cdots, G_n^*\}),$$

where $f_0^* = 0$.

The value of the Choquet integral provides the coverage of the considered competence group *G*. For a case base with several competence groups, the sum of the individual group coverage gives the overall coverage of the case base.

## 7.4 Experiment Results

In this section, some empirical results are provided to demonstrate the effectiveness of the fuzzy integral method (Section 7.3) in closely matching the actual competence of a case base. At the same time the S-K competence model (Section 7.2) is shown not to be a good predictor when the case base is not uniformly distributed.

For this purpose, a small case base containing 120 cases, each of dimension two, is considered. Each case is chosen randomly so that the case base satisfies non-uniform distribution. During the investigation, 50 randomly chosen cases in the case base are used as unknown target problems, the remaining 70 cases are used to form the experimental case bases.

The success criterion used is a similarity threshold: if the system does not retrieve any cases within this threshold, a failure is announced. True competence is regarded as the number of successfully solved problems.

The experiment was repeated 100 different times. The results shown in Table 7.1 are the average values computed over these iterations. In the table, "Error_percentage" represents the relative error of coverage of a model with respect to the True model, and is defined as Error_percent = (Error_number / True_competence)×100%.

Table 7.1 Comparison of the three competence models

| Index | True | S-K Model | Fuzzy Integral Model |
|---|---|---|---|
| Density | - | 0.4 | 0.6 |
| Competence | 34.5 | 49.6 | 38.9 |
| Error_number | 0 | 15.1 | 4.4 |
| Error_percent | 0 | 43.8% | 12.8% |

As expected, the error_percentage of the fuzzy integral model is rather lower than that of the S-K competence model. When the number of cases increases, the former can strikingly reduce the competence error compared to the latter.

In this experiment, the case base considered has a non-uniform distribution, but in the situation of uniform distributed case bases, the fuzzy integral competence model can still be used. Because, if there is no weak-link, the competence computed by the fuzzy integral model will be the same as that obtained using the S-K competence model.

## 7.5 Summary

In this chapter, we built a case base competence model based on fuzzy integrals, which are able to describe the interaction among case coverage. We have presented firstly the concepts of fuzzy measures (or called non-additive set functions) and fuzzy integrals. Then the competence model proposed by Smyth and McKenna is briefly reviewed, which is based on the group density and the size of given case base. Finally, one common type of fuzzy integral, the Choquet integral, is used to model the case base competence. Different from the model of Smyth and McKenna, this developed competence model has taken into account the interaction among the competence groups. The experimental results show that the fuzzy integral based model can reflect the case base competence more accurately. This will help to evaluate the extracted case knowledge bases through FR and CS in previously chapters.

# Chapter 8

# Query Dispatching Policies in CCBR

In Chapter 7, we have built a fuzzy integral based case base competence model, which can handle the overlaps of case coverage. This chapter presents an application of this developed competence model for dispatching queries in the context of distributed or collaborative case-based reasoning (CCBR).

In a CCBR environment, multiple CBR systems are distributed in different places and each system can solve the problems independently. Here each CBR system is considered to be a problem-solving agent. An input query case could be compared with the old cases that are resided in the different CBR agents in the network. How to obtain the best solution effectively and efficiently from this distributed CBR network depends on a carefully designed query dispatching strategy.

In this chapter, we propose a group of competence-preserving query dispatching policies based on the fuzzy integral based competence model developed in Chapter 7. The coverage of each CBR agent in the network is measured and three strategies are proposed: To-Top policy, Strong-Strong policy and Best-Committee policy. The experimental result shows that our proposed policies are comparatively better than the existing ones developed by Plaza and Ontañón [Plaz 1997].

## 8.1 Introduction

Traditionally, intelligent systems are developed in a standalone and insolated manner. However, the recent growth of the World Wide Web and multi-agent systems triggers the need of designing intelligent systems in a distributed and collaborative manner. Being a successful intelligent system technology, CBR also has the need to develop its applications into a full fledged and distributed environment. Currently, there are two main approaches for selecting CBR agents, the first one is proposed by Prasad et al. [Pras 1996] based on the concept of task decomposition, and the second one is proposed by Plaza et al. [Plaz 1997] based on the random selection of agents. The use of task decomposition is only effective when the problem can be nicely decomposed into a set of sub-problems, and each sub-problem can be solved by an individual CBR agent. However if conflicts exist (e.g. the solutions from two sub-problems could not be integrated), additional heuristics from the users may be needed. Sometimes this way of collaboration may be even worse than a single and isolated system [Leak 2001].

Plaza et al. [Plaz 1997] proposed two modes of cooperation among CBR agents (i.e. Distributed Case-based Reasoning (DistCBR) and Collective Case-based Reasoning (ColCBR)). DistCBR means that a problem can be dispatched to any agent for solving, disregarding who generates the problem. ColCBR means that the owner of the problem tries to collect the useful cases and methods from other agents, and decide how to solve the problem. Three collaboration policies (i.e. Committee policy, Peer-Counsel Policy and Bounded-Counsel Policy) were developed for the DistCBR framework by Plaza and Ontañón [Plaz 2001]. However, these policies are all based on a random selection of the agents, which does not guarantee the quality of retrieved cases. Therefore, it will be very consuming to obtain the best solution among different CBR agents in a CCBR network. In 2003， Yang et al. [Yang 2003] built a service agent network (SANet) for call center automation, which integrates both human and software service agents in providing customer service in real time. For each service request, SANet selects the most appropriate agents according to the availability and capabilities of the agents in the network. The capability of a given agent is proportional to the quality of solution it provides, which can be described by the average number of successfully solved problems. In this research, we do not consider multi-agent systems which contain human agents.

Instead of having random selection of agents in [Plaz 2001], we propose our policies based on the concept of competence, which is defined as the range of problems that a particular agent (or case) could solve. Since the purpose of these policies is preserving the competence, they are comparatively better than most of the existing ones based on random agent selection.

The structure of this chapter is as follows: Section 8.2 reviews two methods of calculating the cases competence. The first one is proposed by Smyth and McKenna [Smyt 1998] while the second one is proposed by our research group [Shiu 2001b]. The competence computation and ranking policies of CBR agents are given in Section 8.3. In Section 8.4, three policies for dispatching a new query case are proposed. Each of these policies is developed based on a different assumption of how to obtain the best solution. An experimental comparison of our approaches to the existing ones is provided in Section 8.5. Finally, Section 8.6 gives the conclusions.

## 8.2 Modeling Case Base Competence

The concept of case-based competence was first proposed by Smyth and McKenna [Smyt 1998], (i.e. refer as the S-K model in this research), and subsequently it has been developed further to a whole range of concepts which are useful for measuring the problem solving ability of case-bases. As mentioned in Chapter 7, in the S-K model, many statistical properties of a case base, such as the size and density of cases, are used as input parameters for measuring competence. However, this model assumes that there is no overlap among different group of cases. Therefore, if simply taking the group competence as the sum of the individual case competence, and each individual case competence is computed independently without considering the overlapping effects, the resulting group competence may be over- or under-exaggerated. This feature overlap problem has been tackled by Shiu et al. [Shiu 2001b] using fuzzy integral (refer as the S-L model in this chapter). These two models are used as the basis to develop our query case dispatching strategies.

In the previous chapter, we have presented in detail the S-K model (see Section 7.2) and the S-L model (see Section 7.3). Therefore, we will not describe them again in this chapter.

## 8.3 Competence of the CBR Agents

Based on the fuzzy integral approach for calculating the competence of each case group, we can compute the competence of the CBR agents using the S-K and the S-L models respectively. The policies proposed here are based on the DistCBR mode (see Section 8.1), in which different CBR agents are able to communicate and cooperate with one another for recommending a solution. For instance, when agent $A$ is unable to solve a problem, will delegate its authority of solving the problem to $A_j$.

For a given CCBR system, there are $n$ ( $n \geq 1$ ) case-based reasoners, denoted by $CBR_1, CBR_2, \cdots, CBR_n$ . These CBR reasoners can be regarded as $n$ agents $A_1, A_2, \cdots, A_n$ for problem solving in a distributed manner. The corresponding case-bases are $CB_1, CB_2, \cdots, CB_n$, with competence groups $G_1, G_2, \cdots, G_n$ respectively. Each case in these competence groups are represented by $m$ features, $F_1$, $F_2$, …, $F_m$.

***Compute the group competence:***

In computing the competence, we define the similarity between two cases $p$ and $q$ by the following equation: $SM_{pq} = 1/(1 + \sqrt{\sum_{j=1}^{m} (x_{pj} - x_{qj})^2})$, where $x_{ij}$ corresponds to the value of feature $F_j (1 \leq j \leq m), (i = 1, 2, \cdots, m)$.

*Step 1 Detecting the weak-links in the above competence group $G_i$ ($i$=1, 2, …, $n$):*

If $\exists c^* \in G_i$, such that $CompetenceError(c^*) \geq \alpha$, then the competence group $G_i$ is a non-uniform distributed competence group. Otherwise, $G_i$ is a quasi-uniform distributed competence group (see Definition 7.6 in Chapter 7).

*Step 2 Partition the CBR agents according to their competence:*

(1) $CCBR1 \leftarrow \phi, CCBR2 \leftarrow \phi, i = |G|$; where *CCBR*1 consists of those agents who have no feature interactions (or no overlaps among competence groups), while *CCBR*2 consists of those agents who have feature interactions (or overlaps among competence groups).

(2) If $i \neq 0$, compute $CompetenceError(c)$, $\forall c \in G, i = i - 1$;

(3) If there is no *weak-link* in *G*, then *G* is called a quasi-uniform distributed competence group, then add *G* to *CCBR*1, otherwise *G* is called a non-uniform distributed competence group, then add *G* to *CCBR*2, end;

(4) For $1 \leq j \leq n, G \leftarrow G_j$, repeat the above steps (1) to (3).

*Step 3 Compute the competence of each CBR agent in the CCBR1 & CCBR2:*

Assume that there are *m*1, *m*2 competence groups in CCBR1 and CCBR2, respectively.

(1) Compute the competence of each CBR agent in the *CCBR*1 group according to the S-K model. Since the cases are distributed uniformly in each CBR agent, we can get the competence using the S-K competence model [Smyt 1998] directly. They are then ranked in a descending order according their competence, and are denoted as $C_1^1, C_2^1, \cdots, C_{m1}^1$.

(2) Compute the competence of each CBR agent in the *CCBR*2 group according to the S-L model, and rank them as $C_1^2, C_2^2, \cdots, C_{m2}^2$.

*Step 4 Rank the CBR agents according to the respective competence of each CBR agent in the CCBR1 & CCBR2*:

According to the competence, rank the CBR agents in the CCBR1 and CCBR2 system in a descending order as $\{A_1^1, A_2^1, \cdots, A_{m1}^1\}$, $\{A_1^2, A_2^2, \cdots, A_{m2}^2\}$.

## 8.4 Query Dispatching Policies

Based on the computation of each CBR agent in CCBR, three query dispatching policies are proposed in this section. They are: To-Top policy; Strong-Strong policy and Best Committee policy, which are described in Sections 8.4.1, 8.4.2 and 8.4.3, respectively.

### 8.4.1 To-Top Policy

The main idea of this policy is to choose the CBR agent which has the maximal competence in the corresponding *CCBR* system, i.e. $A_1^1$ in *CCBR*1 or $A_1^2$ in *CCBR*2 system. *CCBR*1 is chosen as the problem-solving agent if there are no feature interactions among different competence groups; otherwise, *CCBR*2 is chosen.

For example, in a travel-planning problem which will be described in Section 8.5, the hotels are classified by the number of stars, therefore when the user specify the type of accommodations (e.g. the number of stars), this will limit the choices of the available hotels. In this case, the features "accommodation" and "hotel" are interacting. The dispatching procedure is as follow: if agent $A_i$ receives an input query, it will try to solve it. When the solution is satisfactory (i.e. within a user defined threshold of solution accuracy, and efficiency), it becomes the answer to the input query. Otherwise, it will dispatch the problem to $A_1^1$ or $A_1^2$ for solving. If $A_i$ is one of the agents in *CCBR*1, then it dispatches the problem to the agent $A_1^1$, otherwise the problem will goes to $A_1^2$ in *CCBR*2.

### 8.4.2 Strong-Strong Policy

In this policy, we assume that we could not determine whether the features are having interactions or not. Consider the travel-planning problem again, we are not sure whether there are feature interactions or not between the features "season" and "hotel" or between the features "holiday duration" and "season". Thus, it is better to ask more than one agents to suggest the solutions. We choose the most competent agent (i.e. one from each collaborative CBR system). That is, if the agent $A_i^j (i \neq 1,\ j = 1, 2)$ receives the problem, and cannot solve it satisfactorily, it will ask the agents $A_1^1$ in the *CCBR*1 and agent $A_1^2$ in the *CCBR*2 to solve it in parallel. One of these suggested solutions will be used based on an earlier assessment of these two agents' ability. (Note that these two agents belong to the two *CCBR* systems, therefore the selection has already considered the feature interaction property).

### 8.4.3 Best-Committee policy

If time is not the critical issue and getting better solution is the main concern, then the user can ask more agents for suggested solutions. In general, the more the agents are involved, the more accurate the answer will be. That is, if the agent $A_i^j (i \neq 1,\ j = 1, 2)$ receives the problem, it could follow the To-Top and Strong-Strong policies first for solving the problem. However, if the solution is not satisfactory, it can ask the agents $A_1^j, A_2^j, \cdots, A_{i-1}^j$, (i.e. those agents that are better in competence), to solve the problem. Each agent will offer a solution to the problem, and the final solution is chosen according to the user’s preference, such as preferred accuracy.  This policy provides the user a flexible choice, when he wants to get the best solution, then he can ask all the agents for suggestions. This policy is the same as the "Committee" policy proposed by Plaza and Ontañón.

## 8.5 Experimental Evaluation

This section demonstrates some experimental results of the proposed query dispatching policies (Section 8.4) and their comparisons with some existing ones.

We used the travel case base that is available from AI-CBR web-site (i.e. www.ai-cbr.org). It contains 1470 cases, and we randomly selected 1100 cases for our experiment. Each test case describes a holiday-package tour from Europe/ North Africa, and consists of 9 features. In dividing the cases into different groups for measuring competence, we use a random selection approach. The reason is that because in real life, there may be missing values in the cases, therefore a feature based grouping of cases may not be possible. In the experiment, 800 cases are chosen randomly as learning data, and 300 cases are chosen as testing data. The learning data are further divided into 4, 5, 6, 7 and 8 groups (i.e. each group represent one CBR agent, therefore if 4 agents are used, each of them consists of 200 cases, etc). The feature "price" is chosen as the solution feature. The testing is based on the evaluation of the solution accuracy and the mean cost of solving (i.e. time consumption).

The objective of the experiment is to determine the "price" of each travel plan using our proposed policies. A comparison of our approach to some exiting ones [Plaz 2001] is also carried out. The mean relative error (i.e. the difference between the actual result and the predicted result and divided by the actual result) is used to compute the accuracy.

In our experiment, if four agents are used to predict the "price" of a particular testing case (such as Case number 987), our three policies will give the following results respectively: \$4,708.25, \$3,536.72, and \$4,561.35. The accuracies are 84.94%, 86.43% and 88.26% respectively, which are shown in Fig. 8.1. The mean cost is the relative CPU time of the isolated agent, and assuming the mean time cost of the isolated agent is ONE unit, then the mean time costs of the collaborative policies are given in Fig. 8.2. We have conducted five testing runs, and each testing has different number of agents. These agents are formed by randomly re-organize the 800 testing cases.

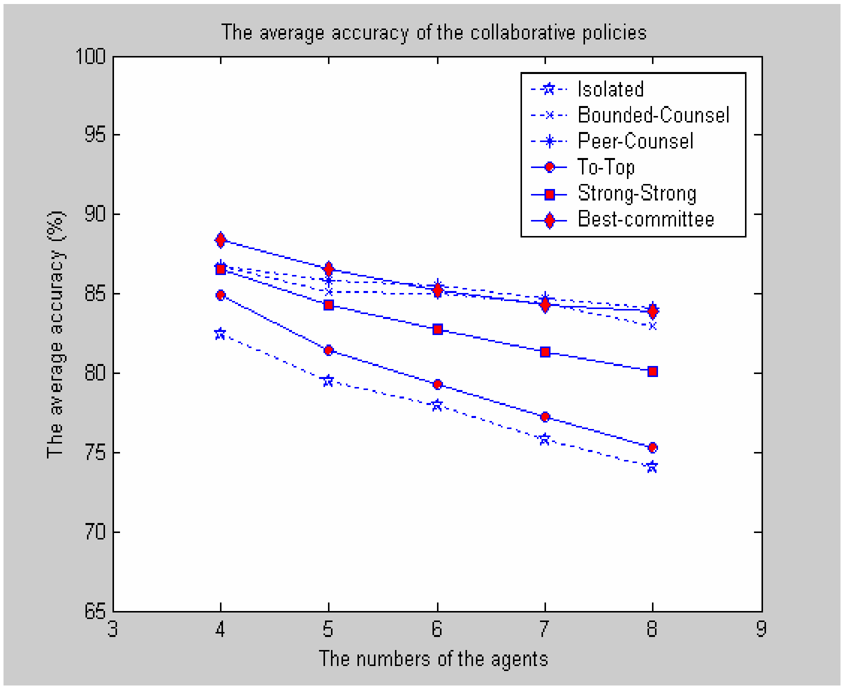

Fig. 8.1 The average accuracy of collaborative policies

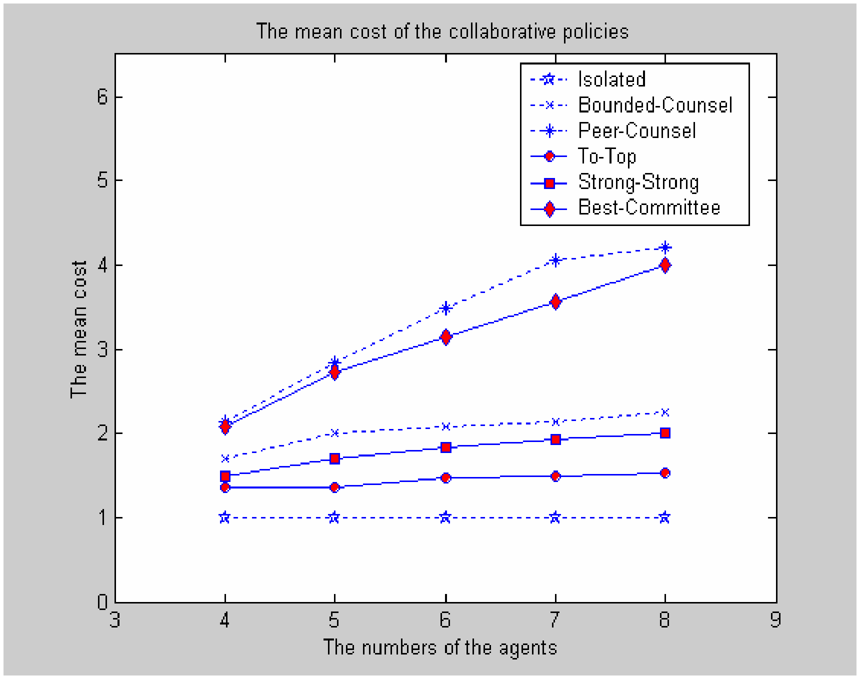

Fig. 8.2 The mean cost of the collaborative policies

The result shows that our policies use less time and still can achieve the same accuracy as the other existing ones. More specifically, Fig. 8.1 shows that all of six case dispatching policies are better than the isolated agent. The Strong-Strong policy is better than the To-Top policy and the Best-Committee policy is the best one. The Strong-Strong policy has similar accuracy to the Bounded-Counsel policy and the Peer-Counsel policy. Here we did not include the Committee policy in the experiment because it can be viewed as a special case of Best-Committee policy.

Some limitations of our experiment include: (1) since the number of cases is fixed, an increase of the number of agents will decrease their competence correspondingly, as the result, the experimental accuracy will decrease with the increasing number of the agents; and (2) a pre-processing of the agents' competence is required.

On the other hand, the merits of our method are: (1) the computation time for finding a satisfactory solution is comparatively less than the current approaches; (2) our three policies can provide alternative case dispatching methods to users according to their preference; and (3) our approach can be used to model feature interaction among cases.

## 8.6 Summary

In this chapter, we have presented our approach of dispatching query to different CBR agents in the context of CCBR. The policies are based on the concept of case and group competence. The problem of feature interaction among cases is also tackled using the fuzzy integral model. Our approach has been demonstrated empirically with some testing cases from the travel domain, and the result shows that our approach is better than the existing ones. Further research includes a more detail investigation of case feature interactions, as well as their modeling in distributed CBR environments. Furthermore, a more theoretical analysis and evaluation of our approach can be carried out.

# Chapter 9

# Conclusions and Future Work

In previous chapters, we have developed some soft computing techniques to extract case knowledge in the development of CBR systems. In this chapter, we evaluate and conclude our research work and provide the directions of future research. Section 9.1 presents a qualitative evaluation of each developed technique through analyzing its characteristics such as merits, limitations, and the situations that this technique can /cannot be used. Based on Section 9.1, we summarize the whole research work in Section 9.2. The possible future work is discussed in Section 9.3.

## 9.1 Evaluation of the Case Knowledge Extraction Techniques

Since all the techniques for case knowledge extraction are developed in the context of CBR systems, we firstly discuss the similarity assumption in CBR which is important in case retrieval.

### 9.1.1 Similarity Assumption in CBR

Throughout this research work, we use the conventional similarity assumption during case retrieval, i.e., it is preferred to retrieve a set of cases that are maximally similar to the problem in question. This similarity assumption works well in most problem domains where similar cases have the similar solutions. However, in 2001, Barry Smyth and Paul McClave [Smyt 2001a] suggested that diversity can be as important as similarity in some other domains, such as in case-based recommender systems. In these systems, users prefer to obtain diverse choices for their input queries. In their work, a number of different retrieval strategies are proposed to improve diversity among the retrieved cases without compromising similarity.

## 9.1.2 Rough Set-Based Feature Reduction

The advantage of using rough sets for feature reduction is that the irrelevant features can be removed without affecting the ability to distinguish cases in different equivalence classes. On the other hand, the main disadvantage of the traditional rough set-based feature reduction methods is the high computational complexity. For example, if there are $n$ cases and $m$ features, $O(n^2 \times m)$ computations are required (See Section 3.2). In this research, we overcome this limitation through introducing a new concept of approximate reduct (Definition 3.7), which can be obtained quickly. The computational complexity has been reduced to be linear with the number of cases and features. Therefore, the developed feature reduction technique in Chapter 3 is fast and effective, which has been demonstrated in the experimental results in Section 3.4.

As mentioned in Chapter 3 and Chapter 5, the fast rough set-based feature reduction is different from KPCA mainly in the aspects of rationale; training type; data type; and required training time. For details, readers can see Table 3.6 and Table 5.11. The qualitative comparisons in Table 5.11 are essentially a comparison between the developed FR technique and KPCA. The main difference of the rough set-based FR and KPCA is that the former is supervised while the latter is unsupervised. Supervised methods can only be used to problems with known class labels (e.g, classification problems), and unsupervised methods can be also used to problems without knowing the class labels such as clustering problems. Since we mainly consider classification problems, the supervised methods such as the proposed FR can be used. Based on these analysis, users can choose either of the methods depending on the used data and the requirements on efficiency and accuracy.

To give a better understanding of our feature reduction method, we also present the limitations as follows:

(1) It works better with symbolic data. When handling numerical data, the data need to be discretized firstly to induce the equivalence classes. This may result some information loss and therefore affect the performance in terms of accuracy.

(2) The determination of the parameter $\beta$ is empirical and heuristic based during the testing and the best values are data dependent. The finally determined parameter values may not the globally optimal values.

### 9.1.3 Learning Similarity Measures of Nominal Features

In Chapter 4, we presented a GA-based method to learn the similarity measures of nominal features. The main contribution is that the similarity between two different nominal feature values is described by a degree, i.e., a value in [0, 1], instead of either zero or one. For example, in Table 4.5, the average similarity of two different colors Yellow and Purple is 0.61. This improves case matching and retrieval and the clustering performance. The method is suitable to handle nominal data or symbolic data with limited feature values.

There are also some limitations during the development of the GA-based learning method: (1) For large nominal feature set and feature value set, the computation cost is high; (2) the used GA algorithm can be further improved through optimizing the selection probability and the mutation probability (see Section 4.3).

### 9.1.4 Case Selection Methods

We built up different case selection strategies (see Figs 5.2, 5.3, and 5.6) in Chapter 5 based on the similarity measure, the concepts of case coverage and reachability (Definitions 5.3-5.4), and the nearest neighbor principle. They have the abilities of

(1) Reducing the size of case bases and preserving the competence;
(2) Dealing with different case densities;

(3) Identifying and removing both redundant and noisy cases;

(4) Producing a subset of cases which does not change the knowledge representation;

(5) Improve the efficiency and the accuracy.

Furthermore, it has been shown that, by combining the fast rough set-based feature reduction with these case selection methods, the performance can be further enhanced in terms of both storage and accuracy. Two combination methods, RFRCS1 (see Fig. 5.7) and RFRCS2 (see Fig. 5.8) are given based on different definition of the "best" approximate reduct in case selection. The former is fast and the latter is accurate. Users can choose one of them according to different requirements of the efficiency and accuracy to construct the final case base.

It should be pointed out that, in the development of case selection methods, we have only considered the case bases which consist of homogeneous data regions. In these case bases, the noisy cases are defined as the cases that cannot be correctly classified by their k-nearest neighbors. Therefore, our approach cannot be directly used on case bases containing heterogeneous data regions, which may result that some useful cases are misclassified as noisy cases.

## 9.1.5 Rough LVQ-Based Case Generation

Case generation is considered as an alternative way of case selection to reduce the size of case bases. In Chapter 6, the representative cases can be generated using fuzzy sets, rough sets, and learning vector quantization (LVQ). It deals with numerical data and the LVQ algorithm is supervised process based on the reduced feature set after FR (see the LVQ-based Case Generation Algorithm in Section 6.3.2). It has been shown that the storage decreases and the accuracy is improved comparing with case generation based on random, SOM and LVQ (see Tables 6.4-6.5). Better clustering performance is also obtained based on the generated cases (see Table 6.6). The main weakness is that, in the conducted experiments, the used case bases are relatively small which contain 150-768 cases and 4-10 features. Larger case bases are needed in the testing to further demonstrate the effectiveness of the developed case generation method.

### 9.1.6 Fuzzy Integral-Based Competence Model

In Chapter 7, a competence model is built based on fuzzy measures (Definition 7.1) and the corresponding fuzzy integral (Definition 7.3). The merits of this model include: (1) It is able to describe the interactions among case coverage due to the non-additive characteristic (Definition 7.2) of the fuzzy measure; (2) non-uniformed case distribution has been taken into account. Therefore, the fuzzy integral-based competence model can reflect the case base competence more accurately than the S-K model (see Section 7.2.1).

Table 9.1 Comparisons of the fuzzy integral-based competence model and the S-K model

| Factors | Fuzzy integral model | S-K model |
|---|---|---|
| Group Size | √ | √ |
| Case density | √ | √ |
| Case distribution | √ | × |
| Overlaps of case coverage | √ | × |

Table 9.1 describes the comparisons between these two competence models. There are four factors which are considered to affect the case base competence: the size of the competence group; case density; case distribution; and the possible overlaps among the case coverage sets. Here "√" means that the factor has been taken into account; and "×" means that the factor has been ignored. From Table 9.1, the developed fuzzy integral-based model is more comprehensive and can be used to case bases with different distributions. It can be considered as a generalization of the S-K model.

However, there are various fuzzy measures and fuzzy integrals which have different characteristics and different number of parameters. Additional computational effort is required determine the fuzzy measures, e.g., if there are $n$ competence groups, $2^n$ computations are needed. Therefore, this model can be applied when time is not the critical issue. On the other hand, the S-K model is appropriate when the case distribution is uniform and there is no overlap among case coverage sets.

To summarize, this section provides a brief qualitative evaluation of the developed soft computing based techniques for case knowledge extraction. Firstly, we have discussed the similarity assumption in CBR, which works well in most CBR systems but may have some limitations in case-based recommender systems. The characteristics of each technique are then analyzed, including the advantages and disadvantages.

## 9.2 Conclusions of the Research Work

The whole research work is conducted in the context of CBR systems, especially CBR classifiers. We mainly consider three performance criteria: accuracy, efficiency and competence. Larger case bases require more case retrieval time and therefore degrade the problem-solving efficiency; on the other hand, these larger case bases enhance the competence and accuracy because they have more cases to cover the problems. How to make a balance between the efficiency and competence becomes a main issue in this research. Furthermore, in many real world situations, data and information collected are always incomplete, uncertain and vague, thus, the use of soft computing principles to achieve tractability, robustness and low solution cost is inevitable. Case knowledge extraction is the essential objective to build both compact and competent CBR systems.

We have developed a set of soft computing based techniques for the extraction of case knowledge from data. They reduce the size of the case base through removing the redundancy and noises, as well as preserve the problem-solving ability in terms of competence. The built techniques include rough set based feature reduction, GA-based supervised algorithm for learning similarity measures, case selection based on case coverage and reachability and NN principle, rough LVQ based case generation, and fuzzy integral based competence model. Among these techniques, the rough set-based FR and

CS are the most important methods, which identify and remove both irrelevant features and noisy and redundant cases. Thus, the size of the case bases is reduced and therefore the efficiency is improved, and simultaneously the accuracy and competence of the case bases are preserved or even improved. The combination of the proposed FR and CS methods are tested and compared with traditional methods such as KPCA and SVM. The experimental results are very promising, and support our objective of trying to develop a compact and competent CBR system through case knowledge extraction. Our proposed FRCS combination is much faster than KPCA and SVMs and can still achieves acceptable accuracy. When the updating of case bases and the retraining of SVMs are very frequent, the FRCS can also outperform KPCA and SVMs in terms of classification time.

## 9.3 Future Work

This section briefly discusses the possible future work in the aspects of similarity assumption, parameter determination, data distribution, case adaptation, and application to large case bases. These extensions target to overcome the limitations of the developed techniques for case knowledge extraction, which we have mentioned in Section 9.1.

Throughout the research work, the conventional similarity assumption is used in case matching and retrieval. The assumption may not be applicable in situations that require diversity of solutions, e.g., a case-based recommender system. The future work may incorporate the measurement of case diversity.

In the rough set-based feature reduction and case selection, the parameters, i.e., $\beta$ and $\eta$, are determined empirically and also heuristics based. The values for these parameters may not be the global optimal. Therefore, in future work, we may apply some learning methods such as neural networks to optimize these values. However, additional training effort is required in the learning process. Therefore, new strategies may need to be developed to strike a trade-off between the "best" value and the cost of obtaining them.

As mentioned in Section 9.1.4, we have only considered cases from the homogeneous regions in the development of the case selection strategies. Therefore, our built case selection algorithms can not be directly used for heterogeneous data regions. We may tackle this problem by re-clustering the case bases to transform the heterogeneous distributions to homogeneous ones. Another possible solution is to enhance our developed prototypical case selection strategy, or combined with others.

In Chapters 5 and 6, CS and CG methods are used to extract representative cases for the same purpose of reducing the size of case bases. The selected cases using CS forms a subset of the original case base and therefore have the same features. On the other hand, the generated prototypical cases using CG methods are not necessarily the original cases. A more comprehensive comparison study of the CS and CG methods is required in future work, which can provide readers better understanding of these techniques. Another straightforward further work is to consider more and larger real-life case bases in the testing in Chapter 6 to demonstrate the effectiveness of the GA-based learning method and the rough LVQ-based case generation.

In the research work, three main criteria of storage requirement, classification accuracy, and problem-solving efficiency are taken into account in the performance evaluation of the developed techniques. However, adaptation ability of the solutions is also one of the important criteria to evaluate the problem-solving quality of the CBR systems. In the current research work, we have not investigated these properties much because of its domain-dependent characteristics. In our future work, for a given specific application domain, the adaptation of the retrieved cases may be considered and defined during the development of the soft computing techniques. This will make the whole research work be more complete for building efficient and effective CBR systems.

A case study may be considered involving such a case base containing noise, redundancy, uncertainties in both features and cases. The developed techniques in this research are used to deal with the case base and extract case knowledge from it. This work may consist of the following stages: (1) selecting an appropriate problem domain such as diagnosis; (2) collecting raw data with features and solutions, i.e., symptoms and the corresponding diseases for the diagnosis problem; (3) applying the proposed techniques to handle the noise, redundancy and uncertainties that possibly contained in the collected data; (4) evaluating the performance based on accuracy, efficiency and solution quality; (5) improving the techniques based on the evaluation results. This will make the work be in a more consistent framework and in a more integrated manner. Note that, it does not necessarily using all the methods in one application. Whether a technique should be incorporated or not is determined by the characteristics of the problem domain and the requirement of systems or users. For example, if the features are carefully selected by the experts beforehand, the FR process can be ignored.

Finally, it is a possible direction of our future work to use multi-objective optimization as an alternative way to achieve our objectives in case knowledge extraction. In this research, our main concern is to balance several different performance criteria such as efficiency and accuracy, either in individual task or overall task. Multi-objective optimization such as evolutionary computing is widely used to trade-off optimal solutions of multiple tasks and therefore should be appropriate method to fulfill the tasks in this work. Based on this idea, an optimization model is required to be built for each task in question. For example, in case generation, we can use GA to minimize the intra-distance and maximize the inter-distance. For a set of generated cases, the fitness function in GA may be defined as Ratio = Inter-distance/ Intra-distance. The objective is to maximize the Ratio value. However, it seems that the multi-objective optimization is not feasible to dealing with the overall task of case knowledge extraction. This is because that, (1) it is difficult to define a fitness function which can accurately reflect the relationship among efficiency, competence and accuracy; (2) the evolutionary algorithm requires substantial computational load. In each generation in the evolutionary algorithm, the fitness value is needed to be computed, which involves the computation of retrieval time, classification time for efficiency, case base competence and classification accuracy. Here each solution can be regarded as a weight vector of the features and cases, which evaluates the feature and case importance.

# References

[Aha 1991] D. W. Aha, K. Dennis, and K. A. Marc. Instance-based learning algorithms. Machine Learning, vol. 6, pp. 37-66, 1991.

[Aha 1992] D. W. Aha. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. International Journal of Man-Machine Studies, vol. 36, pp. 267-287, 1992.

[Ande 1973] M. R. Anderberg. Cluster Analysis for Applications. Academic Press, San Diego, CA, 1973.

[Astr 1970] M. M. Astrahan. Speech analysis by clustering, or the hyperphoneme method. In Stanford A. I. Project Memo. Stanford University, CA, 1970.

[Baza 2000] J. Bazan, H.S. Nguyen, S.H. Nguyen, P. Synak, and J. Wróblewski. Rough Set Algorithms in Classification Problem. In Rough Set Methods and Applications, L. Polkowski, S. Tsumoto and T.Y. Lin (eds.). Physica-Verlag, Heidelberg, New York, pp. 49-88, 2000.

[Beas 1993] D. Beasley, D. R. Bull, and R. R. Martin. An overview of genetic algorithms part 1 fundamentals. Technical report, University of Purdue, 1993.

[Boni 1997] P. P. Bonissone and W. Cheetham. Financial application of fuzzy case-based reasoning to residential property valuation. In Proceedings of the Sixth IEEE International Conference on Fuzzy Systems (FUZZ-IEEE-97), pp. 37-44, Barcelona, Catalonia, Spain, July 1-5, 1997.

[Brig 2002] H. Brighton and C. Mellish. Advances in instance selection for instance-based learning algorithms. Data Mining and Knowledge Discovery, vol. 6, no. 2, pp. 153-172, 2002.

[Brow 1990] P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. A statistical approach to machine translation. Computational Linguistics, vol. 16, no. 2, pp. 79-85, 1990.

[Cao 2003] G. Cao, S. C. K. Shiu and X. Z. Wang. A fuzzy-rough approach for the maintenance of distributed case-based reasoning systems. Soft Computing, vol. 7, no. 8, pp. 491-499, 2003.

[Chan 1974] C. L. Chang. Finding prototypes for nearest neighbor classifiers. IEEE Trans. Computers, vol.C-23, no.11, pp.1179-1184, 1974.

[Chan 1998] C. C. Chan. A rough set approach to attribute generalization in data mining. Information Science, vol. 107, no. 1-4, pp. 169-176, 1998.

[Crof 1979] W. B. Croft and D. Harper. Using probabilistic models of document retrieval without relevance information. Journal of Documentation, vol. 35, no. 4, pp. 285-295, 1979.

[Darr 1993] W. Darrell. A genetic algorithm tutorial. Technical Report, CS-93-103, Colorado State University, 1993.

[Domi 1995] P. Domingos. Rule induction and instance-based learning: A unified approach. In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95), pp. 1226-1232, Montreal, Canada, August 20-25, 1995.

[Devi 1982] P. A. Devijver and J. Kittler. Pattern Recognition: A Statistical Approach. Englewood Cliffs: Prentice Hass, 1982.

[Emam 2001] K. E. Emam, S. Benlarbi, N. Goel and S. N. Rai. Comparing case-based reasoning classifiers for predicting high risk software components. Journal of Systems and Software, vol. 55, no. 3, pp. 301-320, 2001.

[Fore 2002] G. L. Foresti. Invariant feature extraction and neural trees for range surface classification. IEEE Trans. Systems, Man and Cybernetics (Part B), vol. 32, no. 3, pp. 356-366, 2002.

[Fuhr 1992] N. Fuhr. Probabilistic models in information retrieval. Computer Journal, vol. 35, no.3, pp. 243-255, 1992.

[Gama 1998] J. Gama and P. Brazdil. Constructive induction on continuous spaces. In Feature Extraction, Construction and Selection: A Data Mining Perspective. H. Liu and H. Motoda (eds.). Kluwer Academic Publishers, pp. 289-303, 1998.

[Gate 1972] G. W. Gates. The reduced nearest neighbor rule. IEEE Trans. Information Theory, vol. 18, no. 3, pp. 431-433, 1972.

[Garr 1999] J. M. Garrell i Guiu, E. Golobardes i Ribé, E. Bernadó i Mansilla and X. Llorà i Fàbrega. Automatic diagnosis with genetic algorithms and case-based reasoning. Artificial Intelligence in Engineering, vol. 13, no. 4, pp. 367-372, 1999.

[Gowd 1992] K.C. Gowda and E. Diday. Symbolic clustering using a new similarity measure. IEEE Trans. Systems, Man and Cybernetics, vol. 22, pp.368-378, 1992.

[Gela 1989] P. Geladi, H. Isaksson, L. Lindqvist, S. Wold and K. Esbensen. Principle component analysis of multivariate images. Chemometrics and Intelligent Laboratory Systems, vol. 5, no. 3, pp. 209-220, 1989.

[Guo 1992] H. Guo and S. B. Gelfand. Classification trees with neural network feature extraction. IEEE Trans. on Neural Networks, vol. 3, no. 6, pp. 923-933, 1992.

[Han 2004] J. Han, X. Hu, and T. Y. Lin. Feature subset selection based on relative dependency between attributes. In Proceedings of the Fourth International Conference of Rough Sets and Current Trends in Computing (RSCTC-04), pp. 176-185, Uppsala, Sweden, June 1-5, 2004.

[Hart 1968] P. E. Hart. The Condensed Nearest neighbor Rule. Institute of Electrical and Electronics Engineers Transactions on Information Theory, vol. 14, pp. 515-516, 1968.

[Hett 1998] S. Hettich, C. L. Blake, and C. J. Merz, UCI Machine Learning Databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

[Hinr 1992] T. R. Hinrihs. Problem Solving in Open Worlds. Lawrence Erlbaum Associates, Hillsdate, NJ, 1992.

[Hsu 1999] C. C. Hsu and C. S. Ho. Acquiring patient data by an intelligent interface agent with medicine-related common sense reasoning. Expert Systems with Applications: An International Journal, vol. 17, no. 4, pp. 257-274, 1999.

[Hu 1998] Y. Hu. Construction induction: Covering attribute spectrum. In Feature Extraction, Construction and Selection: A Data Mining Perspective, H. Liu and H. Motoda (eds.). Kluwer Academic Publishers, pp. 257-272, 1998.

[Jens 2004] R. Jensen and Q. Shen. Fuzzy-rough attribute reduction with application to web categorization. Fuzzy Sets and Systems, vol.141, no. 3, pp. 469-485, 2004.

[Joll 1986] I. T. Jolliffe. Principle Component Analysis. New York: Springer, 1986.

[Jone 2000] K. Sparck Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments. Part I. Information

Processing and Management, vol. 36, no. 6, pp. 779-808, 2000.

[Kala 2001] E. Kalapanidas and N. Avouris. Short-term air quality prediction using a case-based classifier. Environmental Modelling & Software, vol. 16, no. 3, pp. 263-272, 2001.

[Kim 1997] D. Kim and C. Kim. Forecasting time series with genetic fuzzy predictor ensemble. IEEE Trans. Fuzzy Systems, vol. 5, no, 4, pp. 523-535, 1997.

[Kocs 2004] A. Kocsor and L. Toth. Kernel-based feature extraction with a speech technology application. IEEE Trans. Signal Processing, vol. 52, no. 8, pp. 2250-2263, 2004.

[Kram 1991] M. A. Kramer. Nonlinear principle component analysis using autoassociative neural networks. A. I. Ch. E. Journal, vol. 37, no. 2, pp. 233-243, 1991.

[Koho 1988] T. Kohonen. Self-organization and associative memory. New York, Springer-verlag, 1988.

[Koho 1997] T. Kohonen. Self-organizing maps. New York, Springer-verlag, 1997.

[Kolo 1993] J. Kolodner. Case-Based Reasoning. Morgan Kaufmann, San Francisco, 1993.

[Laff 2003] J. Lafferty and C. Zhai. Probabilistic relevance models based on document and query generation. In Language Modeling and Information Retrieval, W. B. Croft and J. Lafferty (eds.). Kluwer Academic Publishers, 2003.

[Leak 1998] D. Leake and D. Wilson. Case base maintenance: Dimensions and directions. In Proceedings of the Fourth European Workshop on Case-based Reasoning (EWCBR-98), pp. 196-207, Dublin Ireland, September 23-25, 1998.

[Leak 2000] D. Leake and D. Wilson. Guiding case-base maintenance: Competence and performance. In Proceedings of the Fourteenth European Conference on Artificial Intelligence Workshop on Flexible Strategies for Maintaining Knowledge Containers, pp. 47-54, 2000.

[Leak 2000] D. Leake and D. Wilson. Remembering why to remember: performance-guided case-base maintenance. In Proceedings of the Fifth European Workshop on Case-Based Reasoning (EWCBR-00), Trento, Italy, Springer-Verlag, Berlin, pp. 161-172, 2000.

[Leak 2001] D. B. Leake and R. Sooriamurthi. When two cases are better than one: exploiting multiple case-bases. In Proceedings of the Fourth International Conference on Case-Based Reasoning (ICCBR-01), Springer-Verlag, pp. 321-335, Vancouver, BC, Canada, July 30-August 2, 2001.

[Lewi 1999] D.D.Lewis. Reuters-21578 text categorization test collection distribution 1.0. http://www.research.att.com/~lewis, 1999.

[Liao 1998] T. W. Liao, Z. Zhang, and C. R. Mount. Similarity measures for retrieval in case-based reasoning systems. Applied Artificial Intelligence, vol. 12, pp. 267-288, 1998.

[Liao 2004] T. W. Liao. An investigation of a hybrid CBR method for failure mechanisms identification. Engineering Applications of Artificial Intelligence, vol. 17, no. 1, pp. 123-134, 2004.

[Liu 1998a] H. Liu and H. Motoda. Feature Selection for Knowledge Discovery and Data Mining. Kluwer Academic Publishers, 1998.

[Liu 1998b] H. Liu and H. Motoda. Feature Extraction Construction and Selection: A Data Mining Perspective. Kluwer Academic Publishers, 1998.

[Mang 1996] P. Mangiameli, S. K. Chen and D. West. A comparison of SOM neural network and hierarchical clustering methods. European Journal of Operational Research, vol. 93, pp. 402-417, 1996.

[Mash 1993] M. L. Masher and D. M. Zhang. CADSYN: A case-based design process model. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, vol. 7, no. 2, pp. 97-110, 1993.

[Mall 1998] Y. Mallet, O. de Vel, and D. Coomans. Integrated feature extraction using adaptive wavelets. In Feature Extraction, Construction and Selection: A Data Mining Perspective. H. Liu and H. Motoda (eds.). Kluwer Academic Publishers, pp. 175-189, 1998.

[Mika 1999] S. Mika, B. Sch olkopf, A. Smola, K.R. Mller, M. Scholz, and G Ratsch. Kernel PCA and de-noising in feature spaces. Advances in Neural Information Processing Systems, vol. 11, pp. 536-542, 1999.

[Mitr 2002a] P. Mitra, C. A. Murthy, and S. K. Pal. Density based multiscale data condensation. IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 24, no. 6, pp. 734-747, 2002.

[Mitr 2002b] S. Mitra, K. M. Konwar, and S. K. Pal. Fuzzy decision tree, linguistic rules and fuzzy knowledge-based network: generation and evaluation. IEEE Trans. Systems, Man and Cybernetics (Part C), vol. 32, no. 4, pp. 328-339, 2002.

[Mitr 2002c] P. Mitra, C. A. Murthy, and S. K. Pal. Unsupervised feature selection using feature similarity. IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 24, no. 3, pp. 301-312, 2002.

[Nguy 1997] H. S. Nguyen and A. Skowron. Boolean reasoning for feature extraction problems. In Proceedings of the Tenth International Symposium on Methodologies for

Intelligent Systems (ISMIS-97), pp. 117-126, Charlotte, North Carolina, USA, October 15-18, 1997.

[Pal 2001] N. R. Pal and S. Chakraborty. Fuzzy rule extraction from ID3-type decision trees for real data. IEEE Trans. Systems, Man and Cybernetics (Part B), vol. 31, no. 5, pp. 745-754, 2001.

[Pal 2002] S. K. Pal and P. Mitra. Multispectral image segmentation using rough set initialized EM algorithm. IEEE Trans. Geoscience and Remote Sensing, vol. 40, no. 11, pp. 2495-2501, 2002.

[Pal 2004a] S. K. Pal and S. C. K. Shiu. Foundations of Soft Case-Based Reasoning. John Wiley, New York, 2004.

[Pal 2004b] S. K. Pal and P. Mitra. Case generation using rough sets with fuzzy representation. IEEE Trans. Knowledge and Data Engineering, vol. 16, no. 3, pp. 292-300, 2004.

[Pal 2004c] S. K. Pal, B. Dasgupta and P. Mitra. Rough-self organizing map. Applied Intelligence, vol. 21, no. 3, pp. 289-299, 2004.

[Pan 2005] R. Pan, Q. Yang, J. J. Pan, L. Li. Competence Driven Case-Base Mining. In Proceedings of the AAAI 2005, pp. 228-233, Pittsburg, PA, USA, 2005.

[Pap 1996] E. Pap. Null-additive Set Function. Kluwer Academic, Dordrecht, Netherlands, 1996.

[Pawl 1982] Z. Pawlak. Rough sets. International Journal of Computer and Information Science, vol. 11, no. 5, pp. 341-356, 1982.

[Pawl 1991] Z. Pawlak. Rough Sets: Theoretical Aspects of Reasoning about Data.

Kluwer Academic, Dordrecht, 1991.

[Pras 1996] M. V. N. Prasad, V. Lesser, and S. Lander. Retrieval and reasoning in distributed case-bases. Journal of Visual Communication and Image Representation, Special Issue on Digital Libraries, vol. 7, no. 1, pp. 74-87, 1996.

[Plaz 1997] E. Plaza, J. L. Arcos, and F. J. Martín. Cooperative case-based reasoning. In Distributed Artificial Intelligence meets Machine Learning: Learning in Multi-Agent Environments, G. Weiss (ed.), Springer-Verlag, Berlin, pp.180-201, 1997.

[Plaz 2001] E. Plaza and S. Ontañón. Ensemble Case-Based Reasoning: collaboration policies for multi-agent cooperative CBR. In Proceedings of the Fourth International Conference on Case-Based Reasoning (ICCBR-01), pp. 437-451, Vancouver, BC, Canada, 2001.

[Quin 1986] J. R. Quinlan. Induction of decision trees. Machine Learning, vol. 1, no. 1, pp. 81-106, 1986.

[Quin 1987] J. R. Quinlan. Simplifying decision trees. International Journal of Man-Machine Studies, vol. 27, no. 3, pp. 221-234, 1987.

[Quin 1993] J. R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kauffman, 1993.

[Raci 1997] K. Racine and Q. Yang. Maintaining unstructured case bases. In Proceedings of the Second International Conference on Case-based Reasoning (ICCBR-97), pp. 553-564, Providence, Rhode Island, July 25-27, 1997.

[Rijs 1979] C. J. van Rijsbergen. Information Retrieval, second ed. London: Butterworth & Co. (Publishers) Ltd., 1979.

[Ritt 1975] G. L. Ritter, H. B. Woodruff, S. R. Lowry, and T. L. Isenhour. An algorithm for the selective nearest neighbor decision rule. IEEE Trans. Information Theory, vol. 21, no. 6, pp. 665-669, 1975.

[Robe 1977] S. Robertson. The probability ranking principle in IR. Journal of Documentation, vol. 33, no. 4, pp. 294-304, 1977.

[Salz 1991] S. Salzberg. A nearest hyperrectangle learning method. Machine Learning, vol. 6, no. 3, pp. 251-276, 1991.

[Scho 1999] B. Scholkopf, S. Mika, C. J. C. Burges, K. R. Muller, and A. J. Smola. Input space versus feature space in kernel-based methods. IEEE Trans. Neural Networks, vol. 10, no, 5, pp. 1000-1017, 1999.

[Seti 1998] R. Setiono and H. Liu. Feature extraction via neural networks. In Feature Extraction, Construction and Selection: A Data Mining Perspective. H. Liu and H. Motoda (eds.). Kluwer Academic Publishers, pp. 191-204, 1998.

[Shen 2002] Q. Shen and A. Chouchoulas. A rough-fuzzy approach for generating classification rules. Pattern Recognition, vol. 35, pp. 2425-2438, 2002.

[Shen 2004] L. Shen and H. T. Loh. Applying rough sets to market timing decisions. Decision Support Systems, vol. 37, no. 4, pp. 583-597, 2004.

[Shiu 2001a] S. C. K. Shiu, D. S. Yeung, C. H. Sun and X. Z. Wang. Transforming case knowledge to adaptation knowledge: An approach for case-base maintenance. Computational Intelligence, vol. 17, no. 2, pp. 295-313, 2001.

[Shiu 2001b] S. C. K. Shiu, Y. Li, and X. Z. Wang. Using fuzzy integral to model case-base competence. In Proceedings of Soft Computing in Case-Based Reasoning Workshop in Conjunction with the Fourth International Conference in Case-Based Reasoning

(ICCBR-01), pp. 206-212, Vancouver, Canada, July 30-August 2, 2001.

[Sing 2001] P. K. Singal, S. Mitra, and S. K. Pal. Incorporating of fuzziness in ID3 and generation of network architecture. Neural Computing and Applications, vol. 10, pp. 155-164, 2001.

[Skow 1992] A. Skowron and C. Rauszer. The discernibility matrices and functions in information systems. Intelligent Decision Support-Handbook of Applications and Advances of the Rough Sets Theory, K. Slowinski (ed.), Kluwer, Dordrecht, pp. 331-362, 1992.

[Smyt 1995] B. Smyth and M. Keane. Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems. In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-95), pp. 377-382, Montreal, Canada, August 20-25, 1995.

[Smyt 1998a] B. Smyth and E. McKenna. Modeling the competence of case-bases. In Proceedings of the Fourth European Workshop on Case Based Reasoning (EWCBR-98), pp. 208-220, Dublin, Ireland, September 23-25, 1998.

[Smyt 1998b] B. Smyth. Case-base maintenance. In Proceedings of the Eleventh International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE-98), vol.2, pp. 507-516, Castellón, Spain, June 1-4, 1998.

[Smyt 1999a] B. Smyth and E. Mckenna. Building compact competent case bases. In Proceedings of the Third International Conference on Case-based Reasoning (ICCBR-99), pp. 329-342, Monastery Seeon, Munich, Germany, July 27-30, 1999.

[Smyt 1999b] B. Smyth and E McKenna. Footprint-based retrieval. In Proceedings of the Third International Conference on Case-based Reasoning (ICCBR-99) pp. 343-357, Monastery Seeon, Munich, Germany, July 27-30, 1999.

[Smyt 2000a] B. Smyth and E. Mckenna. An efficient and effective procedure for updating a competence model for case-based reasoners. In Proceedings of the Eleventh European Conference on Machine Learning, pp. 357-368, Barcelona, Catalonia, Spain, May 29-June 2, 2000.

[Smyt 2000b] B. Smyth and E. Mckenna. Incremental footprint-based retrieval. In Proceedings of ES-00, Cambridge, UK, Spring-Verlag, Berlin, pp. 89-101, 2000.

[Smyt 2000c] B. Smyth and E. Mckenna. Competence guided instance selection for case-based reasoning. In Instance Selection and Construction: A Data Mining Perspective, H. Liu and H. Motoda (eds), Kluwer Academic, Boston, pp.1-18, 2000.

[Smyt 2001a] B. Smyth and P. McClave. Similarity vs. Diversity. In Proceedings of the Fourth International Conference on Case-Based Reasoning (ICCBR-01), pp. 347-361, Vancouver, Canada, July 30-August 2, 2001.

[Smyt 2001b] B. Smyth and E. Mckenna. Competence models and the maintenance problem. Computational Intelligence, vol. 17, no. 2, pp. 235-249, 2001.

[Tay 2003] F. E. H. Tay and L. Shen. Fault diagnosis based on rough set theory. Engineering Applications of Artificial Intelligence, vol. 16, no. 1, pp. 39-43, 2003.

[Tibs 2002] R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression. In Proceedings of the National Academy of Sciences (PNAS), vol. 99, no. 10, pp. 6567-6572, 2002.

[Tome 1976] I. Tomek. An experiment with the edited nearest-neighbor rule. IEEE Trans. Systems, Man, and Cybernetics, vol. 6, no. 6, pp. 448-452, 1976.

[Tver 1977] A. Tversky. Features of similarity. Psychological Review, vol. 84, no. 4, pp. 327-352, 1977.

[Utgo 1998] P. E. Utgoff and D. Precup. Constructive function approximation. In Feature Extraction, Construction and Selection: A Data Mining Perspective. H. Liu and H. Motoda (eds.). Kluwer Academic Publishers, pp. 219-235, 1998.

[Vapn 1998] V. N. Vapnik. Statistical Learning Theory. Wiley, New York, 1998.

[Vapn 1999] V. N. Vapnik. The Nature of Statistical Learning Theory. Springer, New York, 1999.

[Wang 1992] Z. Y. Wang and G. J. Klir. Fuzzy Measure Theory. Plenum, New York, 1992.

[Wang 1999] Z. Y. Wang, K. S. Leung, and J. Wang. A genetic algorithm for determining nonadditive set functions in information fusion. Fuzzy Sets and Systems, vol. 102, issue 3, pp. 463-469, 1999.

[Wang 2000a] Z. Y. Wang, K. S. Leung, M. L. Wong, J. Fang, and K. Xu. Nonlinear nonnegative multiregressions based on Choquet integrals. International Journal of Approximate Reasoning, vol. 25, issue 2, pp. 71-87, 2000.

[Wang 2000b] X. Z. Wang and D. S. Yeung. Using fuzzy integral to modeling case-based reasoning with feature interaction. In Proceedings of the 2000 IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC-00), vol. 5, pp. 3660-3665, Nashville, TN, USA, October 8-11, 2000.

[Wang 2001a] X. Z. Wang, D. S. Yeung, and E. C. C. Tsang. A comparative study on heuristic algorithms for generating fuzzy decision trees. IEEE Trans. Systems, Man and Cybernetics (Part B), vol. 31, no. 2, pp. 215-226, 2001.

[Wang 2001b] J. Wang and J. Wang. Reduction algorithms based on discernibility matrix: The ordered attributes method. Journal of Computer Science & Technology, vol. 16, no. 6, pp. 489-504, 2001.

[Wils 1972] D. R. Wilson and L. Dennis. Asymptotic properties of nearest neighbor rules using edited data. IEEE Trans. Systems, Man, and Cybernetics, vol. 2, no. 3, pp. 408-421, 1972.

[Wils 1997] D. R. Wilson and T. R. Martinez. Improved heterogeneous distance functions. Journal of Artificial Intelligence Research, vol. 6, pages 1-34, 1997.

[Xu 1999] M. Q. Xu, K. Hirota, and H. Yoshino. A Fuzzy theoretical approach to representation and inference of case in CISG. International Journal of Artificial Intelligence and Law, vol. 7, no. 2-3, pp. 259-272, 1999.

[Yang 1999] Y. Yang. An evaluation of statistical approaches to text categorization. Information Retrieval, vol. 1, no. 1, pp. 69-90, 1999.

[Yang 2001] Q. Yang and J. Zhu. A case-addition policy for case-base maintenance. Computational Intelligence, vol. 17, no. 2, pp.250-262, 2001.

[Yang 2003] Q. Yang, Y. Wang, and Z. Zhang. SANet: An intelligent service agent network for call-center scheduling. IEEE Trans. Systems, Man and Cybernetics (IEEE SMC Part A), vol. 33, no. 3, pp. 396-406, 2003.

[Yuan 1995] Y. Yuan and M. J. Show. Induction of fuzzy decision trees. Fuzzy Sets and Systems, vol. 69, no. 2, pp. 125-139, 1995.

[Zhu 1999] J. Zhu and Q. Yang. Remembering to add: competence-preserving case addition policies for case base maintenance. In Proceedings of the International Joint Conference in Artificial Intelligence (IJCAI-99), pp. 234-239, Stockholm, Sweden, July

31-August 6, 1999.

[Zupa 1998] B. Zupan, M. Bohanec, J. Demšar, and I. Bratko. Feature transformation by function decomposition. In Feature Extraction, Construction and Selection: A Data Mining Perspective. H. Liu and H. Motoda (eds.). Kluwer Academic Publishers, pp. 325-340, 1998.