Discernibility Concept in Classification Problems

Zacharias N. Voulgaris

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy for the University of London

May 2009

To my family

Abstract

The main idea behind this project is that the pattern classification process can be enhanced by taking into account the geometry of class structure in datasets of interest. In contrast to previous work in the literature, this research not only develops a measure of discernibility of individual patterns but also consistently applies it to various stages of the classification process.

The applications of the discernibility concept cover a wide range of issues from pre-processing to the actual classification and beyond that. Specifically, we apply it for: (a) finding feature subsets of similar classification quality (applicable in diverse ensembles), (b) feature selection, (c) data reduction, (d) reject option, and (e) enhancing the k-NN classifier. Also, a number of auxiliary algorithms and measures are developed to facilitate the proposed methodology. Experiments have been carried out using datasets of the University of California at Irvine (UCI) repository. The experiments provide numerical evidence that the proposed approach does improve the performance of various classifiers. This, together with its simplicity renders it a novel, useful and versatile tool for the classification process.

Acknowledgements

I would like to take this opportunity to express my gratitude to all those people who helped me, in their way, for the completion of this research.

First of all, I would like to thank my family, without whose invaluable support this project would have remained an unfulfilled dream. Through the past four years they showed great understanding, support and trust, despite the adverse circumstance that I found myself in. Also, their advice on general matters involving the project, and their genuine interest helped fuel my motivation to complete it, at times when the adversities were overwhelming.

I would also like to thank my supervisors, Dr. George Magoulas and Prof. Boris Mirkin, who showed patience, tolerance and great intelligence in their guidance through this mental journey that, due to their help, brought about a number of new skills, experiences and helped me develop the priceless know-how that made this project worthwhile. I am very grateful especially to Dr. Magoulas, for his encouragement through the difficult times of the research, as well as for his priceless advice, on all types of practical matters, throughout these years. His help at the writing up stage as well as during the defence viva is especially acknowledged.

This acknowledgements part would be incomplete if I were not to include Matina Karastatira, who helped me make this research project an enjoyable and quite easier task. Her patient understanding and support transmuted this period of my life to a great lesson of combining things and striking a healthy balance between professional duty and personal life. Also her support during the proofreading part of the thesis writing saved me a lot of time and effort, while it made the whole process of editing and enriching it more easily acceptable to me.

I would also like to extend my thanks to the various members of the academic staff here at Birkbeck, who with their inspiring example, helped me understand that this PhD is not an end-product, but a door to an interesting and creative life. Special thanks to Dr. Jenny Pedler and Dr. Ian Harrison for the invaluable opportunities they offered me in teaching and getting a clearer perspective of the responsibilities involved in such a task. Also, I would like to thank Dr. Eli Katsiri for her advice.

Special appreciation to my friends, especially Julian Scott and Sabine Leitner (for their support in many levels), Melissa Andrade, Dr. Kostas Filipopolitis, Dr.

George Themelis, Ken Tsang, Dionysis Dimakopoulos, Dr. Harris Alexakis, Dr. Rajesh Pampapathi and Dr. Aris Moustakas (for demystifying the post-PhD life for me). Also, I would like to thank Dr. Simon Coupland for his help at the last minute corrections of the thesis.

Furthermore, I would like to extend my sincere thanks to my examiners, Prof. Shen and Dr. Pontil. With their scholastic review of this thesis, many mistakes were pinpointed and I had the opportunity to make a number of clarifications in the text. Also, they drew my attention to the most essential part of this project and helped me gain a better understanding of the field. Moreover, they aided in my development as a researcher by showing me how to view my work from another researcher's perspective.

My final words of gratitude go to my family again, especially my brother Costas and his family, who maintained a more or less objective viewpoint providing me with great insight and a clearer perspective, when it came to strategic matters. Also, their presence in this country in the first years of my research made the first phase much easier. Also, I would like to express my gratitude to my godfather who made this project of strategic importance for me, offering the possibility of a bridge between academic qualifications and professional experience.

Table of Contents

List of Tables

List of Figures

Chapter 1 – Introduction

"Science is the systematic classification of experience"
– George H. Lewes (1817-1878)

Concept classification is one of the fundamental attributes of intelligence and finds application in many different aspects of our lives. Its introduction to Computer Science influenced fields like Pattern Recognition and Machine Learning. Yet, as our dealing with data becomes more and more complicated, our demands for an efficient and sustainable use of classification leads us to the development of (quite often) sophisticated methods for analysing and employing it for various types of predictions, may it be static or dynamic. Whatever the case, as predictions tend to come along with errors, it has been our task to develop classification systems that minimise these errors and provide more and more reliable predictions. This project constitutes one of the many attempts to make this task more understandable and improve its applications.

In general, classification as Pattern Recognition is "the act of taking raw data and taking an action based on the 'category' of the pattern" (Duda et al, 2001). The applications of classification include automated procedures for sorting letters based on post-codes, assigning individuals to credit status, as well as providing a preliminarily disease diagnosis while waiting for definitive test results (Mitcie et al., 1994). Due to the broad spectrum of applications, classification has attracted the interest of many researchers and many techniques for it have been developed.

Yet, simple as the problem may sound, its treatment often involves sophisticated methods in order to accomplish an accurate and reliable classification. This is one of the main reasons that spawned the development of a number of information systems focused on this particular activity using various approaches. Nevertheless, the underlying process remains largely the same in all approaches: typically a pre-processing stage takes place, which prepares the data in a way that they are more suitable for the classifier used (this may involve segmentation and/or feature selection/extraction). Then, the training of the classifier takes place, where the patterns of a particular part of the dataset (the training set) are used. Afterwards, in some cases, the cost of classification is taken into account. Finally, the actual classification takes place, rendering the unlabeled patterns into the predefined classes.

Another point of view regarding classification is that put forward by McKenzie and Low (1992). According to them, classification is the application of certain rules, which are defined by the process of discrimination. Yet, these rules are often in a form that is not accessible or comprehensible to us, since they depend on the structure of the classifier used.

In all cases, classification systems need to overcome a number of difficulties. The most important of these is the over-fitting problem. This involves learning the training pattern in such a rigid way that the classifier is unable to perform well on novel patterns. The classifier needs to be sophisticated enough so that it can distinguish the difference among the different classes, but simple enough so that it can classify novel patterns accurately (Duda et al, 2001; Vapnik, 2000a).

1.1 Motivation of this Project

There has been a great deal of research on classification systems, aiming at the development of general methods and specialised techniques to tackle particular classification problems. These techniques often employ the statistical properties of the data involved (e.g. McKenzie & Low, 1992; Jain et al., 2000), or adaptive mechanisms (e.g. the classifier described by Wang et al., 2007) to exploit every little piece of information that may reveal a useful property of the data, leading to a reliable classification.

Yet, datasets not always follow the assumed statistical distribution; this renders the statistical classifiers unable to cope with the problems and reduces their reliability. Also, many classifiers tend to become "confused" when dealing with large datasets, as the excessive information that is there often compromises the classifiers' performance, sometimes due to the problem of over-fitting. This can be attributed to the fact that they often consider all patterns being equally important, instead of taking into account their underlying structure which may yield more useful information for the classification process.

Motivated by this, we decided to view the problem anew, focusing on first understanding and evaluating what we know, and then tackling the pieces of data that are unknown, trying to discern which class is more appropriate for their classification. That is why we developed and proposed a new measure for investigating and evaluating the structural properties of the dataset in hand, without any *a priori* information about it. This we coin as Discernibility, an insightful tool to facilitate and enhance the classification process, regardless of the dataset or the classifier used.

1.2 Aim

In this project we aim to investigate how the various aspects of classification, namely features selection, data reduction, reliability evaluation, and information fusion can be improved by employing a new perspective to it through the use of the Discernibility concept. Discernibility is a novel concept introduced in this research, aspiring to clarify the structural properties of a dataset, in both a pattern and dataset level.

1.3 Objectives

There are several objectives that we plan to fulfil in this project in order to achieve our aim. These are the following:

- Introduce the new concept of Discernibility and identify relationships with other similar metrics
- Demonstrate how the proposed concept functions and how it can be applied in the classification process, particularly for improving the kNN classifier
- Develop feature selection and data reduction techniques based on the Discernibility concept
- Develop new methods for measuring reliability for different classifiers, as well as their combination
- Explore how Discernibility can be incorporated to enhance the performance of a diversity ensemble (a set of classifiers aiming to yield an improved performance by basing its structure on the diversity of errors of its members)

1.4 Methodology

The philosophy of this project is to investigate the newly introduced concept of Discernibility and explore its various applications in classification problems. The objectives presented in Section 1.3 will be accomplished by employing a diverse methodology, centred however around this philosophy.

As regards the first objective, a descriptive methodology will be employed, first explaining the idea and the implementations of it, then describing the details of the algorithms developed, and finally comparing them with the existing measures that share the same general approach.

Regarding the second objective, the function of Discernibility in the classification process will be shown by examining how the kNN classifier is enhanced by incorporating this notion in its function. For this purpose, a number of benchmark datasets will be used and an experimental comparison with the original kNN classification algorithm will be conducted. The findings of this round of experiments will then be presented and discussed.

Concerning the feature selection and data reduction applications of Discernibility, a literature review will be conducted and the shortcomings of the traditional methods will be pinpointed. Afterwards, some techniques based on Discernibility will be presented and it will be shown how they attempt to address these problems. A number of experiments on some benchmark datasets will be carried out to demonstrate their function and performance.

With regards to meeting the reliability objective, firstly a literature review will be carried out pinpointing the limitations of the probability-based methods. Afterwards, the Discernibility-based approach will be presented and it will be described how it attempts to tackle these shortcomings. Furthermore, the findings of the relevant experimentation will be exhibited and discussed.

The final objective, related to the application of Discernibility in classifier ensembles, will be accomplished by first reviewing the relevant literature on ensembles, concentrating on one particular type (as this field is vast and would demand an excessively larger research to do it justice). This method will then be explained and its limitations pinpointed. Then, it will be shown how (and why) by employing Discernibility this method can be improved. Afterwards, the experiments involved will be conducted and their results analysed, putting forward the comparison of the different variations of the method and the improvement due to Discernibility.

1.5 Thesis Structure

This PhD thesis comprises eight chapters. In Chapter 2 a review of the general literature of the field is conducted. This serves as an introduction to Pattern Recognition and a taxonomy of the various AI techniques developed for it.

Afterwards, in Chapter 3, the core concept of this project, Discernibility, is presented and its relation with similar measures is discussed. This chapter describes how discernibility operates through the use of some examples and examines its relationship with other measures, namely the SOM-based Class Overlap Degree

Coefficient (Lemeni & Tepus, 2008) and the Silhouette Width (Kaufman & Rousseeuw, 1990). It presents two forms of discernibility: the spherical index and the harmonic index and discusses their role and potential as tool in the classification process.

Chapters 4 to 7 are devoted to applying the concept of Discernibility to various parts of the process of classification. Specifically, in Chapter 4 the application of Discernibility in the classification process is shown. Particularly, it is demonstrated how kNN variations based on Discernibility can be developed, and their performance is investigated by comparing them against the classical kNN classifier.

Chapter 5 describes how the Discernibility concept can be applied in the pre-processing stage of classification. Namely, it is shown how Discernibility can be employed for the development of feature selection and data reduction techniques. Moreover, the effectiveness of the proposed techniques is demonstrated through various experiments.

The possibility of a reliability measure and its application as a reject option are discussed in Chapter 6. After providing a literature review on the subject, an alternative method for measuring reliability is proposed. Then, its application on a number of different classifiers as well as on their combination is exhibited.

Afterwards, in Chapter 7 the possibility of incorporating Discernibility in diverse ensembles, in order to improve their accuracy rate, is explored. After thorough experimentation, a new automated approach to improve the current one is proposed and discussed.

Finally, in Chapter 8, the conclusions based on the findings of this research are presented, along with the avenues of future research that are opened by this project.

1.6 Contribution of the Thesis

The field of Pattern Classification is quite broad and several valuable contributions have been made so far to improve the effectiveness of the classification process. This thesis introduces and explores the Discernibility concept investigating the various forms it can take. It demonstrates how versatile this concept can be, and how its application in the various classification stages can enhance the classification process.

The thesis reviews (Chapter 2) the various classification systems that are used today, focusing on statistical and the AI-based classifiers, and discusses the conditions under which the latter are preferable to the former in terms of flexibility with respect to the data and assumptions made about data distribution.

The thesis describes in detail (Chapter 3) the concept of Discernibility and shows the different forms it can take, emphasising on the Spherical Index of Discernibility, which appears to be closer to one's intuitive understanding of the notion. This concept allows the development of a variety of straight-forward methods that can make the classification process better and faster. The thesis also introduces two measures namely the Degree of Certainty and Net Reliability that can be applied in a variety of cases and are independent of the classifiers use as well as the datasets.

The contribution Discernibility can make to the field of Pattern Classification is investigated through the various Discernibility-based classifiers that are developed based on the kNN philosophy (Chapter 4). The effectiveness of classifiers, which are equipped with the Index of Discernibility, is demonstrated, while the experiments carried out show that these classifiers generally outperform kNN. This part of the thesis also contributes a few alternative approaches to improving the kNN classifier and also proposes another distance-based classifier similar to kNN, namely the Minimum Spanning Tree classifier. Moreover, the advantages and limitations of the proposed methods are discussed.

The thesis also makes another contribution by showing how Descernibility can be used in order to tackle some open problems in the classification literature offering innovative approaches to feature selection and data reduction. In particular, the thesis (Chapter 5) shows how the Spherical Index of Discernibility can be used without major alterations to develop feature selection techniques. This leads to contributing two feature selection methods: one operates by evaluating each individual feature of a dataset and then selecting the best ones while the other by taking different groups of features (feature sub-sets of the dataset) and comparing their discernibilities with that of the original feature set. The results of the corresponding experiments are encouraging showing that these methods are quite effective in reducing the features of a dataset without compromising the classification performance, both in terms of accuracy rate and in CPU time.

The thesis also contributes a data reduction method that employs the Discernibility concept (Chapter 5). It shows that by taking into account the discernibilities of the various patterns as well as the distances among them, we can remove the ones which are discernible and distant (i.e. their average distance in the data space is large), thus ending up with a smaller dataset, which however maintains the structure of the original one. This is mirrored by the performance of a number of different classifiers, which (for most of them) appears to be at the same level or might even get better when using the reduced data set.

The thesis then proceeds by demonstrating how the Discernibility concept can be applied to measure the classification reliability of a single classifier and of an amalgamation of classifiers (Chapter 6). Also, it shows how it can be elaborated into a parametric reliability measure taking into account other factors, particularly the change of accuracy rate and the change of Degree of Certainty. Our hypothesis that Discernibility can be a useful reliability metric, especially if used in combination with the other two factors, is verified by the quite encouraging experimental results.

In that respect, the thesis also contributes in the field of Pattern Classification by offering a new measure of reliability for the classification process that does not depend on probabilities or assumptions about the distributions of the dataset. The flexibility that this characteristic offers allows the proposed reliability measure to be more applicable to classification problems, particularly when the dataset is relatively small. Also, this part of the thesis describes how with practically no changes in its function, Discernibility can be applied to an entirely different domain, providing useful information on the unknown patterns, *a priori*. Considering that Discernibility was developed mainly to tackle situations where all the patterns were *known* (labelled), it is interesting how it can be easily adapted to deal with *unknown* ones as well. This is significant as it opens the possibility of a reject option, which can enhance the classification process.

Furthermore, the thesis explores how Discernibility can support the generation of diverse ensembles, which is a popular approach to combine classifiers, by introducing a novel approach for partitioning the feature set (Chapter 7). This approach involves an ensemble of classifiers that are specialised on different feature subsets of the dataset, forming a diverse feature set. The advantage of the proposed method is that, because of the use of Discernibility, the feature subsets are more or less of the same classification quality; therefore this method allows creating ensembles whose members possess adequate generalisation potential. The aim of this part of the thesis is not to give a definite solution to the problem of creating diverse ensembles – an area of Pattern Recognition where there is a lot of ongoing research – but to offer a new perspective to this problem which might be potentially useful for developing alternative approaches for generating diverse ensembles.

Chapter 2 – Review of Classification Literature

A number of classification systems have been developed, often specialising in particular problem domains or dataset types (Duda et al., 2001). Their performance is evaluated according to their ability to classify novel patterns, based on the ones they are trained with. The most popular measure that reflects this is accuracy (usually expressed as a rate), which is defined as the number of correct classifications over the total number of patterns classified.

When developing classification systems there is an underlying tendency towards simpler models, as the overly complex ones may compromise the accuracy of the classification (Duda et al., 2001) while at the same time exhibit higher computational cost. Therefore, the dominant approach for many years was to use (often basic) statistics and develop what are now known as the Statistical Methods of classification. Yet, relatively recently, another approach has been developed, namely, that using Artificial Intelligence (AI) methods. These two types of approaches are the two main categories of pattern recognition (McKenzie & Low, 1992).

Statistical Methods include Linear and Quadratic Discriminant Functions (LDFs and QDFs respectively), Bayesian classifiers, Parzen Classifier, Logistic Classifier, etc. (McKenzie & Low, 1992; Jain et al., 2000). The characteristic which appears in the majority of them is their use of probabilities for the estimation of the most suitable class of the unknown patterns using the assumed probability density functions based on the known patterns.

The main difference between statistical and AI methods is that the former make assumptions of the data (such as the distribution they form), while the latter make less strict assumptions (Montgomery et al., 1997; Shanin et al., 2001) However, since there have been developed a lot of classifiers which take elements of both approaches, the boundaries between Statistical and AI classification methods are not clear-cut.

Another important issue is that of noise, which influences the data values by making the structure of the classes appear to be more irregular. Though its presence is often a hindrance to statistical classifiers, AI based classifiers often deal with it effectively and sometimes it even helps them make better generalisation (e.g. in the case of Neural Networks), as argued by Duda et al. (2001).

Lastly, a significant issue for classification systems is that of computational complexity. Even though in some cases it may be possible to attain an error-free classification by using a vast amount of resources (particularly time and storage), this is often not practical (Duda et. al, 2001). Many AI methods are developed so that they are "lighter" and therefore perform the classification process in a fast and efficient way, something which is not usually shared by the statistical methods. They are like that because they often rely on heuristic measures or methods for the classification process.

2.1 Artificial Intelligence Methods for Classification

The Artificial Intelligence methods used in classification vary in their function and their structure. The most popular ones are:

- Support Vector Machines (SVMs)
- Decision Trees
- Artificial Neural Networks (ANNs), such as
    - Multi-Layer Neural Networks or Multi-Layer Perceptrons (MLPs)
    - Radial Basis Function networks (RBFs)
    - Self-Organising Maps (SOMs)
    - Probabilistic Neural Networks (PNNs)
- K Nearest Neighbour (kNN) and its variations
- Other distance-based classifiers
- Fuzzy Logic classifiers
- Stochastic methods (according to Duda et. al (2001) this type of classifiers belongs to the AI category, even though they have many Statistical elements in their function)
- Hybrid classification systems

Apart from these classifiers, there are also approaches which aim at combining different classifiers, either of the same kind, or, quite often, of different philosophy such as *mixture of expert models*, *pooled classifiers*, or *classifier ensembles* (Ruta & Gabrys, 2005; Shipp & Kuncheva, 2002). These methods employ an information fusion technique (e.g. majority vote (Lam & Suen, 1997)) to combine the outputs of the different classifiers they comprise of (ensemble members). In some cases, the combination of the members of the ensemble takes place in a lower level, particularly that of the training phase (Fard, 2006). The classifier ensembles can have statistical methods (such as Linear and Quadratic Discriminant Functions, Bayesian classifiers, Parzen Classifier, Logistic Classifier, etc.) as their members, yet they usually include one or more AI methods. In the remainder of the section a review of the AI techniques will be given.

2.1.1 Support Vector Machines

Most of the AI methods have their own advantages and exhibit an edge in particular problems or dataset types. SVMs for example perform quite well in two-class problems.

SVMs perform classification by finding a hyperplane that separates the two classes of the dataset in the best possible way (i.e. having a relatively high margin). They form decision rules based on that hyperplane and perform classification accordingly. In order to accomplish this, they make use of a usually much higher dimensionality than that of the original feature space. This is because with the right nonlinear mapping to a high enough dimension, patterns from two different classes can always be separated by a hyperplane. Yet, the boundaries they form in this space can be non-linear as well, depending on the kernel parameter of the SVM (Vapnik, 2000b). By finding a separating hyperplane with the highest possible margin (i.e. distance from actual patterns of the dataset), SVMs provide the generalisation they require for the classification task (the higher the margin, the better the generalisation).

Although SVMs were primarily able to tackle two-class classification problems, with the appropriate adjustments they are able to deal with multi-class datasets (Cristianini & Shawe-Taylor, 2000). Although SVMs are quite good at tackling various types of problems, often the large number of parameters involved makes them difficult to fine-tune and often their performance is compromised if the user is not knowledgeable of the optimum choice of parameters, even though they are only two of them.

Support Vector Machines and other classification methods that share the same philosophy are thoroughly described in Vapnik (2000b).

2.1.2 Decision Trees

Decision Trees form a number of queries based on one or more of the attributes of the dataset. Based on these queries, which depend on the data type and on the percentage of patterns that are considered enough to form a rule (usually if 90% of the patterns at a given node comply with the rule, it is accepted). The rule of thumb for creating decision trees is simplicity, since cumbersome trees may be too complicated resulting to the problem of over-fitting.

Decision Trees comprise of a number of nodes, where certain “checks” take place, guiding the classifier’s decision about what pattern belongs where. The top-most part of the tree is called the root node and the lowest part is made up of the so-called leaf-nodes. The various nodes of the tree are connected with one or more of the other nodes via certain lines called links.

Decision Trees classify a given pattern starting at the root node, where the value of a particular property of the pattern is checked. The various links stemming from the root node correspond to the various possible outcomes of this check. Based on the outcome the appropriate link is followed to a descendent node. Usually only one link will is followed, although there are other types of Decision Trees where this is not the case. Following, the decision is made at the sub-tree appropriate subsequent node, which can be seen as sub-tree’s root. This process is repeated until a leaf node is reached, where there is no further check and where the class label is assigned to the pattern.

Decision Trees exhibit fast performance and can tackle a variety of problems, including dataset with nominal data. Also, they can easily incorporate expert knowledge in their structure and their decisions are easy to interpret (Duda et al, 2001). Lastly, their performance is not compromised by the existence of missing values or errors in data (since not all attributes are used at the same time), and are particularly useful when dealing with noisy data or when disjunctive expressions are involved (since these do not affect the rules that the Decision Trees form) (Mitchell, 1997; Winston, 1992).

Since they first appeared (Winston, 1969) they have evolved significantly and adopted more elaborate approaches for their construction. CART (Classification and Regression Trees) was one of the most well-known Decision Trees, which due to its high computational cost, its limited use of features (only single ones and linear combinations of them were considered at each node) as well as its inability to create an optimal sub-tree (Safavian & Landgrebe, 1991) rendered it less popular. ID3 is another well-known approach which is designed to deal with nominal data only (Duda et al., 2001). Its refined version which succeeds it, the quite popular C4.5,manages to deal with real-valued variables as well, and makes use of statistical significance for obtaining the optimum tree structure. As C4.5 is used in many of the experiments in this research, its function is described more thoroughly (see Algorithm 2.1).

Algorithm 2.1 – Pseudo-code of C4.5 classifier
Inputs: Input patterns of training set (P), labels of training set (T), number of neighbours (*k*), patters of testing set (PT)
Outputs: classification vector of testing set (y)

```
1  Initialisation: Set $i = 0, j = 0, th = 0.10$ (proportion of alien entities in a node)
2  N ← number of data elements in P
3  na ← number of attributes in P
4  do $i \leftarrow (i + 1)$
5    find normalized information gain ($G_i$) from splitting on attribute $i$, so that no more than $th$*N of
     patterns from different classes exist in each split
6  until $i$ = na
7  $i\_best \leftarrow i$ value for which $G_i$ is maximised
8  create a decision node than splits on $i\_best$
9  repeat steps 4-8 for all cases obtained by splitting on $i\_best$ and add the additional nodes created
   as sub-nodes of node
10 do $j \leftarrow j + 1$
11   classify pattern $j$ based on decision tree created previously
12 until $j$ = n
```

2.1.3 Artificial Neural Networks

An Artificial Neural Network (ANN) consists of a number of nodes, called neurons, usually organised in layers, which can handle incoming signals (input values of the dataset or outputs from other neurons) by means of a transfer function. The latter can be any monotonous function yielding values between -1 and 1, although in most cases it is the sigmoid function (Cybenko, 1989). Auxiliary inputs called biases are also used throughout the network. The strength of each signal and the biases are represented by weights and constants, which are estimated through the training phase. For the latter there is a number of training algorithms used, among which backpropagation-based methods (Rumelhart et al., 1986) are the most popular; these employ the gradient descent algorithm (Valafar & Ersoy, 1994).

Originally, ANNs started in the form of a single neuron, proposed in the McCulloch and Pitts model in the 1940s (McCulloch & Pitts, 1943). Yet, it was in the 1960s when they attracted some serious research interest (starting with Rosenblatt's perceptron in 1962). In the 1980s, they became more popular and great advances took place in their field (Hopfield networks came about with Hopfield's research

(Hopfield, 1982), while at around the same time the Kohonen Self-Orginising Maps appeared (Kohonen, 1982). Also, in the same decade, the Back-propagation learning algorithm was rediscovered and developed further (Rumelhart et al., 1986). In the 1990s Radial Basis Function ANNs were developed (Niranjan & Fallside, 1990; Musavi et al., 1992), while in the 2000s, the creation of ensembles of ANNs has attracted a lot of research interest (Jiang & Zhou, 2004; Brown, 2004).

ANNs can model quite complex datasets and establish classification rules that evade other classifier types. Also, their performance tends to be insensitive to noise, which in some cases improves their generalisation (Sugiyama & Ogawa, 2000). Yet, in certain cases the generalisation can be enhanced by eliminating the noisy patterns from the training set, in order to avoid overfitting (Nakashima & Ogawa, 2000).

Multilayer perceptrons (MLPs), which can be trained using a backpropagation method, is a very popular choice for many researchers. Radial Basis Function networks (RBFs) are preferable when the data form clusters (Roy, Govil, & Miranda, 1995; Kaylani & Dasgupta., 1994) because of the way they handle the data they use. Also, RBFs have a number of properties, such as localisation, interpolation, approximation of functions and cluster modelling, making them suitable for a number of applications in a variety of fields (Bors, 2001).

Lastly, the Self-organising Maps (SOMs) are quite handy when there is a need for visual representation of the solution obtained (Kohonen, 2001; Carpenter & Grossberg, 1991) since we can visually examine the structure of their neurons and connections among them on two-dimensional space. In general ANNs generate a complex set of rules, which are not visible to the user. Yet, there have been developed systems for extracting knowledge from an ANN that has already been trained.

Probabilistic Neural Networks (PNNs) are a special case of ANNs, based on Parzen windows (a method for estimating the univariate normal density of probabilities, which is a function showing the structure of a distribution consisting of a single variable). PNNs are exceptionally fast, since their training phase requires only one pass through the training patterns. However, the storage they require is relatively high. Also, the new patterns classified can be easily incorporated into another classifier (already trained), rendering PNNs a good alternative for on-line applications (Specht, 1990). In other words, their output can be later processed by another classification system, to improve the overall performance, and as this happens very fast, PNNs are suitable for on-line applications where a real-time classifier is required.

2.1.4 The k Nearest Neighbour and Other Distance-Based Classifiers

The k Nearest Neighbour method classifies an unknown pattern based on the ones in close proximity to it (the k parameter is the number of these proximal patterns). Although some researchers consider it a statistical classifier (Jain et al., 2000), the absence of any consideration of statistical metrics in its function renders it an independent classifier. Also, many of its variations make use of AI techniques (e.g. Fuzzy k Nearest Neighbour). Therefore it fits better the profile of the AI methods.

The kNN classifier, originally put forward by Cover & Hart (1967), is very fast and with the right choice of the number of neighbours (k), it usually yields good results (though it is suboptimal, compared to the Bayesian classifier (Duda et al, 2001)). Also, because of its simplicity, it has found a number of applications in various fields, such as image analysis, spatial data processing, etc. Its inherent weaknesses, namely its suboptimal performance and its inability to tackle data of high dimensionality effectively, have been a subject of ongoing research which has led to the development of a number of kNN variants. A special case of kNN is the Nearest Neighbour classifier (1NN), where only the closest pattern is taken into account. The pseudo-code of kNN can be found in Algorithm 2.2.

Algorithm 2.2 – Pseudo-code of k Nearest Neighbour classifier
Inputs: Input patterns of training set (P), labels of training set (T), number of neighbours (*k*), patters of testing set (PT)
Outputs: classification vector of testing set (y)

```
1  Initialisation: Set $i = 0, j = 0$
2  N ← number of data elements in P
3  n ← number of data elements in PT
4  q ← number of unique values of T (classes)
5  do $i \leftarrow (i + 1)$
6     calculate distance array (d) based on the Euclidean distances (Eq. 2.2) between test pattern $i$
      and each one of the training patterns
7     sort distances and store indexes (ind)
8     get $k$ nearest patterns based on the first $k$ values of ind
9     do $j \leftarrow j + 1$
10       count number of neighbours that are labelled as class $j$ ($m_j$)
11    until $j = q$
12    Q ← $j$ value for which $m_j$ is maximised
13    classify pattern $i$ to class Q ($y_i$ ← class Q)
14 until $i = n$
```

One interesting variation of kNN, which is used in the experiments involved in this research is the Fuzzy kNN (Keller et al., 1985). This classifier makes use of a function of the distance of each pattern to each neighbour. These functions take the form of weights which are calculated using Eq. 2.1.

$$w(i) = \frac{1}{d_{ij}^{(m-1)}} \tag{2.1}$$

where *w(i)* is the weight of pattern *i*,

d_{ij} is the distance between pattern *i* and neighbour *j*

m is a parameter, having a default value of 2

After normalising these weights, they are used in combination with the class labels of the neighbours yielding the classification output. Generally, the performance of Fuzzy kNN is more robust than that of the classic kNN classifier, especially when there is a high class overlap (Yu et al., 2002). The function of Fuzzy kNN can be viewed in Algorithm 2.3

Algorithm 2.3 – Pseudo-code of Fuzzy k Nearest Neighbour classifier
Inputs: Input patterns of training set (P), labels of training set (T), number of neighbours (*k*), patters of testing set (PT)
Outputs: classification vector of testing set (y)

```
 1 Initialisation: Set $i = 0$, $m = 2$ (fuzzy parameter)
 2 N ← number of data elements in P
 3 n ← number of data elements in PT
 4 q ← number of unique values of T (classes)
 5 transform T into a q x N binary matrix (consisting of a row for each pattern, in which there is a 1
   in the column corresponding to the class it belongs to, and 0 in all other places)
 6   do $i \leftarrow (i + 1)$
 7   calculate distance array (d) based on the Euclidean distances (Eq. 2.2) between test pattern $i$
     and each one of the training patterns
 8   sort distances and store indexes (ind)
 9   get $k$ nearest patterns based on the first $k$ values of array ind
10   set weights array (w) equal to membership function based on Eq. 2.1
11   if a value of w is infinite then replace this value with 1
12   calculate memberships: $M \leftarrow T * w^T / \Sigma w$ for all the neighbours (based on ind array)
13   Q ← $j$ value for which $M_j$ is maximised
14   classify pattern $i$ to class Q ($y_i \leftarrow$ class Q)
15 until $i = n$
```

Other distance-based classifiers, often having an inherent similarity to the kNN method, have been developed for pattern recognition problems. Two of these are the Gravity Model Classifier, which is based on the paradigm proposed by Ruta & Garbys (2003), and the Reduced Coulomb Energy Networks (thoroughly presented in Jeon et al., 2002). For these classifiers, as well as for the kNN, the most popular distance measure used is the Euclidean distance, although other metrics have been considered. The Euclidean distance is defined as:

$$d(x,y) = \sqrt{\sum_{j=1}^{m}(x_i - y_i)^2}, \tag{2.2}$$

where *d(x,y)* is the distance between two patterns *x* and *y*, and *j* denotes the *j*-th feature value while *m* the total number of features.

The gravity model classifier is based on models from physics and applies them in classification. This classifier takes into account the “pull” from all the patterns of each class and as this greatly depends on their distance, the different “forces” involved pull the unknown pattern into one or the other direction (class). The class yielding the greater “gravitational pull” to the unknown pattern “wins” the classification. Naturally, the greater the number of patterns near the unknown pattern, the stronger the pull, so the nearest neighbours influence the classification decision more. The main advantages of this method are its high speed and the fact that it does not require any parameters. Its function is shown in Algorithm 2.4.

The Reduced Coulomb Energy (RCE) network is similar to a Probabilistic Neural Network, although their function is more similar to the kNN method. RCE assumes a fixed radius λ around each pattern. During the training phase, each λ is chosen to be as large as possible so as not to contain any pattern from different classes. If however this parameter becomes too small, this may result to the classifier being unable to classify a given pattern (i.e. yields an “ambiguous” label). Nevertheless, just like kNN, RCE yields high speed and has low memory requirements. (Jeon et al., 2002). Its function is presented in Algorithm 2.5 (note that as the “ambiguous” label is not useful practically, in this version of the method we have replaced it with a random classification).

Algorithm 2.4 – Pseudo-code of Gravity Model classifier
Inputs: Input patterns of training set (P), labels of training set (T, patters of testing set (PT)
Outputs: classification vector of test set (y)

```
 1 Initialisation: Set $i = 0, j = 0, k = 0$
 2 N ← number of data elements in P
 3 n ← number of data elements in PT
 4 q ← number of unique values of T (classes)
 5 do $i ← (i + 1)$
 6   $f_i$ ← patterns of training set which belong to class $i$
 7 until i = q
 8 do $j ← j + 1$
 9   compute distance array (d) between test pattern j and all patterns of training set
10   calculate “force of attraction” array ($F = 1 / d^2$)
11   for k ← k + 1
12     calculate total gravity forces of class k ($FC_k$) by adding the elements of F that belong to $f_i$
13   until $k = q$
14   compute total gravity force of class k: $FT_k$ = sum of all elements of $FC_k$ array
15   Q ← $k$ value for which $FT_k$ is maximised
16   classify pattern j to class Q ($y_j = Q$)
17 until $i = n$
```

Algorithm 2.5 – Pseudo-code of Reduced Coulomb Energy classifier
Inputs: Input patterns of training set (P), labels of training set (T, patters of testing set (PT), maximum radius parameter (λ_m (default value = 0.5))
Outputs: classification vector of test set (y)

```
 1 Initialisation: Set $i = 0, j = 0$, L = 0 ($\lambda$ parameter), e = 0.0001 ($\varepsilon$ parameter), D = {}
 2 N ← number of data elements in P
 3 n ← number of data elements in PT
 4 do $i ← (i + 1)$
 5   calculate distance array based on the Euclidean distances (Eq. 2.2) between test pattern $i$
     and each one of the training patterns and sort them (d)
 6   find nearest point which is not in class of pattern $i$ (X)
 7   calculate lambda for pattern i: $L_i = \min ( d_X - e , \lambda_m )$
 8   until $i = N$
 9 q ← number of unique values of T (classes)
10 do $j ← j + 1$
11 calculate distance of pattern $j$ of testing set, to all patterns of training set (d)
12   find all patterns whose distances are smaller than L and add them to set D
13   if there are no patterns close enough then classify pattern $j$ randomly
14   if the label of all patterns belonging to D is the same then classify pattern $j$ accordingly ($y_j$ =
     label of patterns in D)
16   until $j = n$
```

2.1.5 Fuzzy Logic Classifiers

Fuzzy Logic classifiers are classification systems that make use of fuzzy sets or fuzzy logic (Kuncheva, 2000), converting real-world data values into membership degrees through the use of the so-called "membership functions" (Zadeh, 1965), thereby forming high level rules that are then used for the classification process. This is done by defining "categories" (a concept different to classes, as in this case it refers to different values or intervals of values) for each one of the attributes. In cases where there is limited precision in the data values, or when classification is required in real-time, fuzzy logic classifiers are preferable to all the other methods (Siler & Buckley, 2005), due to their very high speed. Also, their decisions are easily interpretable by the user and expert knowledge on a particular domain can be used directly for them. Their main problem is the "curse of dimensionality", which renders these classifiers inadequate for problems having a large number of features. Also, they behave poorly on complex problems, while there is a limited amount of knowledge that the designer can incorporate in the system (if n is the number of fuzzy categories used, and m is the dimensionality of the problem, the number of possible rules that can be created is n^m, rendering it a very time-consuming problem. Typically n = 3). As the fuzzy logic classifier is trained by partitioning the dataset to n parts for each one of the m features, this slows down the classification process significantly (Babuska, 1998). Finally, pure fuzzy logic systems do not make use of the training data, something that gave rise to hybrid fuzzy systems, as we will see later on.

Although Fuzzy Logic first appeared in the 1960s, with the innovative research of Zadeh (1965), its application in Classification came about much later, in the 1980s. The first work on the field was that of Takagi & Sugeno (1985). Since then, Fuzzy Logic in Classification has attracted a lot of interest, rendering it a popular choice for a variety of applications, directly or indirectly related to Classification. For example, it has been successfully used in feature selection (Shen & Jensen, 2008; Shang & Shen, 2006; Jensen & Shen, 2005) rendering Fuzzy Logic a very useful tool for pre-processing.

2.1.6 Stochastic Methods

Stochastic methods constitute an approach focusing on complex and real-world problems, where the analytical methods often fail to produce adequate results. They can be classified into two main categories: the methods inspired by ideas and models from physics, particularly statistical mechanics, and the methods inspired by ideas from biology and specifically from the mathematical theory of evolution (Michalewicz, 1995; Michalewicz, Schoenauer, 1996). The former is more established theoretically and constitutes the majority of the stochastic models of pattern recognition. The latter is more heuristic and offers more flexibility; it has attracted an increasing number of researchers over the years and appears to be quite promising. Stochastic methods of the first category include Simulated Annealing and Botzmann Factor / Learning (Aarts & Korst, 1990). Some of the methods of the second category are Genetic Algorithms, Particle Swarm Optimisation (PSO) (Kennedy & Eberhart, 1995; Clerc, 2006; Poli, Kennedy, Blackwell, & Freitas, 2008), Ant Colony Optimisation (ACO) (Dorigo et al., 1991; Dorigo & Blum, 2005; Dorigo & Stützle, 2004), and other nature-inspired models, such as Bee Colony Optimisation (BCO) (Karaboga, 2005; Pham et al, 2006). Note that most of these stochastic methods have applications in other fields of computer science, such as complex optimisation problems, function approximation, etc.

2.1.7 Hybrid Classification Systems

Hybrid classification systems enable a self-management of resources and structural inputs (Cung et al, 2006) and combine the strengths of two or more AI algorithms. For example, the PSO method is known to perform well with continuous features, while it fails to function with nominal data for instance. By combining it with the ACO method, it is possible to tackle problems which involve nominal features as well, aiming at finding the optimum classification rules (Holden & Fietas, 2007). Another example of a hybrid classification system is the widely known neuro-fuzzy classifiers (Bezdek et al, 1992; Castellano, Fanelli, & Mencar, 2004), which incorporate fuzzy logic in the neurons of an MLP. Finally, there have been attempts of combining different types of ANNs to form a hybrid ANN which can tackle highly complex problems. Such a system is the Parallel Probablistic Self-Organising Hierarchical Neural Netwrok (PPSHNN) which was developed by Valafar & Ersoy (1994).

There is often a distinction between hybrid classification systems and ensembles, although sometimes this is not clear, as in the case of Cung et al. (2006), where the definition given for a hybrid system is broad enough to include many types of classifier ensembles. Yet, often the classifier ensembles include a greater number of classifiers; these classifiers sometimes are of the same type.

The main advantage of the hybrid classification systems is that they are versatile and often capable of adapting since they are designed to deal with a variety of problems and apply a different approach to each one of them. Also, they are able to tackle classification problems that would not be feasible by traditional classification methods. Particularly the more sophisticated ANNs (PPSHNNs) are able to function effectively on problems of high dimensionality and/or high non-linearity (Valafar & Ersoy, 1994).

2.2 Summary and Contribution of the Chapter

In this chapter we gave a brief presentation of the different types of classification systems, namely the statistical and the AI-based classifiers. In addition, we examined how the latter are in many ways preferable to the former due to their more flexible approach to the data, involving less strict assumptions about their distribution. Moreover, the specific characteristics of each one of the main AI-based classification systems, as well as the ensembles of classifiers, were described based on the relevant literature. Furthermore, we elaborated on their differences and the main advantages they exhibit, as well as their main weaknesses.

Chapter 3 – The Concepts of Discernibility and Data Quality in Classification Problems

In this chapter the concept of Discernibility is defined and the two Indexes of Discernibility are introduced. The term Discernibility involves the concept of discerning between (or among) the different classes of a dataset. So a dataset having well-separated classes generates high Discernibility. Also, this concept can be applied for the patterns of the dataset, in the same way. This notion has been implemented in a variety of ways, two of which stand out in terms of performance and speed, namely the spherical and the harmonic Indexes of Discernibility, and are described below. The others are earlier forms of these Indexes and attempt to express the same idea but with different geometric forms.

3.1 Spherical Index of Discernibility

One way to define a measure for assessing Discernibility is to take a pattern *i* and use of a hyper-sphere around it so as to examine how many other patterns of its class fall within that area. We assume a fixed radius around each element of the dataset with the radius being the average distance between the elements of this element's class. Note that the radius depends on the element so that elements belonging to different classes may have different radii (since the distances are different for each element, resulting to a different average distance for each one of them). Once the radius r of an element *i* is established by taking the average distance among the elements of i's class we can count both the total number of elements of the dataset and the number of elements of the dataset belonging to the same class as *i*, within the radius distance r from *i*. The aim of having a non-fixed radius for each pattern is that it is guaranteed that there will exist elements within the hypershere around the pattern. Also, the average distance is chosen because it is non-extreme value; if an extreme value for the radius was chosen, the result would be a very inaccurate estimation of the Discernibility of the pattern, and the whole dataset in consequence. Afterwards, by taking the number of patterns that have a Discernibility score over 0.5 and dividing it by the total number of patterns of the dataset, we can obtain a Discernibility score which charaterises the whole dataset.

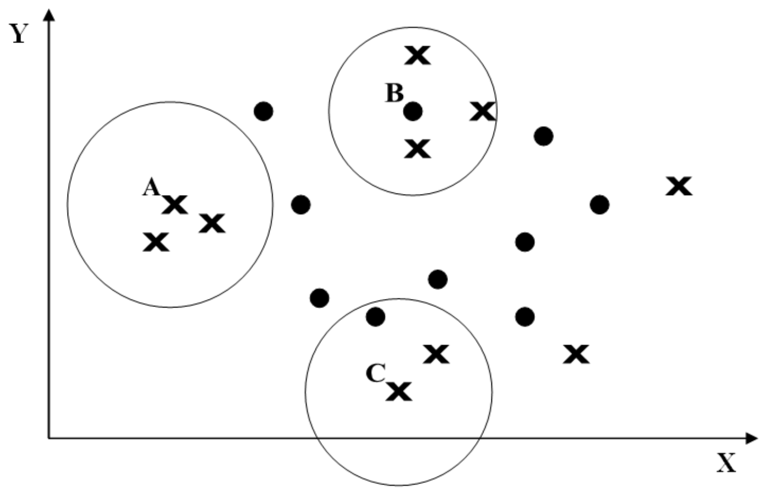

Figure 3.1. An illustration of the Discernibility of elements in a two-class data set. In this example, the Discernibility of the element in the centre of circle A is D1 = 2 / 2 = 1; that of the central element in the circle B is D2 = 0 / 3 = 0, and the Discernibility of the central element in circle C is D3 = 1 / 2 = 0.5.

The Spherical Index of Discernibility (SID) of the element *i*, SID(i), is defined as the ratio of the latter and the former, that is, proportion of *i*'s class elements among all the training dataset patterns in the hyper-sphere of radius r centred at *i* (see Figure 3.1). The greater the degree of Discernibility, the better is the chance that the classifier's prediction is correct. This index has been used recently for conditioning the kNN classifier, which resulted in improved performances of the classifier (Voulgaris & Magoulas, 2008a).

The Spherical Index of Discernibility is implemented in a fast and reliable function having a complexity of O(N^2 na). This function takes the input values of the patterns of a dataset as well as their labels and provides a vector containing the discernibilities of each one of the patterns, and a number for the overall Discernibility of the dataset. The latter is the ration of the number of patterns having Discernibility equal or higher than a given threshold (usually 0.5). Its pseudo-code is presented in Algorithm 3.1.

Algorithm 3.1 – Pseudo-code for calculating the Spherical Index of Discernibility
Inputs: Input data (P), Data Labels (T)
Outputs: Discernibilities of data elements (z_i), Spherical Index of Discernibility (SID)

```
1  Initialisation: set $l$=0 (index over the problem's classes), $i$=0 (index over the data
   elements/patterns), $th$=0.5 (default threshold of discernible element)
2  N ← number of data elements in P
3  q ← number of unique values of T (classes)
4  do $l \leftarrow (l + 1)$
5    $C_l$ ← data elements belonging to class $l$
6    $D_l$ ← distance matrix of class $l$
7    $r_l$ ← calculate radius as average distance between 2 elements of the class $l$
8  until $l$ = q
9  do $i \leftarrow (i + 1)$
10   b ← class of the $i$-th element based on T
11   d ← distances of the $i$-th element from all other elements of the dataset
12   n ← number of elements for which $d \leq r_b$
13   c ← number of elements belonging to class b
         for which $d \leq r_b$
14   $z_i$ ←  c / n
15 until $i$ = N
16 SID ← number of elements for which $z_i \geq th$
```

The computational complexity of this algorithm is $O(N^2 na)$.

3.2 Harmonic Index of Discernibility

The harmonic mean H of a set of numbers $x_1, x_2, .., x_N$ is defined as

$$H = N/(\Sigma_k 1/x_k) \tag{3.1}$$

The basic characteristic of this type of mean tends to be much nearer to smaller values among the set than the arithmetic average. By employing this in the distances of a given pattern to the others patterns of its class and of the distances of that pattern to the other patterns of other classes, we can obtain a more accurate measure of the Discernibility concept. Also, by employing this approach, we do not need to use a hypersphere, rendering this new measure of Discernibility more robust. This measure we call Harmonic Index of Discernibility (HID) as it makes use of harmonic means in its calculation. For a given a pattern *i* belonging to class *c* it is defined using Eq. (3.2)

$$\mathrm{HID}(i) = \frac{z_2(i) - z_1(i)}{z_2(i) + z_1(i)} \tag{3.2}$$

where $z_1(i) = H(dist(i, Ai))$, and $z_2(i) = H(dist(i, Bi))$, where A*i* = entities of class *c*, B*i* = entities of all other classes. In both A*i* and B*i* pattern *i* is excluded from the set to avoid unnecessary null distances. The harmonic distances are calculated using Eq.

(3.1). Also, we add a small positive number to the denominator, to avoid division by 0. In the rare case that HID(i) is negative (i.e. when the distances of the patterns of its own class are generally greater than those of the patterns of the other classes), it is adjusted to be 0, since a negative value would be meaningless and to maintain coherency between the two indexes). The pseudo-code for the calculation of this measure, having complexity of $O(N^2 na)$, can be viewed in Algorithm 3.2.

Algorithm 3.2 – Pseudo-code for calculating the Harmonic Index of Discernibility
Inputs: Input data (P), Data Labels (T)
Outputs: Discernibilities of data elements (z_i), Harmonic Index of Discernibility (HID)

```
1  Initialisation: set l=0 (index over the problem's classes), i=0 (index over the data
   elements/patterns), th=0.5 (default threshold of discernible element)
2  N ← number of data elements in P
3  q ← number of unique values of T (classes)
4  do l ← (l + 1)
5    C_l ← data elements belonging to class l
6    D_l ← distance matrix of class l (a matrix containing all the distances between all possible pairs
     of patterns in this class)
7  until l = q
8  do i ← (i + 1)
9    b ← class of the i-th element based on T
10   d ← distances of the i-th element from all other elements of the dataset
11   a1 ← distances of i-th element from elements of class b
12   a2 ← distances of i-th element from elements not of class b
13   z_1(i) ← harmonic mean of a1 (using Eq. 3.1)
14   z_2(i) ← harmonic mean of a2 (using Eq. 3.1)
15   z_i ← (z_2(i) – z_1(i)) / (z_2(i) + z_1(i) + eps)
16 until i = N
17 HID ← number of elements for which z_i ≥ th
```

Note: eps is a very small number (having a preset value in MATLAB). The computational complexity of this algorithm is $O(N^2 na)$. Yet, it is slightly slower than SID.

3.3 Comparison with Relevant Work

Both the Spherical and the Harmonic index of Discernibility reflect how distinguished the classes of a dataset are, and there is a high correlation among them. However, in some case one performs better than the other. Yet, they both perform generally better than another measure of Discernibility, Silhouette Width (SW). The latter has been used successfully in clustering (Kaufman & Rousseeuw, 1990), although to the best of our knowledge there are no records of it being used in pattern classification. It is calculated using Eq. 3.3.

$$SW = \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{3.3}$$

where a(i) = average similarity between i and all other entities of the cluster it belongs to, b(i) = minimum of the average similarity of i and all the other entities in other clusters. It takes values in the interval [-1, 1].

The advantage the proposed measures (particularly the Harmonic Index of Discernibility) have over the Silhouette Width measure is that they take into account all the different classes, not merely the class of the element examined and the foreign class closest to it. This allows the Index of Discernibility to have a wider perspective of the dataset and provide a more accurate insight over its structural properties.

In order to investigate the difference in performance of the Index of Discernibility and the Silhouette Width measure, we conducted a set of experiments, on some artificial datasets. We used this type of datasets because the extent to which the classes are distinguishable is easily verified, since they are all two-dimensional. These datasets were the following:

- "Concentric" dataset (ELENA Project Artificial Databases, 2008): a dataset consisting of classes in the form of two concentric circles non-linearly separated. It has 5000 points and its class structure can be seen in Figure 3.2.
- Dual Triangles dataset: an artificial dataset consisting of 2 classes forming 2 triangles, linearly separated. It has 500 points.
- Quadruple Squares dataset: an artificial dataset consisting of 4 classes forming 4 squares, linearly separated. It has 2000 points.
- Random Box 2 dataset: an artificial dataset consisting of 2 classes overlapping completely. It has 500 points.
- Random Box 3 dataset: an artificial dataset consisting of 3 classes overlapping completely. It has 900 points.

Note that apart from the first dataset, all the others are of our own design. Also, all of them are balanced in their class structure.

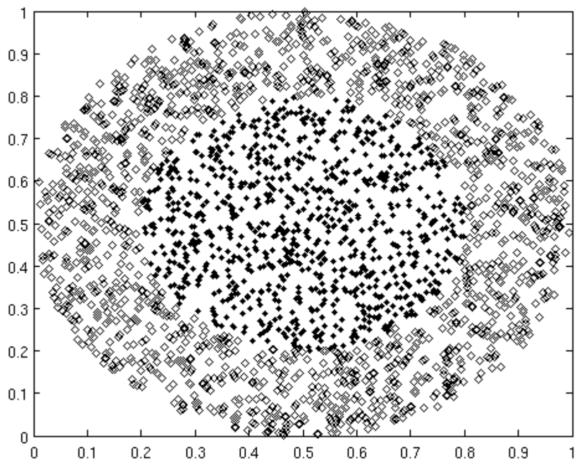
Figure 3.2. Graphical representation of the class structure of *concentric* dataset. The patterns of class “0” are depicted as squares while diamonds denote patterns of class “1” . As it can be seen from the graph, the two classes are easily distinguishable as they form two concentric circles. Therefore, their Discernibility should be high.

Since Silhouette Width takes values between -1 and 1, for the sake of comparison with the two Indexes of Discernibility it was normalised to [0, 1] . Having done this transformation we obtained the measures presented in Table 3.1 for the five different datasets. The results of these demonstrations can be generalised to more complex datasets, consisting of more attributes and a higher number of patterns.

Dataset	Silhouette Width	SID	HID	Desired Disc.
Concentric	0.4905	0.9416	0.9856	1
Dual Triangles	0.7352	0.9720	0.9820	1
Quadruple Squares	0.5996	0.8870	0.9990	1
Random Box 2	0.4986	0.5440	0.5220	0
Random Box 3	0.4863	0.0000 (mean = 0.3331)	0.4033	0

Table 3.1. Results of the Discernibility experiments, carried out using Silhouette Width and the two versions of the Index of Discernibility, proposed in this project. Note that in the last dataset, even if we take the average values of the individual discernibilities of its patterns, measured by SID, the overall Discernibility is quite lower than that yielded by Silhouette Width.

From the results presented in Table 3.1 it can be observed that both SID and HID perform better than Silhouette Width in describing the class overlap of a dataset.

The only exception was in the *Random Box 2* dataset where the overlap is maximum (and therefore all these measures should be zero), and SW had the lowest value. Yet even in this case, the Index of Discernibility (particularly HD) was not far behind, being slightly higher than the SW. Note that in a similar dataset, where there are three classes instead of two, the Indexes of Discernibility (particularly SID) describe it much better than Silhouette Width.

Although originally the Discernibility measures developed in this work were intended as a tool for assessing the dataset structure, it turned out that the discernibilities of the individual patterns of the dataset as well as the discernibilities of the individual features, were equally important (or even more important in some cases). This allowed the concept to expand its applicability in other areas of Classification. This was one of the reasons that different versions of it were developed.

Note that these Discernibility measures presented here should not be confused with the Discernibility tables, or the indiscernibility concept, used in Rough Sets (Komorowski et al., 1998). The latter are completely different both in function and in field of application since they do not deal with individual patterns or whole datasets, but rather with variables.

Also, it is noteworthy that recently there has been similar research, independent of this project, investigating a supplementary concept but with the use of SOMs (Lemeni & Tepus, 2008). Particularly, a metric for describing the extent to which the classes of a dataset overlap (Class Overlap Degree Coefficient) was introduced. This was defined as the number of the nodes of a SOM that fall on the common area of two or more classes, over the total number of nodes.

Yet, due to the increased overhead of that method it is not considered a practical alternative to measuring the class overlap in a dataset. Between the other two, Index of Discernibility and Silhouette Width, the latter is relatively inferior as it has been shown in the experiments of the previous section.

It is interesting how the Discernibility concept can be applied to the feature evaluation of a dataset. This can be done either individually or in groups. For the former type of evaluation, each feature is evaluated separately, as seen in Algorithm 3.3.

Algorithm 3.3 – Pseudo-code for evaluating feature individually using Discernibility
Inputs: Input data (P), Data Labels (T)
Outputs: Discernibilities of features (z_i)

```
1 Initialisation: set $l$=0 (index over the problem's classes), $i$=0 (index over the features)
2 na ← number of features of dataset
3 do $i \leftarrow (i + 1)$
4    $P_i$ ← feature $i$ of Input data P ($i$-th column or $i$-th row of P, depending on format)
5    $z_i$ ← ID($P_i$,T) of $i$-th feature(using either Algorithm 3.1 or Algorithm 3.2)
6 until $i$ = na
```

The computational complexity of this algorithm, regardless of which ID metric is used, is $O(N^2\,na^2)$.

3.4 Degree of Certainty and Degree of Reliability

Previously we have examined how the dataset quality can be evaluated be means of the Discernibility concept, implemented in either one of the two measures developed in our research. The question that now arises is whether the same can be done with respect to the classification quality. In other words, is there a way of having a more or less objective view of how "good" a classification is *a priori*, i.e. before the evaluation phase? The answer to this question is affirmative and there have been various methods developed to accomplish that. These have been referred to with many names, such as Chow's index (Bresnahan, 1986), Certainty Factor, Inductive (or Transductive) Conformal Prediction (ICP) (Papadopoulos, 2004; Papadopoulos, Vovk, & Gammerman, 2007). There have been a couple of measures proposed by the author of this thesis as well, as we will see later on. The reason for doing so is that we would need a measure that is versatile enough so as to be applicable to all (or almost all) types of classifiers.

Now, let us first consider an analogue to the Chow's index applied to any classifier's scoring function, for which Aidin and Guvenir coined the term Certainty Factor (CF) (Aydin & Guvenir, 2006). It is assumed that the classifier under consideration, for each pattern i and each class, produces a classification score, so that the class for which the score is maximal is predicted for i by the classifier. The relative proportion of this maximum classification score is referred to as the Degree of Certainty (DC):

$$DC_i = \frac{\max\limits_c(S_c)}{\sum\limits_{c=1}^{q} S_c} \tag{3.4}$$

where i denotes the i-th pattern classified, c the class label, q the number of classes and S_c is the classification output of the classifier for class c. Unfortunately, this index by itself cannot be used as a measure of reliability of classification. Experiments have shown that it provides a very low correlation with the true class labels. There is however a correlation between this measure and the expected Index of Discernibility of a test pattern. This is natural if one considers that a discernible pattern (i.e. one with a high Discernibility score) is bound to be predicted correctly, so the classifier will tend to be relatively certain about its classification. This observation led to the development of a measure of reliability, which is independent of the classifier used, namely the Degree of Reliability (DR), defined and discussed below.

The Degree of Reliability is a way of assessing how "sure" a classification is, regardless of the classifier used. It aims to tackle the problem of classifier dependency that arises in the case of Degree of Certainty. The price one pays for it is that the Degree of Reliability is a fairly time-consuming process and in some cases, impractical (e.g. when using ANNs or any other classifier that has a relatively long training phase). This measure reflects the how typical an unknown entity is as a representative of its class, which is evaluated by one of the Indices of Discernibility applied on the entity. Since the class of this entity in not known, the predicted class is used, according to the classifier employed.

The calculation of the Degree of Reliability employs the Index of Discernibility (mainly the Spherical version due to its simplicity and slightly better performance) using the predicted class as a class label of the test pattern to be evaluated. This has been tested with a number of classifiers for different datasets, with promising results (Voulgaris & Mirkin, 2008a), leading to the development of a Reject Option method, which is described in Chapter 6.

The subjectivity of the Degree of Certainty can also be overcome if one considers its relative change instead. This is employed in a version of the Degree of Reliability, called the Parametric Degree of Reliability, which is thoroughly described in Chapter 6. Just like the original Degree of Reliability, the parametric version of it makes use of the Spherical and Harmonic Indexes of Discernibility as they are. The difference in that version is that the Discernibility metric used is accompanied by two other measures, and three parameters. We will investigate this measure further in Chapter 6 where we will also explore its applications in Classification.

It is worth noting that the Degree of Certainty as well as the Degree of Reliability are different to the Inductive (or Transductive) Conformal Prediction-ICP (Papadopoulos, 2004; Papadopoulos et al, 2007; Papadopoulos, 2008), yet they are complimentary in a way. The difference is that the proposed measures (DC and DR) aim to give an insight to how confident the classifier is about the prediction it makes, while the ICP measure, being probabilistic in nature, offers the maximum probability that a prediction is wrong. In addition, DC and DR are generally light in terms of computational cost. Due to these reasons, the proposed methods are the ones employed in most of the experiments conducted for this research.

3.5 Net Reliability

As seen earlier, the Degree of Certainty provides an insight on how probable it is for the classification to be accurate, according to a particular classifier. However, often this may be a misleading piece of information, as in cases where the dataset is complex, the classifier may be "sure" about something completely wrong. Thus, the use of a measure that evaluates the Degree of Certainty of a classifier, and therefore the classifier's reliability, is essential. The use of correlation coefficient (a statistical measure R, taking values between -1 and 1, that show how correlated two variables are, based on their variances and their covariance) is not a good choice because in some cases it fails to provide any results, since it is usually applicable in scale variables. For example, there are cases where the correlation coefficient may yield an infinite value as an output, which is meaningless to the user. Besides, the binary nature of the validity vector (a binary vector *v* which depicts each correct classification with a 1 and each wrong one with a 0) makes it incompatible with the variance metric employed in the correlation coefficient, since it often fails to deal with nominal variables like this one. Therefore we have developed another measure for this, the Net Reliability. This is expressed by Equation 3.4 as seen below.

$$NR = \frac{1}{n}\sum_{i=1}^{n}\left[(2v_i - 1)\cdot DC_i\right] \qquad (3.5)$$

where *v* is the classification validity vector, *DC* is the Degree of Certainty vector, *i* denotes the pattern classified, and *n* is the total number of elements in the test set. A high Net Reliability score for a classifier is generally good for the classifier . The Net Reliability of a classification takes values between –1 and 1 (inclusive).

Although this measure was primarily developed for the evaluation of a classifier after the validation stage, it may find use in the pre-processing stage, by applying in on a subset of the training set as a validation set.

It has been observed that almost all of the classifiers tend to have a high Net Reliability in datasets exhibiting a high Index of Discernibility.

Note that the Net Reliability measure is not linked in any direct way to the Degree of Reliability. Besides, its function lies in assessing how reliable the Degree of Certainty is, based on the correct classifications, while the Degree of Reliability is a way of *predicting* how reliable a classification is, prior to accessing the correct classification values. However, as they both refer to ways of measuring reliability, they both have this term in their name.

3.6 Summary and Contribution of the Chapter

This chapter has introduced and described Discernibility at a conceptual level. Also, it has presented the two versions of its implementation. In addition, a comparison with the known similar measures is carried out– the SOM-based Class Overlap Degree Coefficient and the Silhouette Width. The latter, in particular, is compared with ID experimentally for five different artificial datasets and has been found inferior. Furthermore, Discernibility's relationship with the Degree of Certainty and Net Reliability has been pinpointed and discussed. Finally, the Degree of Reliability, an alternative measure based on Discernibility, has been introduced.

This chapter is considered a fundamental part of the thesis as the whole research involved in this project revolves around the ideas presented here. This chapter contributes to the field of Pattern Classification by offering a few simple to implement methods that can enhance and in many cases speed up the classification process. Furthermore, it is important to note that the metrics introduced in this chapter can be applied in a variety of cases and are independent of the classifiers use as well as the datasets.

Chapter 4 – Discernibility-Based Classification

The simplicity and high convergence speed of the k Nearest Neighbour (kNN) classifier have made it a popular choice for pattern classification (Moreno-Seco et al., 2003) with applications in a variety of cases (Abidin & Perrizo, 2006; Khan et al., 2002; Yu & Ji, 2002). Nevertheless, there are situations where kNN might fail to produce adequate results (Sotoca et al., 2003), or its operation may render it impractical (Moreno-Seco et al., 2003). Such cases are in problems involving high dimensionality and/or complex datasets with high class overlap. In practice, the fact that kNN only requires the user to specify one parameter, the number of neighbours used (*k*), facilitates fine-tuning the method to a variety of situations. The main kNN classifier consists of the following steps: given a set P of *N* labelled points (training set), P={x_1, …,x_N}, classify a set PT of n points (testing set), PT={t_1, …,t_n}, into the same set of classes (labels) by examining the *k* closest points around each point of the testing set, and by applying the majority vote scheme (Duda et al., 2001).

The kNN classifier is a suboptimal procedure, with a few inherent problems such as inability to deal with complex dataset, which is why researchers have proposed different extensions of the kNN (Bermejo & Cabestany, 2000; Manocha & Girolami, 2007; Hattori & Takahashi, 2000; Gao & Wang, 2007; Lai et al., 2007; Warfield, 1996; Wu et al., 2002; Zhou & Chen, 2006; Wang et al., 2006; Wang et al., 2007), or even ensemble formulations of kNN classifiers (Domeniconi & Yan, 2005). Most of these approaches have exhibited some interesting and quite promising results and have motivated further research on improving the kNN method.

In this chapter two kNN variations based on the Spherical Index of Discernibility introduced in Chapter 3 will be presented and discussed. Also, two more kNN variations will be introduced as well as a classifier which is based on Minimum Spanning Trees (Kruskal, 1956), which are graphs describing the shortest possible total length of line segments connecting a given number of points, so as to connect all of them (for an overview see (Graham & Hell, 1985)). The distance computation in all of them is carried out using the Euclidean distance.

Other distance alternatives include the Manhattan (city-block) distance (Kawahara & Shibata, 2005), the Euclidean Squared distance (Bailey, 2004) and the Mahalanobis distance (Torra et al., 2006). However, as the Euclidean distance is the most popular one, it is the one chosen for this research.

4.1 Background Research on kNN Extensions

Although the kNN method when used in classification problems is quite fast, it is often impeded by the size of certain datasets, which is why some researchers have focused on improving its speed (Abidin & Perrizo, 2006; Lai, 2007; Warfield, 1996). SMART-TV for instance (Abidin & Perrizo, 2006) was designed to deal with datasets of high dimensionality by transforming them into a single-dimensional feature space. A similar approach is shared by Khan et al. (2002) for spatial data, where this method works best (Abidin & Perrizo, 2006, Mainar-Ruiz & Pérez-Cortes, 2006) mainly due to the simplicity of the dataset structure . Yet, these approaches concentrate on high speed mainly and often fail to achieve exceptionally good accuracy rate unless they are applied on particular problems, such as spatial datasets (Khan et al., 2002) and images (Warfield, 1996).

Other approaches involve feature selection methods which are tested on several both real and artificial datasets (Sotoca et al., 2003). These methods, which focus on establishing appropriate weights to the various features, appear to be promising; yet, the methods described in the literature are not always very fast, since it appears to be a trade-off between performance and speed (Sotoca et al., 2003).

Changing the way distance is dealt with, in a fundamental level by considering a different method of evaluating distance, is an interesting alternative, which can improve speed considerably (since less calculations are required) without having any significant reductions in the accuracy of the classification rate (Moreno-Seco et al., 2003). When dealing with complex problems this can be quite fruitful (Yu & Ji, 2002), yet these methods appear to be rather cumbersome when applied to other simpler datasets. Other methods of this category consider cam weighted distance[*] (Zhou & Chen, 2006), or an adaptive distance (Wang et al., 2007), and appear quite promising.

Often it is more efficient to combine different classifiers, either by forming a low-level mixture where the classifiers interact during the training phase (Hendrickx & Van den Bosch, 2004) or by building an ensemble (Domeniconi & Yan, 2005). In the first case, it becomes apparent that changes in the structure of the kNN classifier may be essential in order to improve its performance. In the second case, creating a diversity ensemble classifier is attempted by combining negatively correlated

[*] This type of distance takes its name from the deflective cam contours for equal-distance contour in classification it yields.

classifiers (i.e. classifiers whose errors are as different as possible)in an ensemble formulation; an approach that seems to improve the accuracy rate and which is used in Chapter 7. Yet, the results although interesting, denote that kNN-based approaches still require much improvement if they are to be used in ensembles to target various classes of problems.

The use of rules, which take the form of additional features in the dataset, in kNN has been researched by Van den Bosch (2004) who put forward three methods which were tested on several datasets with some success. However, in many datasets the creation of rules may be time-consuming and even computationally expensive. Also, in high dimensional datasets, the additional cost could make the classification process very slow and therefore inefficient.

Another type of extensions considers the use of statistics to make kNN more robust (Manocha & Girolami, 2007; Wang et al., 2006). This is done either by transforming the kNN classifier into a probabilistic classifier (Manocha & Girolami, 2007) by approximating the probability of a classification using a method based on cross validation for various k values, or by using statistics to find the best size for the neighbourhood involved (Wang et al., 2006). The former approach, although it does not have significantly better results than the classic kNN, it provides continuous preditive probabilities thus offering a way of dealing with uneven misclassification costs.

Another method encountered in the literature questions the efficiency of the voting scheme of the kNN (Wang & Bell, 2004), and proposes an alternative measure for determining how each class is related to a test point. This approach is taken one step further in one of the kNN variations proposed in this chapter (see sections 4.2.1 and 4.2.2), since the use of only one measure (distance) to assess the relationship to a class is often insufficient.

Of the methods described above, the one by Wang & Bell (2004) goes beyond the simple counting of neighbours as it evaluates them as well. Sharing this philosophy, we bring forward a kNN variant, which assigns a quality index to each element of the dataset. Also, similarly to Sotoca et al. (2003), we introduce a classifier that makes use of different weights for the various features of the dataset.

4.2 kNN Extensions Based on Discernibility

4.2.1 Discernibility kNN

The Discernibility kNN (D-kNN) first calculates the discernibility of the neighbours as well as their distances from the test patterns. By taking the ratio of these two factors, a score is produced for each one of the neighbours. Based on the neighbours' score, scores for each class are then averaged to produce one classification score for each one of them. The class yielding the highest classification score is selected as the most probable output for the classification. Algorithm 4.1, below, provides a high level description of the D-kNN.

Algorithm 4.1 – The D-kNN classifier
Inputs: Input patterns of training set (P), labels of training set (T), number of neighbours (*k*), patters of testing set (PT)
Outputs: classification vector of testing Set (y)

```
1  Initialisation: set $i$=0 (index of the patterns), $j$=0 (index over the classes)
2  n ← number of patterns of testing set (PT)
3  N ← number of patterns of training set (P)
4  $\mathbf{z}$ ← discernibility vector for elements of P, using Algorithm-1and P, T as inputs; $\mathbf{z} = \{z_i\}$,
       $i = 1 \dots N$
5  q ← number of unique values of T (classes)
6  do $i \leftarrow (i + 1)$
7    $\mathbf{D}$ ← vector of distances of PT($i$) to P based on Eq. (4.1)
8    $\mathbf{sd}$ ← sorted values of $\mathbf{d}$
9    $\mathbf{dk}$ ← $k$ first values of $\mathbf{sd}$
10   $\mathbf{v} \leftarrow \{ v_m : v_m = z_m / dk_m \}, m = 1 \dots k$
11   do $j \leftarrow (j + 1)$ for current $i$
12     $C_j$ ← subset of $k$ nearest elements of P belonging to the $j$-th class
13     Classification score $S_j \leftarrow$ mean($\mathbf{v}_{C_j}$),
       $\mathbf{v}_{C_j} = \{v_m : \forall m \in C_j \}$
14   until $j$ = q
15   $b \leftarrow \arg\max_j S_j$
16   $y_i$ ← class $b$
17 until $i$ = n
```

When the classes of a dataset are overlapping, the D-kNN classifier appears to provide an advantage over the standard kNN. In these cases the kNN classifier is bound to miss many entities of the test set, which D-kNN is bound to predict more accurately by taking into account the structural properties of the neighbours (reflected on their Discernibility scores) as well as their distances from each test pattern.

Particularly it calculates the ratio of Discernibility / distance for each neighbour and takes the average of these ratios for each class. This can be illustrated using an example from the *clouds* dataset, taken from (ELENA Project Artificial Databases, 2008). Compared to kNN's inaccurate classification (as shown in Fig. 4.1), the D-kNN classifies the test element correctly as it takes into account not only the fact that they are nearest neighbours, but also the Discernibility scores and their distances (Fig. 4.2). The D-kNN classifier (Algorithm 4.1) does this by producing for each pattern to be classified a score vector **v**, for all of its neighbours. This is a function of the discernibilities of the neighbouring patterns and their distances from the test pattern. Then, by averaging these scores for each one of the possible classes, the classification score of each class is calculated (*Sj*). Finally, the test pattern is classified by comparing the classification scores of the different classes of the dataset, the highest of which wins the classification.

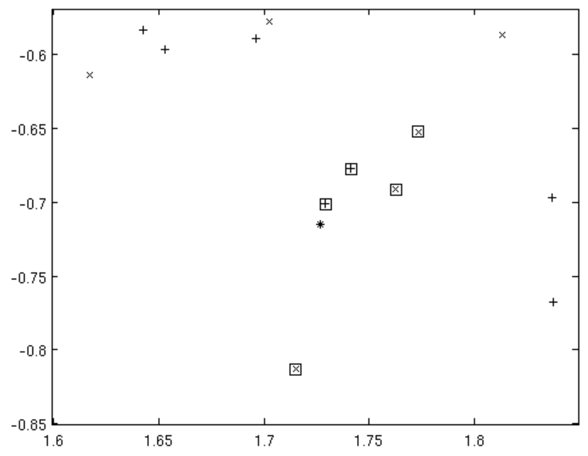

Figure 4.1 – kNN's performance on a pattern of the *clouds* dataset. The test pattern is marked as (*) while the patterns of the two classes of the dataset are marked as (+) and (x) respectively. The boxes around some patterns show the patterns that are taken into account for classification. kNN fails to classify the pattern at hand correctly ($k = 5$).

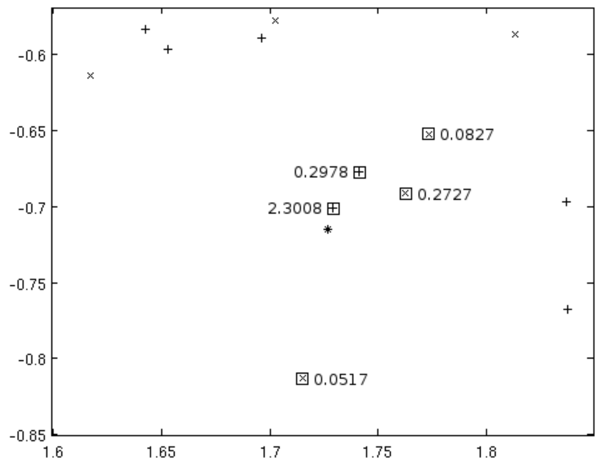

Figure 4.2 – D-kNN's performance on the same pattern of the *clouds* dataset (as Fig. 4.1). D-kNN classifies the pattern at hand correctly to the class represented by (+) since it considers not the number of neighbours but their significance in terms of Discernibility and their individual distances to the test element, in the form of a ratio. The score v_m calculated by the D-kNN is depicted next to each one of the patterns taken into account for the classification. The patterns inside the boxes are the ones taken into account for the classification (k = 5).

4.2.2 Weight-based kNN

The Weight-based kNN (W-kNN) performs an evaluation of the features of a dataset based on the training set P. First, each one of the features f of the training set is evaluated using Algorithm 3.3 and the Indices of Discernibility for the various features are found (in Algorithm 4.2, below, they are denoted as ID_f). The weights are then obtained by normalising the IDs. Lastly, the weights are applied on both the training and the testing set thus changing the features' values, resulting in a transformation of the dataset's feature space. Afterwards, it performs classification on this space, using the kNN classification rule. An algorithm model of the W-kNN is presented in Algorithm 4.2.

This classifier is particularly useful in the case where a dataset has many features, some of which can be considered unnecessary, since W-kNN deals with them by giving them appropriate weights, based on their Discernibility scores. These features often confuse the kNN classifier, yet by transforming the feature space as W-kNN does, this situation can be alleviated as the unnecessary features are bound to have a low Discernibility score ID_f so they will not count significantly in the transformed feature space (due to their low relative weight). Afterwards, one may choose to disregard these features altogether, performing feature selection.

Algorithm 4.2 – The W-kNN classifier
Inputs: Input patterns of training set (P), labels of training set (T), number of neighbours (*k*), patterns of testing set (PT)
Outputs: classification vector of testing set (y)

```
1  Initialisation: set $f$=0 (index over the features)
2  n ← number of patterns of testing set (PT)
3  N ← number of patterns of training set (P)
4  m ← number of features in dataset
5  do $f$ ← ($f$+1)
     $P_f$ ← vector of values for the $f$ feature of P
6    $\mathbf{ID}_f$ ← $\mathbf{ID}$ ($P_f$, T) using Alg. 3.1
7  until $f$ = m
8  $\mathbf{w}$ ← $\mathbf{ID}$ / sum($\mathbf{ID}$)
9  P2 ← P * ($\mathbf{w}$ * I)
10 PT2 ← PT * ($\mathbf{w}$ * I)
11 $\mathbf{y}$ ← classification output of kNN based on P2, PT2 and $k$
```

Where I is the unitary matrix of order m (an mxm matrix containing ones on the diagonal and zeros everywhere else).

An example of W-kNN's edge in performance over kNN is shown in Figure 4.3, where the *clouds* dataset from (ELENA Project Artificial Databases, 2008) is used as an example. The test pattern to be identified, denoted by an asterisk (*) in the figure, belongs to the class depicted with the crosses (+). The patterns inside a square are the ones taken into account for the classification. The kNN takes into account the k = 5 closest neighbours (Fig. 4.1) and fails to classify the test pattern (*) correctly. Note that kNN would still fail with k = 3 or k = 7, in this case. W-kNN on the other hand transforms the feature space by giving more importance to the second feature (y axis) at the expense of the first one (x axis), because the latter has a lower Discernibility score. Because of this, W-kNN classifies the pattern at hand accurately as it makes use of a slightly different set of data points (Fig. 4.3).

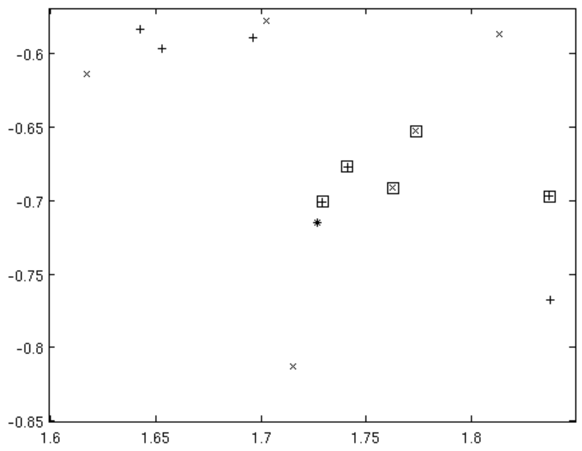

Figure 4.3 – W-kNN's performance on a pattern of the *clouds* dataset. W-kNN classifies the pattern at hand correctly since it makes use of the *k* nearest neighbours in a transformed feature space where the first feature (depicted as the horizontal axis) is considered less important and thus the distances over it are compromised, yielding a better result. The patterns in the boxes are the ones taken into account by the classifier. Note that the rightmost neighbour is taken into account by W-kNN because its distance in the transformed feature space (where the first feature is not so important) is not so great, so it is near enough to be considered as one of the *k* = 5 nearest neighbours.

4.2.3 Experimental Results of the Discernibility-based kNN Extensions

4.2.3.1 Datasets Used and Experimental Setup

The characteristics of the datasets used in our study are summarised in Table 4.1. All the classifiers used the *k* value which was considered the best choice, based on the work in Voulgaris & Magoulas (2008a). These values ranged from 2 to 7, based on the output of the V-kNN classifier, introduced and discussed in Section 4.3.1.1.

The six datasets were downloaded from the UCI repository (UCI Repository, 2008) and the experiments were carried out in MATLAB 2007a, where the tested algorithms were implemented. The CPU time was measured using a built-in function in MATLAB, and took into account the training phase of each classifier. As kNN has no training phase, in all of the experiments its CPU time was set to 0. The experiments included 50 rounds of 10-fold cross-validation (500 classifications altogether for each one of the classifiers).

Dataset	Features	Patterns	Classes
Bupa Liver	6	345	2
Pima Indians	8	768	2
Breast Cancer W.	9	683	2
Heart Disease	13	270	2
Vehicle	18	846	4
Boston Housing	13	506	3

Table 4.1 – Characteristics of the datasets used in the experiments

4.2.3.2 Evaluation Criteria

The classifiers were assessed using the average Accuracy Rate, a measure which has been used extensively in the literature (Gao & Wang, 2007; Wu et al., 2002), the CPU time, and a correlation between Accuracy and Degree of Certainty (Net Reliability, described in Chapter 3). Note that all of the datasets used are more or less balanced, and thus eligible for the evaluation measure of Accuracy Rate.

The CPU time is defined as the interval between two consecutive checks of the CPU clock of the computer. It is considered to be more reliable than the time interval measured with a stopwatch by the user, because it takes into account only the CPU usage of the experiments themselves. This way it is not influenced much by the other processes that may run simultaneously in the OS.

4.2.3.3 Results

In the first dataset used, Bupa Liver, the best classifier in terms of performance was D-kNN, due to the very high Accuracy Rate (over 3% compared to kNN) and the rather high Net Reliability it exhibited. It was moderately fast, as the calculation of the index of discernibility for each pattern was rather time-consuming. The detailed results for this dataset can be seen in Table 4.2.

Classifier	Accuracy Rate	Net Reliability	CPU Time
kNN	62.99%	0.1964	0.0000
W-kNN	62.48%	0.1892	0.0226
D-kNN	66.31%	0.2572	0.0510

Table 4.2 – Results for the Bupa Liver dataset. Undoubtedly, the winning classifier is D-kNN, scoring better in both Accuracy Rate and Net Reliability.

It appears that the W-kNN classifier did not perform as well as the classic kNN, in terms of accuracy rate and Net Reliability. This was attributed to the nature of the dataset, particularly the fact that it contains only 6 features, all of which are of the same importance (which is mirrored by the SID values of the features: 0.5797 for each one of them. Interestingly, the SID of the whole dataset is slightly higher: 0.5855). For W-kNN to perform well, a larger number of features is needed, so that some diversity among their value exists.

Regarding the Pima Indians dataset, as it can be seen in Table 4.3, W-kNN is the best in both Accuracy Rate and Net Reliability (although its accuracy rate is coined by the kNN classifier as well). Also, compared to the D-kNN classifiers, it is the fastest too. As this dataset was not complex (the calculated Index of Discernibility is 0.6862 denoting that the classes of the dataset are not overlapping so much), it is natural that D-kNN does not have an advantage, hence its performance.

Classifier	Accuracy Rate	Net Reliability	CPU Time
kNN	73.70%	0.4199	0.0000
W-kNN	73.70%	0.4241	0.1020
D-kNN	72.99%	0.4299	0.2445

Table 4.3 – Results for the Pima Indians dataset. The winning classifier is W-kNN, combining good Accuracy Rate, Net Reliability and CPU time.

Concerning the Breast Cancer Wisconsin dataset, the most accurate classifier is D-kNN, which has also the best Net Reliability although its CPU time is not so good. Also, the W-kNN scored quite well, and was much faster too compared to D-kNN. The results for this dataset can be seen in Table 4.4.

Classifier	Accuracy Rate	Net Reliability	CPU Time
kNN	94.50%	0.9039	0.0000
W-kNN	94.80%	0.9072	0.0920
D-kNN	95.90%	0.9169	0.1805

Table 4.4 – Results for the Breast Cancer Wisconsin dataset. The winning classifier is D-kNN, scoring better in Net Reliability and Accuracy Rate.

In the Heart Disease dataset, the best classifier is D-kNN, which had the highest Accuracy Rate, the best Net Reliability and a moderately good CPU time. The results for all three classifiers can be seen in Table 4.5.

Classifier	Accuracy Rate	Net Reliability	CPU Time
kNN	80.86%	0.5649	0.0000
W-kNN	80.35%	0.5634	0.0332
D-kNN	81.31%	0.5926	0.0554

Table 4.5 – Results for the Heart Disease dataset. The winning classifier is D-kNN, scoring better in Accuracy Rate and Net Reliability while at the same time its CPU time was reasonably good.

As for the Vehicle dataset, D-kNN was better in Accuracy Rate and Net Reliability. However, it was rather slow. In Table 4.6 one can see all the results for this dataset.

Classifier	Accuracy Rate	Net Reliability	CPU Time
kNN	70.03%	0.3899	0.0000
W-kNN	69.99%	0.3880	0.2403
D-kNN	70.44%	0.4009	0.4795

Table 4.6 – Results for the Vehicle dataset. The winning classifier is D-kNN, scoring better in Accuracy Rate and Net Reliability.

Regarding the last dataset, Boston Housing, the results are somewhat ambiguous, like in the Breast Cancer Wisconsin dataset. W-kNN was the most accurate one, having the best Net Reliability as well, while its CPU time was quite satisfactory. In Table 4.7 one can see the results for all classifiers for this dataset.

Classifier	Accuracy Rate	Net Reliability	CPU Time
kNN	67.57%	0.3308	0.0000
W-kNN	68.96%	0.3587	0.0789
D-kNN	66.99%	0.3483	0.1793

Table 4.7 – Results for the Boston Housing dataset. The winning classifier is W-kNN, scoring better in Accuracy Rate and Net Reliability. At the same time, its CPU time was good as well.

In Table 4.8, we summarise the findings of the experiments with the six datasets, showing the winner with respect to each evaluation criterion.

Dataset	**Accuracy Rate**	**Net Reliability**	**CPU Time (2nd) in sec.**
Bupa Liver	D-kNN (66.31%)	D-kNN (0.2572)	W-kNN (0.0226)
Pima Indians	W-kNN and kNN (73.70%)	D-kNN (0.4299)	W-kNN (0.1020)
Breast Cancer W.	D-kNN (95.90%)	D-kNN (0.9169)	W-kNN (0.0920)
Heart Disease	D-kNN (81.31%)	D-kNN (0.5926)	W-kNN (0.0332)
Vehicle	D-kNN (70.44%)	D-kNN (0.4009)	W-kNN (0.2403)
Boston Housing	W-kNN (68.96%)	W-kNN (0.3587)	W-kNN (0.0789)

Table 4.8 – Winners based on average performance over the 50 rounds. The winning performance metric is shown inside brackets. The Net Reliability is calculated by means of the classification and the Degree of Certainty vectors at the end of each experiment.

Afterwards, a one-to-one comparison was made for each pair of classifiers, showing how many times (rounds) one classifier outperformed the other. Then, these scores were added up for each classifier. The final sum reveals the relative performance of each classifier and is shown in Table 4.9.

Dataset	**Accuracy Rate**	**Net Reliability**	**CPU Time (2nd) in sec.**
Bupa Liver	D-kNN (97)	D-kNN (100)	W-kNN (50)
Pima Indians	W-kNN (67)	D-kNN (76)	W-kNN (50)
Breast Cancer W.	D-kNN (100)	W-kNN (90)	W-kNN (50)
Heart Disease	D-kNN (73)	D-kNN (94)	W-kNN (50)
Vehicle	D-kNN (65)	D-kNN (86)	W-kNN (50)
Boston Housing	W-kNN (94)	W-kNN (88)	W-kNN (50)

Table 4.9 – Winners based on the relative performance, pairwise, over 50 rounds. The numbers in brackets show the total number of times the winning classifier was better than the other two in terms of a particular performance measure, for each dataset. As each classifier is compared against two others, these numbers range from 0 to 100.

It is noteworthy that in all of the six datasets, kNN was outperformed by one of the variations introduced in this work. Also, the only criterion where it actually performed well was speed, since it required no training. The mean performance of the three classifiers over the six datasets is shown in Table 4.10.

Classifier	Accuracy Rate	Net Reliability	CPU Time
kNN	74.94%	0.4676	0.0000
W-kNN	75.05%	0.4718	0.1093
D-kNN	75.66%	0.4910	0.1984

Table 4.10 – Average performance of classifiers, for the six datasets they were tested on. The proposed classifiers have a higher accuracy rate than kNN and a higher reliability as well.

From the above results we can see that while both of the proposed classifiers perform better than the kNN, on average the D-kNN method performs best, taking into account both Accuracy Rate and Net Reliability. Yet, as regards speed, it is outperformed by kNN. Quite close in terms of performance comes the other kNN extension, W-kNN, which is also faster. In addition, there is a correlation between the Accuracy Rate and the Net Reliability, in the winners' tables. This is something expectable, as a winning classifier is bound to be good in both of these criteria.

4.3 Other kNN Extensions

4.3.1 Extensions of kNN with Self-determined k

Although the measures presented in this section are not strictly Discernibility-based, they are interesting variations of kNN and can compliment the methods described previously. These extensions have the interesting feature of not needing a k parameter from the user, since they optimise it on their own. Two such extensions have been developed: the Variable k Nearest Neighbour (V-kNN) and the Class-based k Nearest Neighbour (C-kNN).

4.3.1.1 Variable k Nearest Neighbour

Since the value of the *k* parameter often influences the classification results, sometimes significantly, we devised a classification algorithm that overcomes this issue. By making use of the Degree of Certainty concept it estimates the optimum *k* for each classification.

The Variable kNN (V-kNN) classifier works as follows. First for each one of the training set patterns a classification of it is performed based on various neighbourhoods and the Degree of Certainty (DC) for various *k* values is calculated. Then, the *k* value that maximises the DC of each classification is found (i.e. for every pattern to be classifier, there is bound to be a different *k* which is more appropriate). Therefore, for each training pattern, there corresponds a particular *k* value which is considered the best available (the set of these values constitutes the "best" *k* array). All this is the training phase of the V-kNN classifier. Coming to the testing phase of the classifier, for each unknown element, the nearest neighbour of the test pattern is found and based on the "best" *k* array, its *k* value is assumed. Then, the kNN classifier is applied on that test pattern using that particular *k* value. As a concept, this is something similar to one of the ideas presented by Khan et al. (2002), who showed that the kNN performance can be increased if "instead of taking exactly the k nearest neighbours a closed-KNN set is formed". The pseudocode of this classifier appears in Algorithm 4.3. Note that since the performance of kNN is not affected much by the k parameter for high values of it, a default maximum value of 20 is used. Also, a higher value would compromise the classifier's speed.

Algorithm 4.3 – The V-kNN classifier
Inputs: Input patterns of training set (P), labels of training set (T), patterns of testing set (PT)
Outputs: classification vector of testing set (y)

```
1  Initialisation: set $i = 0$ (the i-th training pattern), $j = 0$ (the j-th testing pattern), $k$ (parameter of
   the KNN, given by the user), kmax = 20
2  $n \leftarrow$ number of patterns of testing set (PT)
3  $N \leftarrow$ number of patterns of training set (P)
4  Do $i \leftarrow (i + 1)$
5    P2 $\leftarrow$ P without pattern $i$
6    do $k \leftarrow (k + 1)$
7      classify $i$ using P2 and kth nearest neighbour
8  D($k$) $\leftarrow$ calculate the Degree of Certainty of Classification using Eq. (3.2) for the particular
          $k$-th nearest neighbour
9    until $k$ = kmax
10   $K_i \leftarrow \arg\max_k D(k)$
11 until $i$ = N
12 Do $j \leftarrow (j+1)$
13   $d$(N) $\leftarrow$ vector of distances of PT(j) to each one of the elements of P, calculated using Eq. 3.1
14   $nn \leftarrow \arg\min_j d(j)$
15   $k \leftarrow$ K(nn) where K = $\{K_i\}$
16   $Y_j \leftarrow$ classification output based on kNN using above $k$
17   $DC_j \leftarrow$ Degree of Certainty based on above classification
18 until $j = n$
```

It is noteworthy that apart from its performance as a classifier, the V-kNN approach yields some useful information about the dataset: a good value for the *k* parameter, which for kNN-type classifiers is very useful to know as it improves their performance, particularly for the classical kNN classifier. However, for very sparse datasets the optimum *k* found may not be valid and the results may not be better than those of kNN. The suggested *k* values for the various dataset used are shown in Table 4.11.

Dataset	**Suggested *k***
Bupa Liver	7
Pima Indians	6
Breast Cancer W.	2
Heart Disease	6
Vehicle	5
Boston Housing	6

Table 4.11 – Suggested k values for the various datasets used. These values were obtained by averaging the *k* values V-kNN found best for each one of the patterns of the datasets it was applied on.

In datasets where the classes are quite entwined and the entities of each class are close together with each other, V-kNN definitely yields better results than kNN, as the *k* parameter needs to be redefined for each entity of the test set.

We can observe V-kNN's advantage over kNN visually in Figure 4.4. This is a close-up of the *clouds* dataset, from (ELENA Project Artificial Databases, 2008). The pattern to be identified (*) belongs to the class depicted with the crosses (+). The patterns inside a square are the ones taken into account for each classification.

While kNN takes into account the $k = 5$ closest patterns, V-kNN decides to use only the $k^* = 1$ closest pattern (Fig. 4.4), as it has found it to yield the best result among the elements of the training set. This difference in the *k* parameter can be the difference between a misclassification and a correct classification, as it is apparent in this case. Note that kNN would still fail with $k = 3$ or $k = 7$, in this case.

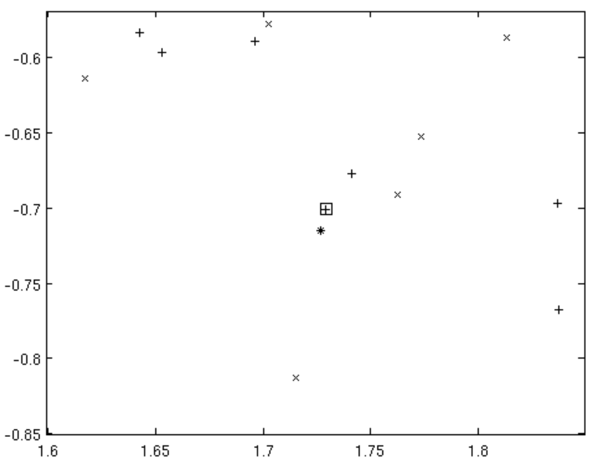

Figure 4.4 – V-kNN's performance on the same pattern of the *clouds* dataset (as Fig. 4.1 – 4.3). V-kNN classifies the pattern at hand correctly since it makes use of the best *k* for this particular pattern, which in this case is equal to 1. Hence the one and only pattern taken into account for the classification is the one inside the box.

4.3.1.2 Class-Based k Nearest Neighbour

The Class Based kNN (CB-kNN) is somewhat different as a kNN extension as it does not take into account a number of neighbours around a given test pattern but a number of neighbours from each class. It was developed because often the datasets are unbalanced as regards their class structure, so it may be a case that one class has too few elements to "win" the vote of a classification of the kNN classifier.

The CB-kNN algorithm deals with these datasets by working in the following way. For every test element, the *k* nearest elements of each class are taken. The value of *k* is automatically selected by the classifier, so as to maximise the DC of the classification. Afterwards, the harmonic mean of the distances of these neighbours is calculated (so that it is not influenced so much by the most distant elements). Finally, these means are compared and the class yielding the lowest value is chosen for the classification. The detailed algorithm of this classifier can be viewed below:

Algorithm 4.2 – The CB-kNN classifier
Inputs: Input patterns of training set (P), labels of training set (T), number of neighbours (*k*), patterns of testing set (PT)
Outputs: classification vector of testing set (y)

```
1  Initialisation: set $l$=0 (class variable), $i$=0 (pattern variable), $j$=0 (another class variable), $k$=0
   (number of neighbours from each class)
2  n ← number of patterns of testing set (PT)
3  N ← number of patterns of training set (P)
4  q ← number of classes of dataset
5  do $l \leftarrow (l + 1)$
6    $C_l$ ← P(elements belonging to the $l$–th class)
7  until $l = q$
8  do $i \leftarrow (i + 1)$
9    do $j \leftarrow (j + 1)$
10     d ← vector of distances of PT($i$) to C($j$)
11     sd ← d sorted in ascending order
12     do $k \leftarrow (k + 1)$
13       apply kNN using current $k$
14       $D(k)$ ← Degree of Certainty of above classification using Eq. (3.4)
15     until $k = 20$
16     $K_i \leftarrow \arg\max_k D(k)$
17     sd ← sd(first $K$ elements)
18     $H_j$ ← harmonic mean (sd)
19   until $j = q$
20   $b \leftarrow \arg\max_j H(j)$
21   $y_i$ ← class $b$
22   $DC_i \leftarrow \text{sum}(H) / \min(H)$
23 until $i = n$
```

CB-kNN's superiority over kNN in some difficult classification instances can be viewed in the "clouds" dataset example (Fig. 4.5). The pattern of interest is indicated by a (*) and squares are used to identify the points taken into account for the classification. As the classifier estimates a K = 1, it takes into account only 1 pattern *from each class*. Since the distance of the pattern of class (+) is smaller than that of class (x), the pattern at hand is classified accurately as belonging to class (+).

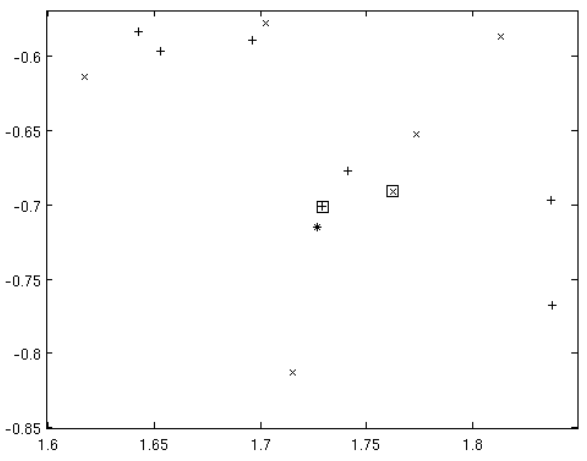

Figure 4.5 – CB-kNN's performance on the same pattern of the *clouds* dataset. CB-kNN classifies the pattern at hand correctly because instead of taking the *k* nearest neighbours, it takes *K* neighbours from each class and compares the harmonic mean of their distances to the test element. In this case, *K* is estimated by the classifier to be equal to 1. Therefore, only 1 pattern from each class is taken into account. These patterns are the ones within the boxes.

4.3.1.3 Experimental Results of kNN Extensions with Self-determined k

The experiments conducted for these kNN extensions are of the same type as those described in the previous section (i.e. they are carried out in the same datasets and for the same number of rounds, using the same evaluation criteria). Their results are as follows:

In the Bupa Liver dataset, CB-kNN was slightly better than kNN (about 0.5%) yet its reliability was slightly less. However, V-kNN had a higher reliability than all of them. Therefore there is no clear winner for this dataset, as it can be seen from Table 4.12.

Classifier	Accuracy Rate	Net Reliability	CPU Time
kNN	62.99%	0.1964	0.0000
V-kNN	62.08%	0.2188	0.0265
CB-kNN	63.59%	0.1903	0.1071

Table 4.12 – Results for the Bupa Liver dataset. There is no clear winner, although CB-kNN scored best in Accuracy Rate.

Apparently, V-kNN did not perform as well as the classic kNN in terms of accuracy rate. This is due to the nature of the dataset, particularly its structure which requires a different approach than that of V-kNN. Particularly, in this dataset V-kNN assumed an average k value of 1 for each one of the classification experiments. This means that is actually turned itself into a simple NN classifier, which is naturally less effective than a kNN one.

As regards the Pima Indians dataset, V-kNN outperformed the other two classifiers by scoring 0.85% higher accuracy rate (see Table 4.13) and a quite higher Net Reliability. Also, compared to CB-kNN, it was much faster as well. If the dataset was more complex CB-kNN would have also outperformed kNN. And if we consider the Discernibility of the dataset, as an expression of this complexity we will see that a pattern emerges. This dataset, just like the previous one (Bupa), has a relatively high Discernibility (Pima's SID is 0.6862 and Bupa's SID is 0.5855). Therefore, such a sophisticated classifier could be considered inappropriate for such a dataset.

Classifier	**Accuracy Rate**	**Net Reliability**	**CPU Time**
kNN	73.70%	0.4199	0.0000
V-kNN	74.55%	0.4707	0.1008
CB-kNN	72.70%	0.3244	0.4211

Table 4.13 – Results for the Pima Indians dataset. Unmistakably, the winning classifier is V-kNN, scoring high in both Accuracy Rate and Net Reliability. Also, it had a quite good CPU time.

Regarding the Breast Cancer Wisconsin dataset, both of the proposed classifiers performed quite well compared to kNN. CB-kNN halved the error rate of kNN, although its Net Reliability was not as high. Therefore, since V-kNN's accuracy rate was not much lower, its Net Reliability the highest of all three classifiers, and its CPU time quite low, we can safely say that this is the winning classifier for this dataset (see Table 4.13).

Classifier	**Accuracy Rate**	**Net Reliability**	**CPU Time**
kNN	94.50%	0.9039	0.0000
V-kNN	96.56%	0.9285	0.0839
CB-kNN	96.93%	0.8522	0.3691

Table 4.14 – Results for the Breast Cancer Wisconsin dataset. Although CB-kNN had the highest Accuracy Rate, V-kNN performed best, exhibiting high Accuracy Rate, the highest Net Reliability and low CPU time.

Concerning the Heart Disease dataset, the winning classifier is kNN. However V-kNN had a higher Net Reliability, yet the accuracy rate gap was much higher in the case of the classic kNN. The results for all of the classifiers are shown in Table 4.15.

Being a very simple dataset (SID = 0.8444), it appears that is requires an equally simple classifier, if we are to obtain the best possible results.

Classifier	Accuracy Rate	Net Reliability	CPU Time
kNN	80.86%	0.5649	0.0000
V-kNN	71.83%	0.5803	0.0180
CB-kNN	79.29%	0.4762	0.0877

Table 4.15 – Results for the Heart Disease dataset. The winning classifier is kNN having the highest Accuracy Rate and relatively high Net Reliability.

As concerns the Vehicle dataset (see Table 4.16), V-kNN was clearly the winning classifier, scoring a very high Accuracy Rate and a relatively high Net Reliability, without compromising much on the CPU time. This can be explained as follows: the Discernibility of this dataset is extremely low (SID gives a value of 0.0024), implying that the classes are very much overlapping. Therefore, a more sophisticated approach would be necessary if we are to obtain a good accuracy in the classification. This is demonstrated by the results of the proposed classifiers, which aim to undertake this role.

Classifier	Accuracy Rate	Net Reliability	CPU Time
kNN	70.03%	0.3899	0.0000
V-kNN	85.08%	0.3831	0.1230
CB-kNN	70.89%	0.2461	1.0179

Table 4.16– Results for the Vehicle dataset. The winning classifier is V-kNN due to the exceptionally high Accuracy Rate and the high Net Reliability. Also its CPU time was quite low.

As for the last dataset used, Boston Housing, CB-kNN is the winning classifier among the three. Its high Accuracy Rate (about 2.5% higher than that of kNN), combined with its relatively high Net Reliability renders it better than the other two (although V-kNN performs very well in comparison). However, if speed is of higher importance, V-kNN can be considered as the best classifier, since it is quite fast and it has the highest reliability as well. Also, its accuracy rate is higher than that of kNN - see Table 4.17 for details. The better performance of CB-kNN (as well as that of V-kNN) can be explained by taking into account the Discernibility concept. Particularly, the SID measure shows that the dataset is quite difficult, as it has an SID value of

0.1917. Therefore, one would expect that a simplistic classifier such as kNN would need an extra boost in its function if it is to perform well.

Classifier	Accuracy Rate	Net Reliability	CPU Time
kNN	67.57%	0.3308	0.0000
V-kNN	67.79%	0.3574	0.0496
CB-kNN	69.03%	0.3327	0.3654

Table 4.17 – Results for the Boston Housing dataset. Here there is no clear winner either. However, both V-kNN and CB-kNN performed better than kNN.

The results of the experiments for the six datasets are summarised in Table 4.18, showing the winning classifier for each evaluation criterion and its performance in brackets.

Dataset	Accuracy Rate	Net Reliability	CPU Time (2nd) in sec.
Bupa Liver	CB-kNN (63.59%)	V-kNN (0.2188)	V-kNN (0.0265)
Pima Indians	V-kNN (74.55%)	V-kNN (0.4707)	V-kNN (0.1008)
Breast Cancer W.	CB-kNN (96.93%)	V-kNN (0.9285)	V-kNN (0.0839)
Heart Disease	kNN (80.86%)	V-kNN (0.5803)	V-kNN (0.0180)
Vehicle	V-kNN (85.08%)	kNN (0.3899)	V-kNN (0.1230)
Boston Housing	CB-kNN (69.03%)	V-kNN (0.3574)	V-kNN (0.0496)

Table 4.18 – Summary results of self-determined k kNN variations based on average performance on all six datasets. The figures in the brackets show the winning performance value.

In addition, we make a one-to-one comparison for each pair of classifiers, showing the number of times one classifier performed better than the other (i.e. the number of rounds one's performance exceed the other one's). Afterwards, these scores are added up for each classifier. This is done for each one of the evaluation criteria. The final sum shows the relative performance of each classifier (see Table 4.19).

Dataset	Accuracy Rate	Net Reliability	CPU Time (2nd) in sec.
Bupa Liver	CB-kNN (74)	V-kNN (85)	V-kNN (50)
Pima Indians	V-kNN (84)	V-kNN (100)	V-kNN (50)
Breast Cancer W.	V-kNN (100)	V-kNN (100)	V-kNN (50)
Heart Disease	kNN (71)	V-kNN (92)	V-kNN (50)
Vehicle	CB-kNN (94)	kNN (86)	V-kNN (50)
Boston Housing	CB-kNN (86)	V-kNN (95)	V-kNN (50)

Table 4.19 – Summary results based on pair-wise of the self-determined k kNN variations on all six datasets. The figures in the brackets show the total number of times the winning classifier performed better than the other two, based on each particular criterion. Since each classifier is compared against two others for a total of 50 rounds, these numbers range from 0 to 100.

It is worth mentioning that in five out of the six datasets, kNN was outperformed by one of the methods proposed in this work. Also, apart from that instance, its only advantage over the others was speed, since it required no training. In Table 4.20, the average performance of all three classifiers over the datasets used can be seen.

Classifier	Accuracy Rate	Net Reliability	CPU Time
kNN	74.94%	0.4676	0.0000
V-kNN	79.16%	0.5440	0.0751
CB-kNN	75.40%	0.4037	0.3947

Table 4.20 – Average performance of the three self-determined k kNN variations based on all six datasets. The proposed classifiers have a higher Accuracy Rate than kNN and V-kNN has a higher Net Reliability as well.

From these results we can see that CB-kNN has a slightly higher accuracy rate than kNN while V-kNN outperforms them in both accuracy and reliability, significantly. Also, the CPU time of V-kNN is considerably low (although kNN is the winner in this criterion). So, it can be argued that V-kNN is the winning classifier in general.

4.3.2 Minimum Spanning Tree Classifier

4.3.2.1 Description of Classifier

The Minimum Spanning Tree Classifier (MSTC) can be seen as an extension of kNN with *k* = 2, utilizing a Minimum Spanning Tree (MST, introduced by Kruskal (1956)) representation of the training instances within each class (see Figure 4.6). The algorithm for developing these MSTs is a variation of the well-known Prim's algorithm, which is thoroughly described in textbooks, such as Baase & Van Gelder (1999).

The algorithm works as follows. Each edge in an MST is depicted as a straight line between the corresponding patterns (i.e. the patterns that are located on the two extremes of the edge) . The distance from a test pattern to an MST is defined as the Euclidean distance to the nearest edge of the MST. The classification score is the inverse of the distance to the class' MST (plus a small positive number to avoid the division by zero). Note the similarity of MSTC with CB-kNN, and yet the alternative approach of employing distances to lines connecting patterns (the MST branches), instead of merely distances to the patterns themselves. Its analytical function can be seen in Algorithm 4.3.

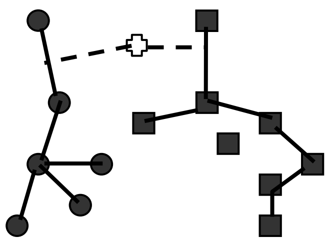

Figure 4.6 – Illustration of Minimum Spanning Tree Classifier. The two classes, comprised by patterns depicted as circles or squares respectively, are represented by their Minimum Spanning Trees. The dashed lines represent the distances from the test pattern (represented by a white cross) to its nearest MST edges.

Algorithm 4.3 – Pseudo-code for the Minimum Spanning Tree Classifier
Inputs: Input patterns of training set (P), labels of training set (T), patterns of testing set (PT)
Outputs: classification vector of testing set (y)

```
1  Initialisation: set $i = 0$ (the i-th testing pattern), $j = k = 0$ (the j-th class)
2  $n \leftarrow$ number of patterns of testing set (PT)
3  $N \leftarrow$ number of patterns of training set (P)
4  q $\leftarrow$ number of distinctly different values in T (number of classes of dataset)
5  do $j \leftarrow (j + 1)$
6     D$j \leftarrow$ distance matrix for patterns of class $j$
7     $C_j \leftarrow$ connections matrix of MST of class $j$, based on $D_j$ (variation of Prim algorithm is used)
8  until $j =$ q
9  do $i \leftarrow (i + 1)$
10    do $k \leftarrow (k + 1)$
11    D $\leftarrow$ minimum distance of pattern $i$ to class $k$
12    ind $\leftarrow$ first pattern for which d = minimum
```

```
13  F1 ← pattern for which a branch of the MST of class $k$ begins from ind, based on $C_k$
14  F2 ← pattern for which a branch of the MST of class $k$ ends to ind, based on $C_k$
15  dtemp ← distance vector of pattern $i$ to patterns of f1 and f2
16  D2 ← shortest distance of $i$ to the group of patterns of class $k$, connected to ind
17  ind2 ← pattern for which d2 = minimum
18  D3 ← distance between ind and ind2 (these are the patterns of the closest branch)
19  if $d2 \geq \text{sqrt}((d1)^2 + (d3)^2)$
20    d ← d1 (the closest part of the branch is the pattern in the closest end of the branch)
21  Else
22    t ← (d1 + d2 + d3) / 2 (semi-circumference of the triangle formed by the test pattern ($i$) and
    the patterns of the closest branch)
23    area ← sqrt(t.(t-d1).(t-d2).(t-d3)) (Heron's method of calculating the area of the triangle)
24    d ← 2.area / d3 (height of the triangle, which by definition is equal to the distance of $i$ to the
    closest branch)
25  end of if
26  $p_j$ ← 1 / (d + eps) (where $p_k$ = voting score for class $k$, and eps is a very small number, to avoid
    division by zero)
27  until $k = q$
28  IND ← class for which p = maximum
29  $y_i$ ← class IND (classification of pattern $i$)
30 until $i$ = n
```

The main advantage of the MSTC is that it views each class as an entity instead as an amalgamation of points. This allows it to "sense" links between pairs of patterns that would be otherwise invisible to a transductive classifier (a classifier that uses a direct associationg between the testing patterns and the training ones), such as kNN and its extensions. Also, it is a relatively diverse classifier, not limited to a particular type of dataset. However, its Net Reliability (for a definition, see Section 3.3) is relatively low, in general, as the way Degree of Certainty is applied on its classification rule is often misleading due to the great variance of the distances involved. In addition, it may be slow when it is applied on very large dataset in terms of number of patterns. Overall, the MSTC is a robust classification system for relatively small datasets, particularly ones having a lot of noise. Also, it performs best when the patterns of each one of the classes of the dataset form one cluster. Furthermore, it may be rather useful in problems tackled with the semi-supervised learning approach.

4.3.2.2 Experimental Results of Minimum Spanning Tree Classifier

Some experiments were conducted to test the performance of the MSTC. These were carried out in five different benchmark datasets from the UCI repository (UCI Repository, 2008): Wine, Glass, Bupa Liver, Vehicle and Vowel. The experiments comprised of 50 rounds of 10-fold cross-validation and involved the Minimum Spanning Tree and k Nearest Neighbour classifiers. The evaluation criteria used were Accuracy Rate and CPU time. The results are shown in Table 4.21.

Classifier	**KNN**		**MSTC**	
Dataset	**Accuracy Rate**	**CPU Time**	**Accuracy Rate**	**CPU Time**
Wine	95.26%	0.0045	95.37%	0.0345
Glass	66.74%	0.0055	69.59%	0.0390
Bupa Liver	60.88%	0.0091	61.51%	0.2929
Vehicle	70.08%	0.0442	70.08%	0.9758
Vowel	91.55%	0.0146	94.87%	0.1725

Table 4.21 – Average performance of the kNN and MST classifiers based on five datasets. The proposed classifier generally has a slightly higher Accuracy Rate than kNN although it is somewhat slower.

From the above results we can observe that MSTC performs at least as well as the kNN classifier. One drawback of this method is that it requires a lot of CPU time, mainly due to the fact that the construction of a MST is a computationally expensive process. Yet, if our primary concern is the accuracy rate, it is worth the effort, particularly when dealing with "difficult" datasets, such as *glass*.

4.4 Summary and Contribution of the Chapter

In this chapter we presented the two classifiers that are based on or related to the Discernibility concept, as well as three other classifiers based on the kNN philosophy. Their function was explained and their behaviour was investigated experimentally. The experiments conducted show that the developed methods generally outperform kNN.

This chapter also contributed a few alternative approaches to improving the kNN classifier and also proposed another distance-based classifier similar to kNN, namely the Minimum Spanning Tree classifier. In addition, the advantages and limitations of the proposed methods are exhibited.

This chapter contributes to the thesis by showing how the Discernibility concept can be used to enhance the performance of a classifier and explain its behaviour on different types of datasets.

Chapter 5 – Discernibility-Based Methods of Data Processing

One of the most practical applications of the Discernibility concept is that of data processing. Particularly, the evaluation of (a part of) a dataset can be used to assess the importance of features and patterns and therefore clear the dataset of the features or patterns which are considered less useful for the classification process. In the first case the application of Discernibility takes the form of a "filter" or a filtering process, used to find the most discernible features (or combination of features), discarding all the others (feature selection). In the second case, it involves finding the most discernible patterns (i.e. those having the highest discernibility) and by taking into account the distances among them removing many of them (data reduction).

5.1 Feature Selection and Discernibility

Feature selection, the process of finding the best features within a dataset and discarding the rest, has been a popular topic over the past few years, as it promises better performance (particularly in terms of speed) and reduced complexity, something very significant in classification, among other fields of application. In addition, sometimes the accuracy rate is increased when the classifier is applied on the reduced feature set. This renders feature selection an important technique for classification problems with high dimensional data.

Feature selection is a special type of dimensionality reduction. The latter has been accomplished using Genetic Algorithms (Frohlich & Chapelle, 2003; Lecocke & Hess, 2005), Locally Linear Embedding (LLE) (Chao & Lihui, 2005), Recursive Salient Analysis (RSA) (Cao et al., 2003), the selection of a support set (Alexe et al., 2005), a combination of PCA and the UKW clustering algorithm (Tasoulis et al., 2006), *t*-tests (Lecocke & Hess, 2005; Mukkamala et al., 2005), Regression Splines (MARS) (Mukkamala et al., 2005), Classification and Regression Trees (CART) (Mukkamala et al., 2005), Random Forests (Mukkamala et al., 2005), Linear Genetic Programs (LGP) (Mukkamala et al., 2005), Neural Networks as a similarity measure (Sawa & Ohno-Machado, 2003), and clustering analysis (Alon et al., 1999).

Method employing Feature Selection in particular include approaches using statistics (Hochreiter & Obermay, 2003; Michalak & Kwa/Snicka, 2006; Liu et. al, 2002), fuzzy logic (Shen & Jensen, 2008; Shang & Shen, 2006; Jensen & Shen, 2005) as well as several other approaches (for an overview see (Dash & Liu, 1997; Hall, 1999)).

The key difference between feature selection and other techniques of dimensionality reduction, which is also the edge of the former, is that it preserves the semantics by merely selecting the most important features and discarding all the rest. Other methods of dimensionality reduction transform the feature space, rendering the new feature set challenging to understand and interpret.

This is why feature selection is a popular choice for various datasets, especially ones of high dimensionality, such as those encountered in bioinformatics applications. In those cases feature selection is not merely a plausible but often an essential part of the classification process as it alleviates the classifier(s) from a lot of unnecessary data, rendering the whole process more efficient. To this end, over the last few years a series of methods for feature selection have been introduced, most of them focusing on microarray data (Frohlich & Chapelle, 2003; Chao & Lihui, 2005; Weston et al., 2001; Lecocke & Hess, 2005 ; Alexe et al., 2005; Tasoulis et al., 2006; Mukkamala et al., 2005; Sawa & Ohno-Machado, 2003; Alon et al., 1999), where the number of data points (patterns) is small whilst the dimensionality of the features is very high.

Feature selection methods used in the literature have been applied successfully with SVMs (Frohlich & Chapelle, 2003; Chao & Lihui, 2005; Weston et al., 2001), which benefit greatly by the reduction of feature space. This is because their performance depends on the dimensionality of the data they are applied on, and a relatively high number of features compromises their performance due to overfitting (Duda et al., 2001).

Other approaches have been also considered (Lecocke & Hess, 2005; Alexe et al., 2005 ; Tasoulis et al., 2006; Mukkamala et al., 2005; Sawa & Ohno-Machado, 2003; Alon et al., 1999) with success as well. However, the underlying problem that all of the above methods have is that there is no particular stopping criterion for the feature selection method. In other words, the reduced feature sets come in a variety of sizes with limited a priori knowledge on the quality of the selected features.

Based on the Index of Discernibility (Spherical version) we have developed two independent feature selection methods. Although of similar philosophy, they differ in their structure and in the way they employ the Index of Discernibility.

5.1.1 Discernibility-based Feature Selection: the IFF Method

The first method, IFF (Individual Feature Filtering), is the simplest and the fastest in its function. As it can be seen from its flow chart in Fig. 5.1, it makes an assessment of each one of the features of the dataset, using the Index of Discernibility, and then selects the ones, which are of a given standard. The latter is expressed by the threshold *th* which is given by the user (this is found empirically although any value between 0.5 and 1.0 is generally good, with the higher yielding less features). Note that since this is an absolute parameter, the number of features at the new feature set heavily depends on the dataset itself. However, a value ranging from 0.7 to 0.85 is a good choice for the *th* parameter for most problems. This parameter is set by the user in the Initialisation stage, along with the definition of the dataset (Fig. 5.1). The outputs of this method are the reduced feature set (P2), as well as a list of the indexes (names) of these features (NFS). Note that the initial number of features (na) is obtained by MATLAB using a built-in function.

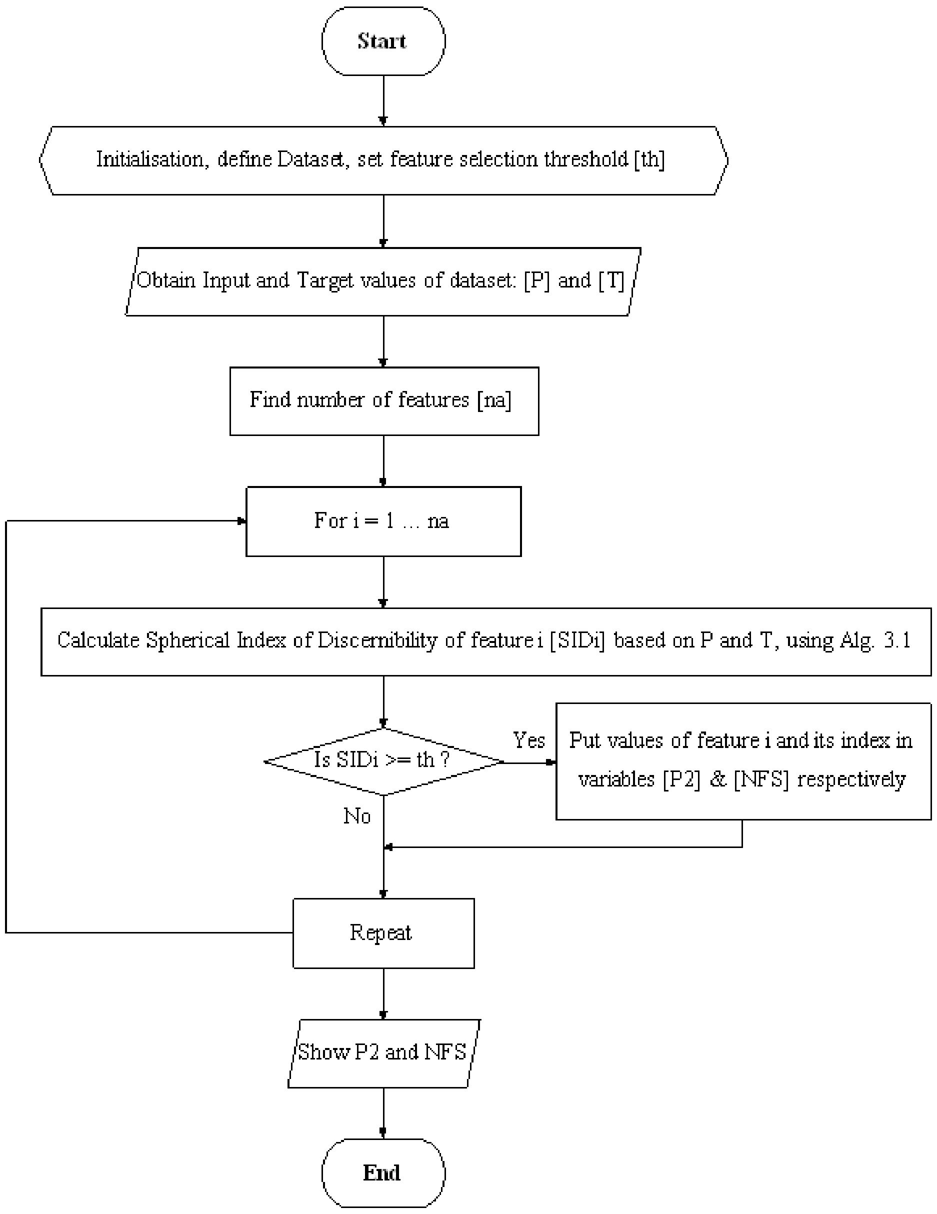
Start
Initialisation, define Dataset, set feature selection threshold [th]
Obtain Input and Target values of dataset: [P] and [T]
Find number of features [na]
For i = 1 ... na
Calculate Spherical Index of Discernibility of feature i [SIDi] based on P and T, using Alg. 3.1
Is SIDi >= th ?
Yes
Put values of feature i and its index in variables [P2] & [NFS] respectively
No
Repeat
Show P2 and NFS
End

Fig. 5.1. Flow chart of the first ID-based feature selection method (IFF).

5.1.2 Discernibility-based Feature Selection: the GFS Method

The second method, GFS (Group Feature Selector), is quite different in its function and somewhat more complex. However, it has the advantage that it is entirely automatic, as it has no need for a threshold parameter for the selection of the features. As it can be seen for Fig. 5.2, where its flow chart is shown, it gradually builds the new feature set (P2) by initially taking the first feature of the dataset and then adding two features at a time, so as to maximise the Spherical Index of Discernibility of the whole feature set. In other words, the features are not evaluated one by one, but as a group, something which although more time-consuming, is a better way of selecting the features, since it takes into account their relationship. Afterwards, the algorithm checks if by removing one of the features of P2 the Spherical Index of Discernibility is increased or at least remains the same. This step helps prevent the accumulation of redundant features in the new feature set. By repeating this process (adding 2 new features and removing 1, if there is a redundancy) until the Spherical Index of Discernibility ceases to increase, this method goes through the rest of the available features of the original feature set. Alike the previous method, its outputs are the reduced feature set (P2), as well as a list of the indexes of these features (NFS). This method tends to yield smaller reduced feature sets and often takes longer, due to the increased number of Discernibility calculations that are required. A high-level description of this method is exhibited in Algorithm 5.1.

Algorithm 5.1 – General description of the GFS method for Discernibility-based Feature Selection
Inputs: Input data (P), Data Labels (T)
Outputs: Selection of best set of features (P2), indexes of these features (NFS)

```
1 Find the Discernibility of the original feature set using Eq. (3.1)
2 Get the first feature of P and the corresponding label from T
3 Find the Discernibility of each one of the combinations of the features acquired so far and all the
  other ones, using Eq. (3.1)
4 Keep the feature that maximises the Discernibility of the pair
5 Check if by removing any one of the features gathered so far is unnecessary (i.e. the
  Discernibility of the new feature set is the same or higher if this feature is removed
6 Get another feature, so as the Discernibility of the new feature set is maximised (i.e. repeat what
  was done in Steps 3 and 4). Store the selected features in P2 and their indexes in NFS
7 Repeat steps 3-6 while the Discernibility of the acquired features is smaller than that of the
  original feature set (calculated in Step 1)
8 Output new feature set (P2) and the indexes of the selected features (NFS)
```

Generally, this method is quite simple to implement and works well with a number of different classifiers. Also, it is independent on the dataset it is applied on.

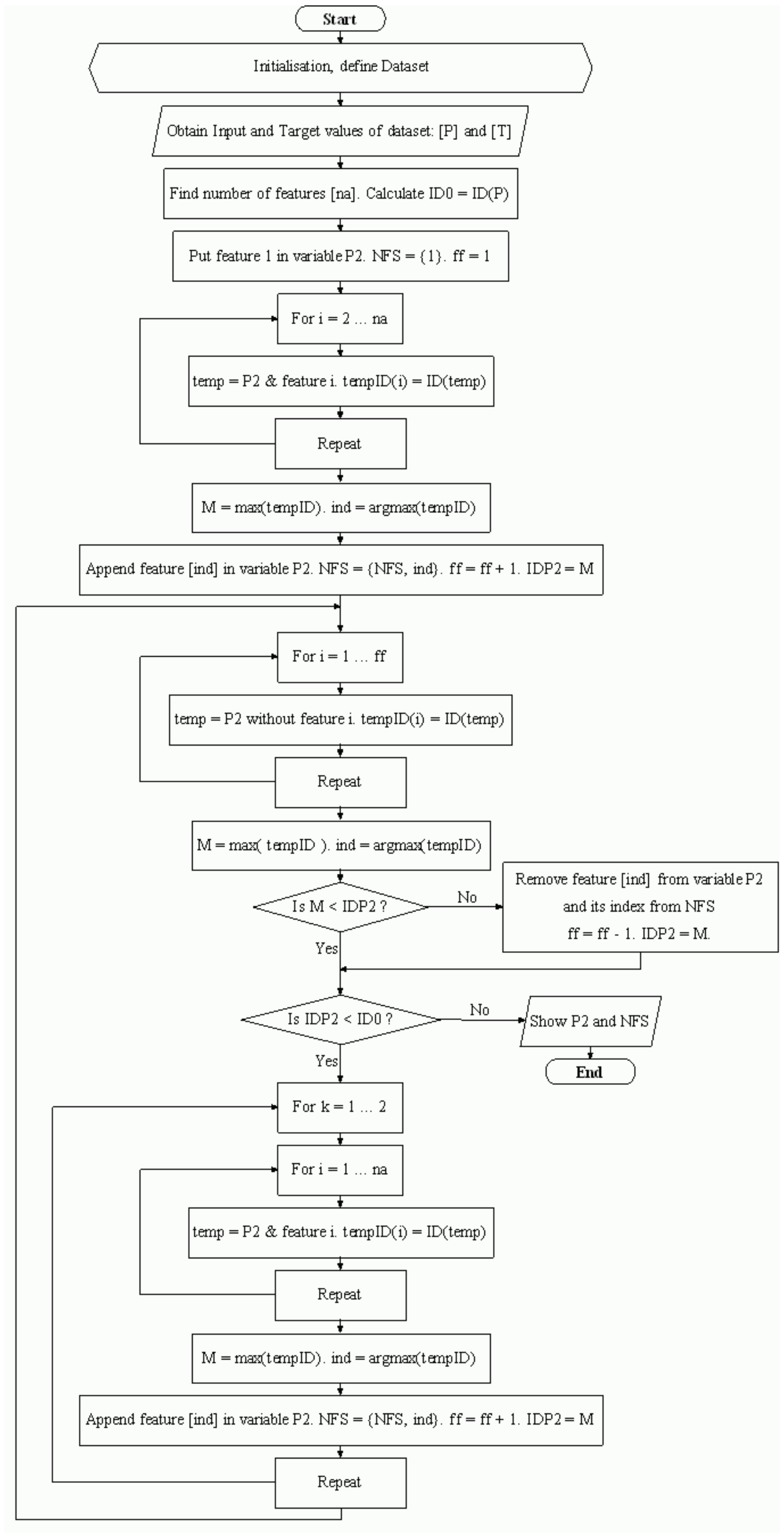
Start
Initialisation, define Dataset
Obtain Input and Target values of dataset: [P] and [T]
Find number of features [na]. Calculate ID0 = ID(P)
Put feature 1 in variable P2. NFS = {1}. ff = 1
For i = 2 ... na
temp = P2 & feature i. tempID(i) = ID(temp)
Repeat
M = max(tempID). ind = argmax(tempID)
Append feature [ind] in variable P2. NFS = {NFS, ind}. ff = ff + 1. IDP2 = M
For i = 1 ... ff
temp = P2 without feature i. tempID(i) = ID(temp)
Repeat
M = max(tempID). ind = argmax(tempID)
Is M < IDP2 ?
No
Remove feature [ind] from variable P2 and its index from NFS ff = ff - 1. IDP2 = M.
Yes
Is IDP2 < ID0 ?
No
Show P2 and NFS
End
Yes
For k = 1 ... 2
For i = 1 ... na
temp = P2 & feature i. tempID(i) = ID(temp)
Repeat
M = max(tempID). ind = argmax(tempID)
Append feature [ind] in variable P2. NFS = {NFS, ind}. ff = ff + 1. IDP2 = M
Repeat

Fig. 5.2. Flow chart of the second Discernibility-based feature selection method (GFS). Note that although this method uses the Spherical Index of Discernibility, it can also employ the Harmonic ID, which is why the term ID is used in the flowchart.

5.2 Data Reduction and Discernibility

The past five decades have been revolutionary in the way in which data acquisition has developed (Skyt et al., 2008). This facilitated the accumulation of data, in both resolution (high number of features) and size (large number of patterns), rendering today's databases on the order of terabytes (Skyt et al., 2008). This gave rise to a constantly growing need for data reduction as well as the adoption of an eclectic attitude towards the attributes of certain datasets, particularly those containing microarray data (Voulgaris & Magoulas, 2008b). In this section we explore how we can achieve data reduction using the novel concept of Discernibility that we have introduced previously.

According to Li and Jacob (2008), there are two distinct approaches to Data Reduction: the general purpose Data Reduction and the task-specific Data Reduction. Since in this research we are dealing with classification, we will focus on the latter approach (which as one would expect, yields better results for the classification task). Contrary to what one would expect the reduced datasets often exhibit better classification performance (accuracy rate). This is probably due to the reduction in complexity, which in the original dataset takes the form of redundant patterns and useless outliers as well as the removal of noisy patterns in some cases.

Data reduction can be accomplished using the Discernibility concept by assessing how "easily" distinguishable the various patterns of a dataset are, and then removing the ones having the highest discernibility, taking into account their distances to the other patterns of their classes. Contrary to the feature reduction techniques, in this application of Discernibility it is unwise to eliminate patterns below a given threshold, since there is a strong inter-dependency among all of them, as regards their Discernibility status and by removing the patterns below a certain threshold, the dataset loses its original geometric structure. Therefore, the patterns to be removed have to be selected carefully; otherwise there is a risk of distorting the class structure of the dataset (resulting to a drop in the accuracy rate of the classifiers). In the method developed here, this is accomplished by using Discernibility along with the distances between the removed patterns. In other words, we remove not only the most discernible patterns (i.e. the ones the are "easy" to classify and therefore relatively redundant), but also the ones that are as far as possible from the ones already removed. An analytical description of the operation of this method can be seen in the flowchart of Fig. 5.3.

From this flow chart, one can observe that the two factors mentioned earlier –, the patterns' Discernibility scores (Z) and their distances (DD) to the other patterns that are to be discarded – are taken into account with equal weight since they are both equally important. Therefore, a removed pattern has to be easy to distinguish (i.e. have a high Discernibility) and be relatively far away from other patterns that will be removed (have a large distance from them). This allows the reduction of the patterns to be "smooth" and balanced, since the removed patterns are taken from the whole dataset space (due to the distance criterion) and thin out the denser areas of each class (Discernibility criterion).

The data reduction method is fine-tuned using a particular threshold parameter (th), which refers to the reduction ratio. This parameter is equal to the amount of patterns that are removed from the original dataset.

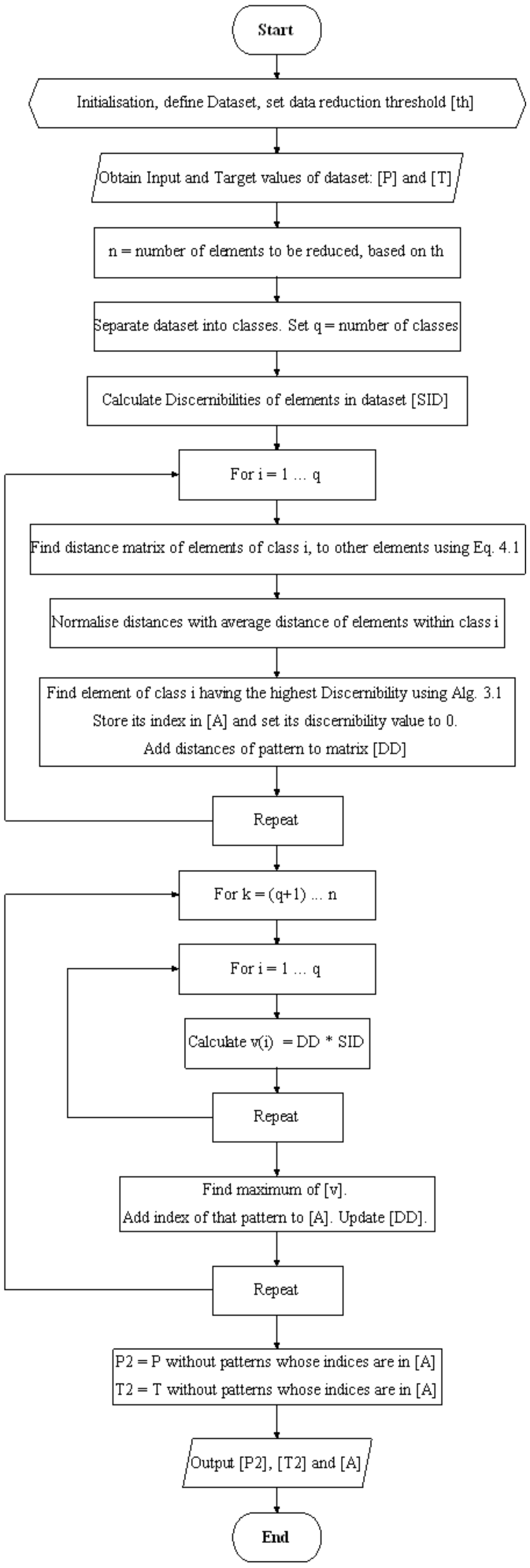
Start
Initialisation, define Dataset, set data reduction threshold [th]
Obtain Input and Target values of dataset: [P] and [T]
n = number of elements to be reduced, based on th
Separate dataset into classes. Set q = number of classes
Calculate Discernibilities of elements in dataset [SID]
For i = 1 ... q
Find distance matrix of elements of class i, to other elements using Eq. 4.1
Normalise distances with average distance of elements within class i
Find element of class i having the highest Discernibility using Alg. 3.1
Store its index in [A] and set its discernibility value to 0.
Add distances of pattern to matrix [DD]
Repeat
For k = (q+1) ... n
For i = 1 ... q
Calculate v(i) = DD * SID
Repeat
Find maximum of [v].
Add index of that pattern to [A]. Update [DD].
Repeat
P2 = P without patterns whose indices are in [A]
T2 = T without patterns whose indices are in [A]
Output [P2], [T2] and [A]
End

Fig. 5.3. Flow chart of the Discernibility-based data reduction method.

5.3 Experimental Results

5.3.1 Experimental Setup

The above methods have been implemented in MATLAB and tested thoroughly on a number of datasets. A number of different classifiers were used, in order to demonstrate the methods' independence to the classification process itself, in terms of performance.

The classifiers used in the experiments were the k Nearest Neighbour and two of its modifications, the Linear Discriminant Analysis (LDA) method, described in Duda et al (2001) as well as Fidler & Leonardis (2003), the Gravity Model Classifier (GMC) which was initially proposed by Ruta & Gabrys (2003), the Fuzzy kNN method (Keller et al., 1985), the Decision Tree C4.5 algorithm, the Reduced Coulomb Energy classifier (RCE), as described in Duda et al (2001) and the Minimum Spanning Tree Classifier (MSTC), which was presented in Section 4.3.2.

The kNN algorithm classifies a pattern according the plurality vote among its k nearest neighbours from the training set: the pattern is assigned with the winning class; if there is a tie, the minimum index wins. The classifier scoring function, for a class, is the number of those of the k neighbours that belong to the class. In our experiments, we take the *k* to be equal to 5, as our experience shows that this a good choice for the neighbours parameter for these datasets.

The kNN extensions used in these experiments are V-kNN (for the feature selection experiments) and the Fuzzy kNN. The first of them was described thoroughly in Chapter 4.4 and was chosen due to its promising performance and high classification speed. The second kNN extension was introduced by Keller et al. (1985) and is a good demonstration of a Fuzzy Logic classification system, based on the kNN paradigm. This classifier is more thoroughly described in Section 2.1.4.

The LDA algorithm implements Fisher's linear decision rule (Fisher, 1938) by deriving a separating hyperplane for each class to minimise the ratio of the "within-class" average error over the "out-of-class" average error. Its scoring function, for each class, is the value of the separating linear rule derived for the class if it is positive, or zero if it is negative. This rule is based on the conditional probability f_i that a test pattern *k* belongs to a particular class *i*, calculated with Eq. 5.1. The class yielding the highest probability value f_i wins the classification.

$f_i = \mu_i C^{-1} x_k^T - \frac{1}{2} \mu_i C^{-1} \mu_i^T + \ln(p_i)$	Eq. 5.1

where μ_i is the average feature values of the patterns of class i,

x_k is the vector of the feature values of pattern k,

C is the covariance matrix, and

p_i is the prior probability related to class i, which is estimated based on the number of patterns in the class.

The function of LDA is described in Algorithm 5.2.

Algorithm 5.2 – Pseudo-code of LDA classifier
Inputs: Input patterns of training set (P), labels of training set (T), patters of testing set (PT)
Outputs: classification vector of testing set (y)

```
1  Initialisation: Set $i = 0, j = 0$
2  N ← number of data elements in P
3  q ← number of unique elements in T (classes)
4  na ← number of attributes in P
5  remove attributes that have variance of 0 (if a is their number, na ← na – a)
4  do $i \leftarrow (i + 1)$
5    find patterns of class $i$ ($X_i$) and count them ($n_i$)
6    calculate the class probability of class $i$: $p_i \leftarrow n_i / N$
7    calculate the mean of class $i$: $m_i \leftarrow sum(p_i) / n_i$
8  until $i = q$
9  calculate the global mean: $M \leftarrow p*m$
10 do $j \leftarrow j + 1$
11   find distance of each pattern of class $j$, to the class's mean: $XX \leftarrow X_j - m_j$
12   compute covariance matrix for class $j$: $c \leftarrow XX^T*XX / n_i$
13   compute within group variance: $C \leftarrow C + c*p_j$
14 until $j = q$
15 calculate the inverse matrix of C: $CC \leftarrow C^{-1}$
16 find most significant eigenvectors: $A \leftarrow -0.5 * diag(m*CC*m^T) + log(p)^T$
17 calculate probability values for different classes: $f \leftarrow m*CC*PT^T + A$
18 $Q \leftarrow j$ value for which $f_j$ is maximised
19 classify all patterns to classes depicted by Q
```

The GMC algorithm used closely follows the method proposed by Ruta and Gabrys (2003). Given a test pattern, its squared Euclidean distances to all training instances are computed, inversed (with an added very small positive to avoid divisions by zero) and summed up within the classes. A within-class sum represents the class' gravity force. The class whose force is maximal wins and is assigned to the pattern. The forces form the classification scores.

The C4.5 method used is the popular Quinlan's algorithm that draws a decision tree over the training set to minimise the entropy between the tree leaf partition and class-partition. Its default parameter is the split stopping threshold (*th* = 10% of alien entities in a node). We have taken the scoring function of this decision rule to be constant and equal to 1 – th = 0.9. This is a very good estimate of the confidence of the classifier, since this is the expected probability of a correct classification, based on the training set (which defines the branches of the decision tree).

The RCE algorithm surrounds each of the training patterns by a sphere of the maximum radius satisfying the property that no training patterns belonging to different classes belong to the sphere. (The radius is defined, thus, as the minimum distance between the current pattern and a pattern from a different class minus a small positive real, typically about 10^{-4}.) Given a test pattern, the classification score of a class is the number of training patterns from the class, whose spheres contain the test pattern.

The MSTC method is a classifier already described in section 4.3.

In the classifiers utilising the between-pattern distances, the (whole) dataset is pre-normalised in such a way that, for each feature, its minimum is zero and the maximum is unity.

The datasets used for feature selection and data reduction were selected so as to be suitable for the application of these techniques. In the first case they are four datasets taken from a bio-medical repository (Kent Ridge Bio-medical Data Set Repository, 2008) and are representative of a class of bioinformatics applications that employ microarray data. These datasets are Lymphoma, Colon, Leukaemia, Prostate, and Brain Cancer. All of them exhibit a very high dimensionality, while the number of patterns is quite limited. This makes the feature selection process more necessary as well as more challenging.

For the data reduction experiments, we made use of two large datasets from UCI repository (UCI Repository, 2008) and the *clouds* dataset from (ELENA Project Artificial Databases, 2008). The datasets from UCI repository are coded as *pendigits* and *magic*. All of these datasets exhibit a large number of patterns. A larger size would render them too large to manage using MATLAB, given the memory restrictions of the computers used. In all of the experiments conducted in this chapter, 50 rounds of 10-fold cross-validation were carried out.

5.3.2 Feature Selection Experiments

5.3.2.1 Results of Feature Selection Methods

The aim of this set of experiments was to investigate whether the methods can cope with different datasets without fine tuning. So we decided not to experiment with different threshold values in order to establish some kind of "optimum" threshold for each dataset but to set the threshold value used in the IFF method equal to 0.75 (this means that any feature having an SID less than 0.75 is omitted from the new feature set). Although this may not be the optimum value, results exhibited in Tables 5.2 – 5.11 were quite promising in general.

As it can be seen from Table 5.1, the reduction of the feature set for each one of the four datasets is dramatic. Particularly after the application of the GFS method, the new feature sets are tiny compared to the original ones. This is translated to smaller complexity of the datasets (as it will be seen later on), and significantly less need for storage space.

Dataset	No. of Patterns	Original no. of Features	Features after IFF	Features after GFS
DLBCL	47	4026	45	4
Colon	62	2000	11	4
CNS	60	7129	19	4
Leukaemia	72	7129	112	2

Table 5.1. Datasets characteristics and sizes of reduced feature sets

5.3.2.2 Classification Experiments with the Reduced Feature Sets

In this set of experiments we investigated whether the use of the reduced feature sets affects the quality of the classification. To this end we used a variety of classifiers and evaluation measures.

Classification results with the original feature sets are presented in Tables 5.2-5.6. The accuracy rate in all these experiments is computed for the test set. Note that for the LDA classifier, there are no results for the CNS and Leukaemia datasets. This is because due to very high computational demands, the system could not complete the relevant experiments.

As regards the overhead of the feature selection process, for the IFF algorithm, the CPU time ranged from 6 to 23 seconds (depending on the dataset), while for the GFS method, from 24 to 104 seconds.

Dataset	Accuracy Rate	Degree of Certainty	CPU Time (sec)
DLBCL	0.5532	0.6511	1.0100
Colon	0.8387	0.8516	0.8900
CNS	0.6667	0.6767	3.0500
Leukaemia	0.9306	0.8861	4.6800

Table 5.2. Results for kNN using the original feature set

Dataset	Accuracy Rate	Degree of Certainty	CPU Time (sec)
DLBCL	0.4894	0.8237	16.880
Colon	0.7903	0.9530	15.820
CNS	0.6333	0.8263	76.520
Leukaemia	0.9028	0.9746	144.45

Table 5.3. Results for V-kNN using the original feature set

Dataset	Accuracy Rate	Degree of Certainty	CPU Time (sec)
DLBCL	0.5532	0.6548	0.9600
Colon	0.8548	0.8561	0.7200
CNS	0.6667	0.6806	3.1900
Leukaemia	0.9306	0.8871	4.7400

Table 5.4. Results for Fuzzy kNN Using the original feature set

Dataset	Accuracy Rate	Degree of Certainty	CPU Time (sec)
DLBCL	0.6383	0.7234	5381.4
Colon	0.5161	0.7736	882.40
CNS	-	-	-
Leukaemia	-	-	-

Table 5.5. Results for LDA using the original feature set

Dataset	Accuracy Rate	Degree of Certainty	CPU Time (sec)
DLBCL	0.5106	0.5771	1.7400
Colon	0.6452	0.6405	1.5300
CNS	0.6500	0.6227	5.7100
Leukaemia	0.6528	0.6926	8.3500

Table 5.6. Results for GMC using the original feature set

The results of the experiments using the reduced feature sets are exhibited in Tables 5.7-5.16 and provide the average of the 50 runs. The first five of them show the results using the IFF method whilst the rest using the GFS method.

Dataset	Accuracy Rate	Degree of Certainty	CPU Time (sec)
DLBCL	0.7234	0.8936	0.0600
Colon	0.8548	0.8871	0.0500
CNS	0.7167	0.7800	0.1200
Leukaemia	0.9444	0.9639	0.1700

Table 5.7. Results for kNN using the reduced feature set created by IFF

Dataset	Accuracy Rate	Degree of Certainty	CPU Time (sec)
DLBCL	0.6809	0.9303	1.2900
Colon	0.9032	0.9480	2.2900
CNS	0.6833	0.8601	2.2200
Leukaemia	0.9444	0.9888	3.5500

Table 5.8. Results for V-kNN using the reduced feature set created by IFF

Dataset	Accuracy Rate	Degree of Certainty	CPU Time (sec)
DLBCL	0.8723	0.8990	0.0900
Colon	0.9032	0.9011	0.0600
CNS	0.7000	0.7821	0.1400
Leukaemia	0.9444	0.9624	0.1900

Table 5.9. Results for Fuzzy kNN using the reduced feature set created by IFF

Dataset	Accuracy Rate	Degree of Certainty	CPU Time (sec)
DLBCL	0.4255	0.7553	0.2000
Colon	0.7742	0.8104	0.1900
CNS	0.5667	0.6945	0.1100
Leukaemia	0.4861	0.7335	0.2600

Table 5.10. Results for LDA using the reduced feature set created by IFF

Dataset	Accuracy Rate	Degree of Certainty	CPU Time (sec)
DLBCL	0.7234	0.6951	0.1300
Colon	0.8387	0.6899	0.2500
CNS	0.6667	0.6437	0.2200
Leukaemia	0.6528	0.7954	0.3400

Table 5.11. Results for GMC using the reduced feature set created by IFF

Dataset	Accuracy Rate	Degree of Certainty	CPU Time (sec)
DLBCL	0.9362	0.8894	0.0700
Colon	0.8226	0.8710	0.0600
CNS	0.7833	0.7400	0.0900
Leukaemia	0.9861	0.9861	0.1400

Table 5.12. Results for kNN using the reduced feature set created by GFS

Dataset	Accuracy Rate	Degree of Certainty	CPU Time (sec)
DLBCL	0.9574	0.9811	1.2700
Colon	0.7581	0.9568	2.3200
CNS	0.7667	0.9285	2.1700
Leukaemia	1.0000	1.0000	3.0300

Table 5.13. Results for V-kNN using the reduced feature set created by GFS

Dataset	Accuracy Rate	Degree of Certainty	CPU Time (sec)
DLBCL	0.9362	0.9246	0.1000
Colon	0.8548	0.8915	0.0600
CNS	0.8500	0.7925	0.1100
Leukaemia	0.9861	0.9885	0.1800

Table 5.14. Results for Fuzzy kNN using the reduced feature set created by GFS

Dataset	Accuracy Rate	Degree of Certainty	CPU Time (sec)
DLBCL	0.8085	0.7872	0.1300
Colon	0.7903	0.8304	0.1900
CNS	0.6833	0.6896	0.0700
Leukaemia	0.9167	0.9004	0.0700

Table 5.15. Results for LDA using the reduced feature set created by GFS

Dataset	Accuracy Rate	Degree of Certainty	CPU Time (sec)
DLBCL	0.9574	0.7384	0.0300
Colon	0.7419	0.7349	0.2200
CNS	0.7333	0.6770	0.2100
Leukaemia	0.9722	0.9036	0.2500

Table 5.16. Results for GMC using the reduced feature set created by GFS

As it can be observed from Tables 5.7-5.16 the average CPU time of the classification is significantly reduced. This is very important, considering that the feature selection itself takes a not neglectible amount of time. However, the reduction of the time involved in the classification (due to the simplicity of the new dataset) might make it worthwhile. Also, as one would expect, the second feature selection method (GFS) takes considerably more time, for all of the datasets. Yet, this is understandable, as it is a more complicated method, working with groups of features instead of single features, at a time.

Another important point is that the average accuracy rate is significantly increased in most of the datasets, for most of the classifiers. This is something expected, since the original feature set contains a large number of useless features which not only do not aid the classification, but for many of the classifiers, they make it more difficult due to the additional noise they often contain. So, by eliminating these features we end up with a relatively “easier” and noise-free dataset to classify. This change is also depicted at the SID, which often is larger for the new datasets, something that rarely happens for smaller datasets (on the contrary, if from a dataset having a small number of features you diminish the feature set by merely one feature, the ID of the whole is bound to drop). This can be seen in Table 5.17.

Dataset	DLBCL	Colon	CNS	Leukaemia
Original feature set	0.7872	0.8226	0.3833	0.8333
Reduced feature set (IFF)	0.9362	0.8871	0.6500	0.9306
Reduced feature set (GFS)	0.8936	0.8871	0.8667	1.0000

Table 5.17. Index of Discernibility values for each dataset, before and after the feature selection process, for both of the methods employed

The fact that the datasets become simpler is backed up by another point, which can be observed in Tables 5.2-5.16: the increase in the average Degree of Certainty, for almost all of the classifiers, for the various datasets. Particularly in the DLBCL dataset, the increase is quite significant.

Also it is noteworthy that the feature selection methods introduced allowed the LDA classifier to be applied to the CNS and Leukaemia datasets, which were unmanageable with the original feature set. This is also important when considering that many of the produced microarray datasets are of this dimensionality or even of a higher one.

A summary of the above results depicting which feature set yielded the highest performance, for each one of the datasets and for each one of the classifiers is provided in Table 5.18. Note that a feature set was considered as yielding the highest performance, for a particular classifier, when it outperformed the other feature sets in two or more of the evaluation measures used.

Classifier/Data	DLBC	Colon	CNS	Leukaemia
kNN	GFS	IFF	GFS	GFS
V-kNN	GFS	IFF	GFS	GFS
Fuzzy kNN	GFS	IFF	GFS	GFS
LDA	GFS	GFS	GFS	GFS
GMC	GFS	IFF	GFS	GFS

Table 15.8. Summary of results

From Table 5.18, it can be observed that the reduced feature sets of both of the feature selection methods introduced here outperformed the original feature set, for all of the classifiers used. Particularly the feature set of the second method, GFS, dominated the other ones for three of the datasets, in spite of the fact that it is slower than the other feature selection method. However, as the reduced feature sets it created were significantly smaller, it managed to yield quite low CPU times on average.

The feature selection of the method IFF appeared to be weaker in the case of LDA for one particular dataset (DLBCL), however the reduced feature set it yielded for this dataset seemed to work very well with all the other classifier. So the weakness was because of a problematic generalisation of the LDA classifier, probably due to the presence of one or more useless features in the reduced feature set. So this raises the issue of whether the threshold of 0.75 for the IFF method is reliable. It could be the case that different datasets require different threshold values in order to yield appropriately reduced feature sets. This would render the IFF method a quite flexible alternative, which when fine-tuned, could perform equally well to the GFS method.

5.3.3 Experimental Results for Data Reduction

For this series of experiments, we tested the data reduction method with three fairly large datasets and applied four different threshold values (1/10, 1/4, 1/3 and 1/2), so as to explore how sensitive the performance of the classifiers is in relation to the reduction rate. The experiments comprised of 30 rounds of 10-fold cross validation. The datasets used were *clouds*, which contains 5000, 2-dimensional patterns, the *pendigits* (10992, 16-dimensional patterns) and *magic* (19020, 11-dimensional). We experimented with a few different classifiers, measuring for each one of them the Accuracy Rate before and after the data reduction, as well as the CPU time taken for each classification. The data reduction overhead was measured separately.

The results of the classifications for each one of the three datasets are shown in Tables 5.19–5.21. The data reduction overhead ranged from 5 to 60 sec., depending on the dataset. It is generally higher for large as well as complex datasets (e.g. the *magic* dataset).

Classifier Reduction, Ev. Measure		kNN	C4.5	Fuzzy kNN	LDA	GMC
None	Accuracy Rate	88.19	65.40	86.02	50.00	86.73
	CPU Time (sec)	0.59	5.10	0.71	<.01	1.64
1/10	Accuracy Rate	88.21	64.55	86.40	50.00	86.70
	CPU Time (sec)	0.54	4.40	0.64	<.01	1.48
1/4	Accuracy Rate	88.24	67.77	86.87	50.00	86.13
	CPU Time (sec)	0.47	5.17	0.53	<.01	1.24
1/3	Accuracy Rate	88.13	63.95	87.18	50.00	85.73
	CPU Time (sec)	0.42	2.90	0.48	<.01	1.10
1/2	Accuracy Rate	87.93	63.97	87.41	50.00	83.40
	CPU Time (sec)	0.34	1.96	0.37	<.01	0.84

Table 5.19. Data reduction results for *clouds* dataset

Classifier Reduction, Ev. Measure		kNN	C4.5	Fuzzy kNN	LDA	GMC
None	Accuracy Rate	97.60	79.47	88.02	82.19	81.13
	CPU Time (sec)	23.88	394.31	24.76	0.09	57.98
1/10	Accuracy Rate	97.60	34.19	97.74	82.59	85.42
	CPU Time (sec)	20.62	287.68	21.09	0.04	51.19
1/4	Accuracy Rate	97.66	46.66	97.66	82.08	87.94
	CPU Time (sec)	16.43	206.69	16.49	0.03	40.98
1/3	Accuracy Rate	97.68	39.88	97.63	81.79	86.39
	CPU Time (sec)	13.53	169.09	13.43	0.03	31.69
1/2	Accuracy Rate	97.40	37.34	97.37	81.36	73.44
	CPU Time (sec)	7.47	102.07	7.99	0.03	22.52

Table 5.20. Data reduction results for *pendigits* dataset

Classifier Reduction, Ev. Measure		**kNN**	**C4.5**	**Fuzzy kNN**	**LDA**	**GMC**
None	Accuracy Rate	83.04	81.17	82.91	77.35	67.79
	CPU Time (sec)	56.72	127.90	59.95	0.04	151.39
1/10	Accuracy Rate	83.29	81.05	83.09	78.22	67.79
	CPU Time (sec)	48.88	108.02	52.56	0.04	130.26
1/4	Accuracy Rate	82.93	79.41	82.67	78.48	68.15
	CPU Time (sec)	35.43	84.00	38.81	0.03	92.00
1/3	Accuracy Rate	82.64	79.17	82.36	78.62	68.69
	CPU Time (sec)	28.58	70.41	30.77	0.02	83.06
1/2	Accuracy Rate	81.57	77.69	81.52	78.08	69.76
	CPU Time (sec)	19.19	47.20	21.45	0.01	58.39

Table 5.21. Data reduction results for m*agic* dataset

From Tables 5.19-5.21, it can be seen that in the majority of cases, both the Accuracy Rate and the CPU time are enhanced, for all of the classifiers. Also, in some cases (e.g. for the GMC classifier in the dataset Magic), the CPU time is decreased so much that even together with the data reduction overhead (which in this case is about 1 minute) it is still faster than with the original dataset for all of the reduction thresholds.

The reduced dataset is often very similar in appearance (i.e. it maintains the general structure of the patterns in the feature space), as it can be seen in Figure 5.4, for the dataset Clouds. This is because the patterns to be excluded are chosen in such a way that they are far apart from each other. Otherwise, if the Discernibility of each pattern were to be used as the sole decision rule (instead of the product of Discernibility and distance), the reduced dataset would not be as robust.

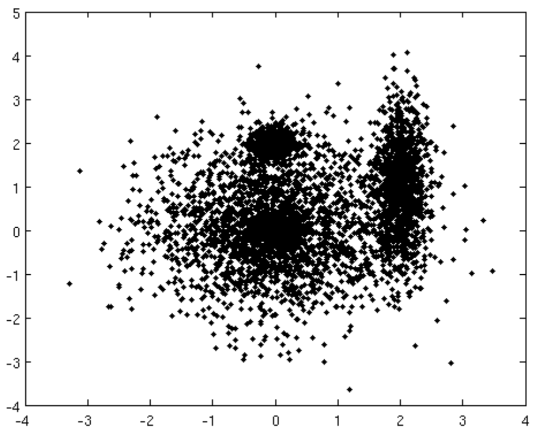
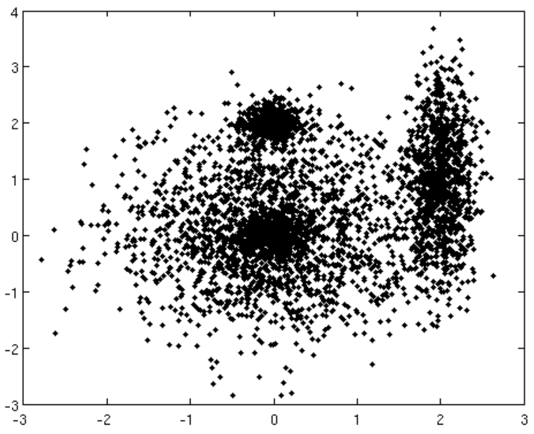
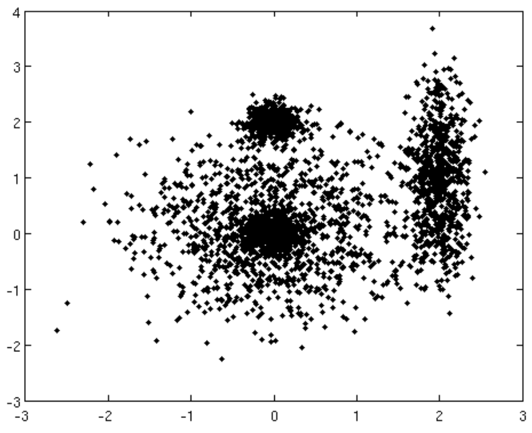
Fig. 5.4. *Clouds* dataset originally (top), with 1/10 reduction (middle) and with 1/3 reduction (bottom). The geometry of the dataset is generally preserved.

5.4 Summary and Contribution of the Chapter

In this chapter two of the applications of Discernibility were presented and discussed. Without any significant modifications, the Index of Discernibility can be applied as part of a feature selection technique. This can be done by (at least) two different ways: either by evaluating each individual feature of a dataset and selecting the best ones

(IFF method) or by taking different groups of features (feature sub-sets of the dataset) and comparing their discernibilities with that of the original feature set (GFS). The results, as the relevant experiments demonstrated, is a significant reduction in the number of features of a dataset and an improvement in the classification performance, both in terms of accuracy rate and in CPU time.

The Discernibility concept can also be applied in data reduction. By taking into account the discernibilities of the various patterns as well as the distances among them, we can remove the ones which are discernible and distant (i.e. their average distance is large), thus yielding a smaller dataset which however maintains the structure of the original one. This is mirrored by the performance of a number of different classifiers, which (for most of them) appears to be as good as or even slightly better than that using the original dataset.

This chapter demonstrated how versatile the Discernibility concept can be, and how its application in the pre-processing stage can enhance the classification performance. Moreover, it contributed innovative approaches to feature selection and data reduction that employ Discernibility which can be potentially useful for other Pattern Classification problems.

Chapter 6 – Reject Option Based on Discernibility

In this chapter we explore how the Discernibility concept can be applicable as a metric for reliability, leading to the possibility of a reject option for a number of different classifiers.

As the classification process is often a costly procedure when it comes to inaccurate classifications, in many cases it is preferable to avoid classifying certain instances, or in other words rejecting the classifications. In addition, there are cases where an unreliable classification can also be costly, due to the need in testing the predictions considered unreliable, by other more expensive ways, e.g. manual classification (Fumera et al., 2000), or developing a new feature set (Giusti et al., 2000). In these cases, the option of rejection to classify particular patterns is applied (Arlandis et al., 2002; Baram, 1998; Cordella et al., 1995; Duda et al., 2001; Fumera et al., 2000; Sansone et al., 2001; Santos-Pereira and Pires, 2005; Thien, 1996a).

The mainstream approach for analysing the trade-off between a potential error and rejection is modelled in terms of posterior probabilities of different classes; if the posterior probability of the best class label is high, then the label is attached to the pattern, otherwise, the classification is not performed (reject-option). The optimal value of the rejection threshold is yielded by what is known as Chow's rule (Baram, 1998; Fumera et al, 2000; Sansone et al., 2001; Santos-Pereira and Pires, 2005; Thien, 1996a, 1996b). This rule is defined as: if the maximum a posteriori probability P_{max} of a classification being correct is less than a predefined threshold T, then the classification is rejected. Also, these probabilities are directly linked with the error rate and the misclassification costs involved. As a result, there is a trade-off between error and rejection (Fumera et al, 2000). Since in most applications the posterior probabilities are not known, some other scoring indexes can be used instead of the posterior probabilities. This approach was implemented a number of times by different authors, specifically, for classifiers based on neural networks in (Cordella et al., 1995; Fumera et al., 2000; Sansone et al., 2001; Santos-Pereira and Pires, 2005) and k Nearest Neighbours in (Arlandis et al., 2002; Denoeux, 1995; Fumera et al., 2000; Giusti et al., 2002). Yet the major premise is the same: the reject-option is defined in terms of the class label scoring function, not independently. This holds true even in the cases where the authors recognise limitations of the approach involving just one rejection threshold. Even when ROC curves (for a definition and overview see (Fawcett, 2006)), that are universal regarding classifiers, are used for rejection of certain patterns (Sansone et al., 2001), there are problems in the cases of more than two classes and of small training sets that lead to unreliable ROC.

Employing the Discernibility concept as a way to predict the reliability of classification yields an independent measure of classifier's reliability over an entity, which is applicable to any classifier on any data set. This measure mirrors the "typicality" of an entity as a representative of its class, which is evaluated by the Index of Discernibility of the class from the entity's location. To fully investigate this possibility we test three different scoring functions for the Discernibility measurement. One of them utilises the proportion of the entity's class within the entity's neighbourhood (Spherical Index of Discernibility). The other two are based on comparison of the average distances from the entity to entities of its class versus distances to entities of other classes. These two measures are the Silhouette Width coefficient (Kaufman and Rousseeuw, 1990) and the Harmonic Index of Discernibility (Chapter 3). Each one of the Discernibility measures defines a reliability elite, the set of patterns with the highest discernibility scores. We experimentally test the different discernibility scoring functions over their elites. Then we use the elites for combining classifiers. Combining classifiers to make a better prediction is a common idea (Ledward, nd; Tsymbal and Puuronen, 2000). This is usually done by using weighted voting or summing schemes (Ledward, 2008). Yet, this subject will be more thoroughly discussed in Chapter 7. In this case, one can combine classifiers according to their discernibility scores in such a way that, at each entity, only classifiers that have highest discernibility score at it are used.

6.1 Experimental Setup

The classifiers used in this series of experiments are kNN, LDA, C4.5, RCE, GMC and MSTC. These are thoroughly described in Section 5.3.1.

Regarding the datasets used, we used five datasets from the UCI repository (UCI Repository, 2008): *Iris*, *Wine*, *Heart*, *E.coli* and *Glass*. A detailed description of these datasets can be found in Appendix A, while their main characteristics are exhibited in Table 6.1 below. These datasets have been chosen because of their:

1. diversity regarding the numbers of attributes and classes,

2. diversity with respect to the proportions of classes, both balanced and unbalanced, and
3. relatively small sizes.

The last point is quite important since the conventional probability-based methodologies for estimating the classification reliability are more problematic to implement on small datasets. The method proposed here aims to fill in this gap.

Dataset	**Number of patterns**	**Number of attributes**	**Number of classes**
Iris	150	4	3
Wine	178	13	3
Heart (disease)	270	13	2
E.coli	336	8	8
Glass (identification)	214	10	6

Table 6.1. Characteristics of the datasets used in our experiments.

All of the experiments were conducted as 50 rounds of the 10-fold cross-validation testing. Therefore there were 500 classification testing exercises for each one of the classifiers used, rendering the results rather stable and, thus, reliable. In each of the experiments all six classifiers were assessed individually as well as collectively, by the amalgamation of their elites, as described previously.

6.2 Reject Option

6.2.1. Reliable Elite for a Classifier

As explained earlier, our concern is in computational prediction of classes in such situations where the cost of errors is overwhelming compared to that of manual prediction (as is typical for complex technical devices). This makes us value the reliability of the computational predictions over the extent of reject-option. Therefore, we focus on the most reliable instances, based on one of the discernibility metrics.

Let us first specify a proportion value α between 0 and 1. Given a classifier, we refer to a set of patterns as its reliability α-elite if they constitute the first αN entities (N is the number of patterns in the training set) in the list of test patterns sorted in the descendent order of the degree of reliability as measured by any of the three Discernibility indexes above. For example, at α=1/2, the reliability elite will be comprised of approximately half of the test data set.

Given a set of classifiers, their reliability elites can be combined in a variety of ways. According to the literature, classifiers have been combined by making use of weighted voting or scoring systems (for a review, see Ledward, 2008). Sansone et al. (2001) used a set of five different classifiers which were used in parallel, and Giusti et al. (2002) made use of two classifiers working together in a serial fashion.

However, in this case we combine classifiers based on their reliability scores. Specifically, for the patterns belonging to more than one of the elites, the classifier yielding a classification with the highest level of discernibility should prevail over the outputs of the other classifiers. This would lead to a simple rule for combining classifiers by combining their elites in the set-theoretic union E and using the most reliable classifier at each one of the patterns in E. However, the overall accuracy rate on the union E can be compromised by combining "unworthy" cases for different classifiers so that the average accuracy rate on E can be lower than the accuracy rates of the individual classifiers over their elites. To address this problem, we introduce two filters in the form of rules for combining the elites of the classifiers. These are the one-third-out rule and the loners-out rule. These rules aim to filter out those elite patterns that might be riskier than the others, leaving an elite with more or less worthy classifications.

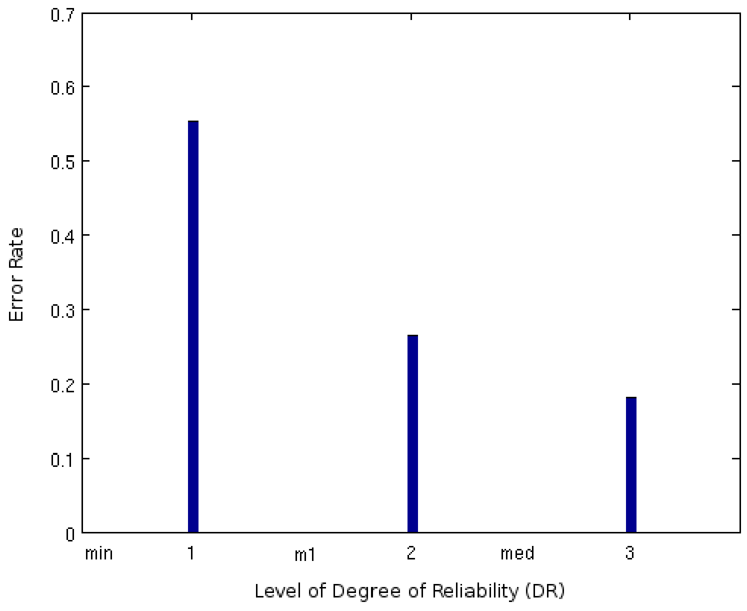

Figure 6.1. The average levels of misclassification for three SID categories for the *Glass* dataset.

6.2.2 The One-Third-Out Rule

This rule discards one third of the elements of the set-theoretic union E of the individual elites. This threshold was selected empirically and is by no means optimum. The objective here is to demonstrate how such a rule works in this paradigm.

Each pattern in E is assigned with a reliability score, which is the maximum of the discernibility scores of the classifiers, whose elite the pattern belongs to. Afterwards, all of the elements of E are sorted according to their reliability scores, after which the bottom third of them is jettisoned. The choice of this threshold is made after careful examination of the levels of misses for the members of E having their discernibility scores within different ranges. In Figure 6.1, one can see that indeed most of the misclassifications occur for the discernibility scores between the minimum and m1, which is the trisection point of the set. The graph is based on the figures obtained for the *Glass* dataset with the Spherical Index of Discernibility, yet they are typical for other sets of data considered. The edge of the One-Third-Out rule is that it guarantees high accuracy rates for the amalgamation of classifiers for a variety of datasets. However, it drastically reduces the size of E.

6.2.3 The Loners-Out Rule

For this filtering technique of the amalgamation elite set E, we concentrate on the number of classifiers supporting patterns in E. From the experiments carried out, we found that in most cases a misclassification occurs on such a member of E which is voted for by only one classifier, i.e., this pattern belongs to the elite of only one individual classifier, as can be seen on Fig. 6.2. This graph is based on the classification results of the G*lass* dataset, but it is typical for other datasets as well.

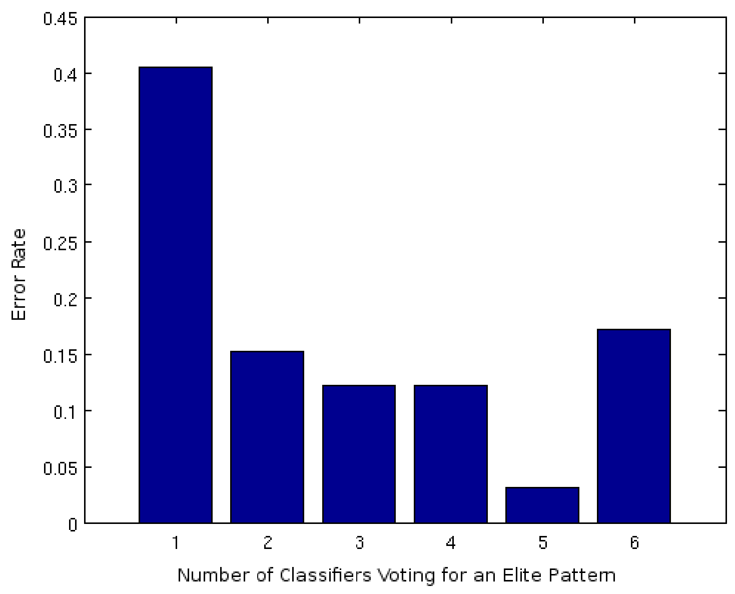

Figure 6.2. Average levels of misclassification for different numbers of classifiers voting for a pattern on the G*lass* dataset.

Therefore, by eliminating those members of E that have been introduced by a sole classifier, we may increase the accuracy rate at the amalgamation. Since, on average, the proportion of such patterns is rather small, the size of E is not that greatly reduced by applying this rule (contrary to the One-Third-Out rule). At the same time, since most of the patterns eliminated are bound to be misclassifications anyway, the overall accuracy rate remains relatively high.

Since the two rules for removing “risky” elite members are independent from each other, they can be used in combination with each other. Apparently, if we apply the Loners-Out rule after the One-Third-Out rule, this would produce larger amalgamated elites, so this is why we always keep this order for the combined rule.

6.3 Evaluation Criteria

The classifiers were assessed using the average Accuracy Rate, a measure which has been used extensively in the literature (Gao & Wang, 2007; Wu et al., 2002), the CPU time, and a correlation between Accuracy and Degree of Certainty, which we call Net Reliability (see Chapter 3).

The CPU time is an .evaluation criterion used previously and is described in Section 4.2.3.2 We also made use of Net Reliability (NR), which is thoroughly described in Chapter 3. This was made possible by calculating the Degree of Certainty (see Chapter 3) in each classification as well.

6.4 Experimental Results

In Table 6.2, the average accuracy rates of the individual classifiers on the whole datasets are exhibited.

Dataset	KNN	LDA	C4.5	RCE	GMC	MSTC
Iris	0.9569	0.8381	0.9456	0.8987	0.9551	0.9541
Wine	0.9514	0.9858	0.9334	0.9095	0.9777	0.9537
Heart	0.8027	0.8389	0.6404	0.6918	0.8005	0.7644
E.coli	0.8655	0.8557	0.8008	0.7691	0.7919	0.7952
Glass	0.6676	0.6049	0.6281	0.6254	0.6800	0.6928

Table 6.2. Accuracy rates of the classifiers under consideration.

One can observe that, with respect to the classifiers, the datasets fall into the following three categories:

1. relatively high accuracy rate of 90-97% on Iris and Wine;
2. medium accuracy rate of about 80% on Heart and E.coli, and
3. relatively low accuracy rate of 60-70% on Glass.

We can also observe that performances of different classifiers peak at different datasets: kNN is the best on Iris and E.coli, LDA is the best on Wine and Heart (and the worst on Iris), and MSTC is the best on Glass. The other three algorithms trail behind regarding the overall performances, yet they should not be discarded altogether – each may have a good performance as mirrored in Table 6.3.

For our experiments, we maintained two levels of elites: $\alpha=1/2$ and $\alpha=1/3$, the former choosing those patterns whose degree of discernibility is better than the median discernibility, and the latter comprising the best third of discernible patterns. These two are maintained at each of the three discernibility indexes defined above, SID, SW and HID. Table 6.4 presents average accuracy rates of the individual classifiers on the five datasets at each of the six combinations of the elite level and discernibility index.

Classifier	D. Meas, Elite		Iris	Wine	Heart	E.coli	Glass
kNN	SID	50%	0.9997	0.9997	0.9139	0.9658	0.6941
		33%	1.0000	1.0000	0.9476	0.9788	0.7328
	SW	50%	0.9997	0.9965	0.9239	0.9366	0.6898
		33%	1.0000	1.0000	0.9511	0.9788	0.7328
	HID	50%	1.0000	0.9990	0.9141	0.9268	0.8596
		33%	1.0000	1.0000	0.9478	0.9305	0.9230
LDA	SID	50%	0.9990	1.0000	0.9146	0.9705	0.6929
		33%	1.0000	1.0000	0.9476	0.9790	0.7219
	SW	50%	0.9995	1.0000	0.9241	0.9319	0.6453
		33%	1.0000	1.0000	0.9511	0.9502	0.6970
	HID	50%	1.0000	1.0000	0.9174	0.9267	0.8124
		33%	1.0000	1.0000	0.9482	0.9302	0.8887
C4.5	SID	50%	0.9997	0.9997	0.9175	0.9605	0.6913
		33%	1.0000	1.0000	0.9416	0.9779	0.7289
	SW	50%	0.9997	0.9972	0.9186	0.9545	0.6413
		33%	1.0000	1.0000	0.9509	0.9579	0.7041
	HID	50%	1.0000	0.9990	0.9062	0.9470	0.8284
		33%	1.0000	1.0000	0.9524	0.9449	0.9157
RCE	SID	50%	0.9997	1.0000	0.9071	0.9581	0.6401
		33%	1.0000	1.0000	0.9453	0.9762	0.7002
	SW	50%	0.9995	0.9967	0.9199	0.9548	0.6669
		33%	1.0000	1.0000	0.9491	0.9745	0.7259
	HID	50%	1.0000	0.9988	0.9116	0.9492	0.8388
		33%	1.0000	1.0000	0.9476	0.9631	0.9300
GMC	SID	50%	0.9997	0.9997	0.9123	0.9621	0.7055
		33%	1.0000	1.0000	0.9476	0.9776	0.7239
	SW	50%	0.9997	0.9967	0.9237	0.9382	0.7080
		33%	1.0000	1.0000	0.9511	0.9501	0.7554
	HID	50%	1.0000	0.9990	0.9140	0.9266	0.8740
		33%	1.0000	1.0000	0.9478	0.9286	0.9408
MSTC	SID	50%	0.9997	0.9997	0.9144	0.9617	0.7440
		33%	1.0000	1.0000	0.9560	0.9777	0.7739
	SW	50%	0.9997	0.9967	0.9224	0.9310	0.7026
		33%	1.0000	1.0000	0.9533	0.9449	0.7540
	HID	50%	1.0000	0.9990	0.9156	0.9254	0.8463
		33%	1.0000	1.0000	0.9520	0.9288	0.9074

Table 6.3. Accuracy rates of various classifiers at different elite levels and Indices of Discernibility.

The results from Table 6.3 show:

1. All three discernibility indexes lead to drastically raising accuracy rates for all the classifiers, reaching 100% accuracy for Iris and Wine datasets and about 95-97% accuracy on E.coli dataset on the 50%-elites. The only diehard is Glass dataset

that does not change much the accuracies at the 50%-elites over SID and SW indexes. Still, HID index leads to a much improved, 85%, accuracy over 50%-reliability elites, and more than 90% accuracy over the 33%-reliability elites.

2. There is no overwhelming winner among the three discernibility indexes, though each of the indexes shows consistent results over all the classifiers. Specifically, SID always wins on Wine and E.coli datasets, HID always wins on Iris and Glass datasets, and SW is the winner on Heart dataset.

The RCE and GMC classifiers, which exhibit a mediocre performance over the total data set, appear to become most effective over the elite.

The results for the amalgamation of reliability elites using rules One-Third-Out, Loners-Out and their combination are shown in Tables 6.4 and 6.5. The former relates to the accuracy rates whereas the latter to the amalgamation sizes.

Rule	**DI Elite**		**Iris**	**Wine**	**Heart**	**E.coli**	**Glass**
Loners Out	**SID**	50%	0.9991	0.9993	0.9054	0.9557	0.6630
		33%	1.0000	1.0000	0.9422	0.9753	0.6907
	SW	50%	0.9991	0.9994	0.9142	0.9332	0.6107
		33%	1.0000	1.0000	0.9476	0.9450	0.6843
	HID	50%	0.9998	0.9991	0.9120	0.9223	0.7714
		33%	1.0000	1.0000	0.9470	0.9317	0.9030
One Third Out	**SID**	50%	1.0000	1.0000	0.9396	0.9763	0.6754
		33%	1.0000	1.0000	0.9710	0.9823	0.7253
	SW	50%	1.0000	0.9997	0.9463	0.9420	0.6277
		33%	1.0000	1.0000	0.9751	0.9630	0.7123
	HID	50%	1.0000	1.0000	0.9407	0.9305	0.8645
		33%	1.0000	1.0000	0.9588	0.9371	0.9164
Combination of Both Rules	**SID**	50%	1.0000	1.0000	0.9425	0.9764	0.6824
		33%	1.0000	1.0000	0.9736	0.9817	0.7484
	SW	50%	1.0000	0.9997	0.9423	0.9478	0.6457
		33%	1.0000	1.0000	0.9760	0.9652	0.7191
	HID	50%	1.0000	1.0000	0.9405	0.9302	0.8677
		33%	1.0000	1.0000	0.9635	0.9417	0.9444

Table 6.4. Accuracy rates at the different methods of amalgamation of reliability elites.

Rule	D. Meas., Elite		Iris	Wine	Heart	E.coli	Glass
Loners Out	**SID**	50%	55.6	50.3	57.0	54.8	68.0
		33%	43.4	39.5	35.6	36.8	46.0
	SW	50%	54.1	50.1	56.3	57.8	70.7
		33%	33.3	37.1	35.0	37.4	47.6
	HID	50%	53.8	50.2	54.7	55.8	66.5
		33%	33.4	37.1	34.1	37.0	39.6
One Third Out	**SID**	50%	43.8	36.0	37.0	35.3	43.2
		33%	20.0	24.7	22.5	23.5	29.3
	SW	50%	34.0	31.7	36.6	37.6	45.3
		33%	20.0	24.7	22.5	24.2	29.9
	HID	50%	33.8	31.7	35.4	36.0	43.0
		33%	21.0	24.7	25.8	27.0	30.1
Combination of Both Rules	**SID**	50%	43.9	35.9	36.9	35.3	43.3
		33%	19.9	24.7	22.6	23.6	28.5
	SW	50%	34.1	31.7	36.5	37.7	45.6
		33%	19.9	24.7	22.5	24.2	30.0
	HID	50%	33.9	31.6	35.5	36.0	42.5
		33%	20.0	24.7	22.3	24.1	24.4

Table 6.5. Elite sizes, per cent, at different methods of amalgamation of reliability elites.

These results lead to the following conclusions. As one would expect, the One-Third-Out rule consistently outperforms the Loners-Out rule; however, this is by just a small margin of the order of 1% or less - with the price of drastically reducing the size of the elite. Overall, the amalgamation does not boost performances of the algorithms that much. However, we can see that HID 33%-elites consistently lead to the accuracy rates of 90% and more. The combined rule raises the accuracy on the Glass dataset – the most difficult for predictions – to more than 94%. The price, in terms of reject-option applied to the non-elite, is rather high indeed: 60%, 70%, and 75% of all cases for Loners-Out, One-Third-Out and Combined rules, respectively. But this may be worth doing in the situations at which the reliability of classification is a must.

Also, Table 6.5 shows that the SID 33%-elite Loners-Out amalgamation leads to somewhat better coverage of the data – about 45%, rather than 33%, of the dataset are there.

6.5 Fine-tuning of the Method

We have observed that the Degree of Reliability can be enhanced by adding two other factors in its calculation. These are the change in Accuracy Rate and the change in Degree of Certainty, within the training set. Therefore, by combining these three factors in a new reliability metric, we can obtain even better results. However, the level of contribution of each one of them is different, so this needs to be reflected in the new metric as well. This is accomplished by introducing three parameters in it, one for each factor. As a result, the new reliability metric works more efficiently.

6.5.1 Parametric Degree of Reliability

The Parametric Degree of Reliability (PDR), which is a function of the three factors mentioned previously, attempts to evaluate how much the unknown pattern X (which is classified into a class Y) "fits" into the training set P. This is done by examining how the Accuracy Rate of the classifier applied on the training set is improved and how the Degree of Certainty of this classification improves as well. The improvement of these measures is measured by taking the ratio of their values after the adoption of X in P over their values before.

The generic formula for the Parametric Degree of Reliability is:

$$PDR = D_X^a \cdot ARR^b \cdot DCR^c \tag{6.1}$$

where D_X is the (expected) Discernibility of the unknown pattern, calculated like the normal Discernibility (as seen in Chapter 3), using the class prediction of the classifier as the class label, *ARR* is the Accuracy Rate Ratio, *DCR* the Degree of Certainty Ratio, and *a*, *b* and *c* the weight parameters which take values between 0 and 1. These parameters, as they are exponents (Eq. 6.1), affect the outcome immensely. Therefore, they need to be chosen carefully, in order to obtain a useful result. In other words, we need to choose values *a*, *b* and *c* using an optimisation technique so that the accuracy rate of the elite of classification, based on this measure, is maximised. The technique used is the Vibrating Grid Optimser, which was developed specifically for this purpose and is described below.

6.5.2 The Vibrating Grid Optimisation Technique

As the parameters needed to be optimised fall in the 3-dimensional space and are restricted within the range of [0, 1] for each dimension, we can depict them as points belonging to a cube defined by the points (0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1). By taking them as a starting point for a grid, we can gradually optimise the accuracy rate of the elite classifications, based on the PDR function (Eq. 6.1). The elite patterns are defined as the ones having the highest reliability value. To select the best classification on the elite patterns, we evaluate each point and then take at random another point in the neighbourhood. The latter is defined as the range of half the distance to the nearest point of the grid. So, originally the neighbourhood is the range [$x \pm 0.5$, $y \pm 0.5$, $z \pm 0.5$]. To speed up the process, if the boarders of the neighbourhood fall below 0 or over 1, we replace them to 0 and 1 respectively. If the new point yields a higher score, then it replaces the grid point related to it (i.e. the grid point around which it was generated). After a new set of points is developed it replaces the older grid and the neighbourhoods are calculated again.

This process is repeated until the accuracy rate of the elite classifications in terms of PDR is reached. Usually the number of classifications is fixed as a given proportion of the testing set. As the process does not involve any complex calculations, it is exceptionally fast and always converges in a relatively small number of steps.

Although this method was developed for optimising a set of three parameters, it can be used for more sophisticated optimisation problems having more dimensions. The only restriction is that the parameters have to be bound in given intervals.

6.5.3 Experimental Results for the Parametric Degree of Reliability

The experiments were carried out on 10 rounds of 10-fold cross validation. Just like the previous set of experiments, five different datasets were used, taken from the UCI repository (UCI Repository, 2008): iris, wine, heart disease, e.coli, and glass identification. The characteristics of the above datasets are exhibited in Table 6.1.

We applied our method using the same six classifiers as previously: the k Nearest Neighbours (kNN), the Linear Discriminant Analysis (LDA) method, the

Minimum Spanning Tree Classifier (MSTC), introduced by Voulgaris & Mirkin (2008), the Gravity Model Classifier (GMC), based on one of the classifiers developed by Ruta and Gabrys (2003), the C4.5 Decision Tree, and the Reduced Coulomb Energy classifier (RCE), as described by Duda et al. (2001). The elite classifications were taken to be the ones having the top one-third values in terms of parametric degree of reliability. In other experiments we have taken the elite to be the top half of the DR scores, but this compromises the accuracy rate of the selection. However, one can experiment with different thresholds around the one we used here, without significantly changing the results.

We attempted optimising the parameters of PDR for all datasets and classifiers together and then for each pair of them. The second strategy proved to be better overall. Its results can be seen in Table 6.6.

Classifier	Iris	Wine	Heart	E.coli	Glass
KNN	1.0000	1.0000	0.9566	0.9287	0.9284
LDA	1.0000	1.0000	0.9544	0.9276	0.9017
MSTC	1.0000	1.0000	0.9566	0.9240	0.9171
GMC	1.0000	1.0000	0.9533	0.9222	0.9466
C4.5	1.0000	1.0000	0.9499	0.9403	0.9312
RCE	1.0000	1.0000	0.9577	0.9638	0.9382
Average	1.0000	1.0000	0.9548	0.9344	0.9272

Table 6.6. Accuracy Rates of elite classifications based on the Parametric Degree of Reliability.

The overall accuracy rate of all the classifiers over all five datasets was 0.9633 while the first strategy yielded an average accuracy rate of 0.9211. Therefore, by employing this method and fine-tuning it to each individual case (classifier-dataset pair), the error rate on average is halved.

Also, if we were to compare it with the original DR measure (which is based on Discernibility alone), PDR outperforms it by showing a 6% lower error rate. Yet, in one of the datasets, Glass, the decrease in the error rate using the Parametric Degree of Reliability was about 25%. This is significant, if one considers that this is the most “difficult” dataset among those in this group.

Note that on average, the one or two of the parameters found for the reliability measure sometimes approached 0, implying that a particular factor was not so important for these instances. Yet, if we were to eliminate it completely, the results would be degraded. This demonstrates that the optimisation function is very sensitive to changes in the parameters and that all of these parameters are essential in order to obtain a good outcome.

6.6 Summary and Contribution of the Chapter

In this chapter we demonstrated how the Discernibility concept could be applied as a measure of classification reliability both for a single classifier and for an amalgamation of classifiers. Also, we showed how it could be elaborated into a parametric reliability measure taking into account other factors, particularly the change of accuracy rate and the change of Degree of Certainty. The quite promising experimental results verified our hypothesis that Discernibility can be a useful reliability metric, especially if used in combination with the other two factors.

The contribution of this chapter in the Pattern Recognition field is that it offers another measure for classification reliability, a measure that does not depend on probabilities or assumptions about the distributions of the dataset. The flexibility that this characteristic offers allows the proposed reliability measure to be more applicable to classification problems, particularly when the dataset is relatively small.

This chapter also contributes significantly to the thesis, as it describes how with practically no changes in its function, Discernibility can be applied to an entirely different domain, yielding useful information on the unknown patterns, *a priori*. Considering that this notion was developed primarily to evaluate *known* patterns, it is quite interesting how without any modifications it can be used to assess unknown ones too, therefore enhancing the classification process by offering a reject option to the classification system.

Chapter 7 – Employing Discernibility in Classifier Ensembles

In this chapter we explore how a set of classifiers can be used together as an ensemble, for classification problems, using the Discernibility concept. This is accomplished by encompassing Discernibility as a criterion for the creation of the feature sub-sets of the dataset, used in the ensemble.

7.1. Techniques for Formulating Ensembles

Over the past few years several methods have been introduced for the creation of ensembles of classifiers, most of which involve the introduction of diversity of errors (in classification) as the main component of the ensemble (Brown et al., 2005). Diversity of errors is defined by Eq. 7.1 below:

$\text{Div(X,Y)} = \dfrac{N_{11}N_{00} - N_{10}N_{01}}{\sqrt{(N_{11}+N_{10})(N_{01}+N_{00})(N_{11}+N_{01})(N_{10}+N_{00})}}$	Eq. 7.1

where X,Y are the diverse classifiers examined

N_{11} and N_{00} are the number of instances where both classifiers are right or wrong respectively, about their predictions on the Validation Set

N_{10} is the number of instances where the first classifier is right and the second classifier wrong, at the same time

N_{01} is the number of instances where the second classifier is right and the first classifier wrong, at the same time

A great deal of research has been carried out on diversity and on ways of measuring it as described in Kuncheva & Whitaker (2003). One of the most popular approaches is that of reverse correlation, as in the case of Zio et al. (2007). However, other researchers prefer more sophisticated methods of diversity measurement, such as entropy, in the case of Cunningham (2001).

Various ways for attaining diversity in a group of classifiers so as to form an ensemble have been proposed (Brown et al., 2005). The simplest method is to train each ensemble member using randomly initialised parameter setting (e.g. initial weights in the case of neural classifiers). A more advanced approach is to train the different classifiers on different subsets of the training set as bagging (Breiman, 1996) or variations of it (Evgeniou et al., 2004), where each training set is created by resampling and replacement of the original training set with uniform probability. Boosting (Freund & Schapire, 1996) also uses resampling of the training set, but the data points previously poorly classified receive a higher probability.

Among the most efficient techniques, particularly for datasets having a large number of features, is that of partitioning the dataset into subsets of features, each of which is used independently as a training set for a classifier, called feature resampling. This approach is sometimes problematic since if the subsets are created at random, the huge number of possible combinations may render the whole process impractical (Cunningham, 2001). Yet, in many cases a random selection of features yields good results, as in the case of Bertoni et al. (2005).

Another approach to feature resampling is with the use of global search methods, such as Genetic Algorithms to improve the quality of the partitioning (Zio et al., 2007), which is particularly useful when dealing with multiple criteria to be optimised.

In all the cases investigated, it becomes evident that diversity plays a significant role in the creation of an ensemble and guarantees a good classification performance. However, to the best of our knowledge, none of the approaches consider the factor of classification quality of the created feature partitions (i.e. subsets of features), an issue that is addressed in our approach.

7.2 The Role of Discernibility in Ensembles

7.2.1 Structure of Proposed Ensemble

Although a high overall Discernibility in the feature subsets does not guarantee diversity (and vice versa), we hypothesise that by using it in the partitioning of the feature set it will yield good quality partitions, which in turn will yield good generalisation for the various classifiers of the ensemble. Note that this method operates at the level of feature resampling, so in the form presented here it cannot be considered as substitute for methods like bagging (Breiman, 1994; Breiman, 1996) and boosting (Freund & Schapire, 1996; Schapire, 2003).

As mentioned earlier, the main objective of feature resampling in ensemble creation is the introduction of diversity in the classifier errors by training them in different subsets of the dataset's feature space. Therefore there is one function that is maximised, that of the diversity, which is calculated using Eq. 7.1. However, some of the subsets may be of potentially low classification quality rendering the classification relatively inaccurate, so even if the diversity of the errors is high, the overall performance may not be optimum. This is resolved by introducing another function, that of the average discernibility. By taking into account the product of the diversity and the discernibility (both of which take values between 0 and 1), we have another function to maximise f = diversity × discernibility. This optimisation takes into account both factors and yields a partition that is both diverse in errors and of overall good quality. Yet, this method for selecting the best partition is suboptimal since it is done using a number of randomly generated partitions. Nevertheless, if this number is large enough, it is quite likely that there will be a near-optimal solution for the feature partitioning, as the candidate partitions will probably include some partitions like the ones we are looking for.

Also, as the partitions of the feature set depend on the dataset and the classifier, for each ensemble there is a variable number of classifiers of each type. In addition, for every different classification, the ensemble adapts to the data and changes its structure so as to be more efficient.

7.3 Experiments and Results

The ensemble employed in the experiments is structured as follows. Eight different classifiers are used and each one of them is trained in a number of partitions of the training set. These are the V-kNN classifier (see Section 4.3.1.1), the D-kNN classifier (see Section 4.2.1), the Minimum Spanning Tree Classifier (see Section 4.3.2), the Linear Discriminant Analysis (LDA), the Reduced Coulomb Energy method (RCE), the C4.5 Decision Tree, the Gravity Model Classifier, and the Fuzzy kNN.

Another partition called the validation set (which is part of the training set) is used for assessing these classifiers. Based on their results in the classification of its patterns, some classifiers specialising in particular subsets of the feature set of the training set are selected. Then, based on the expected Discernibility of each test pattern, as well as its position, the pattern is classified by the appropriate specialist.

For the experiments conducted, three different datasets were used, taken from the UCI repository (UCI Repository, 2008): Glass identification, Wine, and E.coli. A detailed description of the datasets can be found in Appendix A, while the main characteristics of these datasets in Table 7.1. The experiments were carried out on 50 rounds of 10-fold cross validation.

Dataset	**Patterns**	**Attributes**	**Classes**
Glass	214	10	6
Wine	178	13	3
E.coli	336	8	8

Table 7.1. Characteristics of the datasets used in our experiments.

The results of the experiments on the different ensemble types for the datasets used are shown in Table 7.2.

Ensemble Type	**Glass**	**Wine**	**E.coli**	**average**
Diversity only	0.7412	0.9748	0.7866	0.8342
Discernibility only	0.7188	0.9751	0.8446	0.8461
Diversity & Discernibility	0.7398	0.9754	0.8440	0.8531

Table 7.2. Average accuracy rates and average overall performance for the three ensemble types.

As one would expect, the combination of diversity and Discernibility exhibited on average improved performance in the classification of the ensemble. Also, the discernibility as a criterion for the partitioning of the feature set appears to work on average slightly better than the diversity criterion. The only exception is the Glass dataset where diversity exhibited better performance than the other two approaches, although the combination of diversity and Discernibility was not far behind in performance. This is probably due to the fact that the Discernibility of this dataset is very low, yielding partitions of low Discernibility as a result. Therefore, in such a case, the Discernibility criterion is bound to not work as it should.

The number of base classifiers in the ensemble ranged from 15 to 40. Also, the ensemble performance time (which included training, testing, evaluating and fusing the outputs of the different classifiers) was on average significantly higher in the ensembles involving Discernibility, as this is more computationally expensive. Thus, for ensembles that combined diversity and Discernibility the average time was 1.75 times greater than that of the ensembles using diversity only. In addition, we measured the average performance of each one of the base classifiers and found it to be significantly less in all of the ensembles created.

7.4 Discussion

From the results of the experiments, one can argue that although diversity yields promising results, the classification performance of an ensemble can be improved by incorporating Discernibility in the evaluation process of the feature-set partitioning. Also, even though Discernibility appears to work slightly better than diversity, the proposed method of combining the two criteria performs even better and shows an avenue of research worth of further investigation.

The performance of the combined criteria approach can be explained as follows. Diversity eliminates the feature subsets, which are very similar in terms of errors, for every given classifier. This results in having a partition, which in essence is equivalent to *m* different training sets for each classifier (*m* is the number of partitions and depends on the dataset as well as on the classifier). However, some of these subsets may be useless in terms of their contribution to the ensemble performance (the classifiers trained by them do not yield any noteworthy classifications), therefore "confusing" the ensemble. This is where the discernibility criterion enters the scene. Discernibility makes sure that each subset has relatively high classification quality. Note that in some cases it may be that a subset may have higher discernibility than the original dataset because some of the features may act as noise to the classification process, offering no useful information. By making use of the discernibility criterion we make sure that not all of these features end up in the same subset.

The whole process of the feature-set partitioning can be improved by incorporating an evolutionary method for finding the optimum partition; however this falls beyond the scope of this work, which is to show that discernibility can be a useful asset in the ensemble creation process. Based on the experimental results, this is a valid statement, although it comes at the price of additional computation, which is translated into considerably more CPU time.

7.5 Summary and Contribution of the Chapter

In this chapter a novel ensemble-related approach, based on the Discernibility concept was presented. This approach involves an ensemble of classifiers that are specialised on different feature subsets of the dataset, forming a diverse feature set. The edge of the proposed method is that due to the use of Discernibility, the feature subsets are more or less of the same classification quality, thereby offering an adequate generalisation potential for each one of the specialised classifiers.

The aim of this part of the thesis is not to give a definite solution to the problem of creating diverse ensembles but to show that the Discernibility concept can be used, in an altogether different way, for the creation of balanced feature subsets of the dataset, in order to build robust diverse ensembles. This could also make a promising contribution to the field of Pattern Classification offering an alternative perceptive in an area where there is a lot of ongoing research without a definite solution yet to the problem of creating diverse ensembles.

Chapter 8 – Conclusions

8.1 Conclusions and Contribution

Based on the results obtained in the various sets of experiments, we can conclude that the Discernibility concept is a useful component of all the classification systems where it has been applied. This is possible due to the low computational cost of this method, which is more efficient compared to the other ones which are similar to it, such as the Silhouette Width. Its edge is that it reveals structural information on the dataset both as a whole and at a pattern level, in a quick and easy to adapt manner. This allows it to be easily adjusted and encompassed in a variety of applications, offering them an enhanced insight into the structure of the dataset, something that statistical classification methods attempt to do by assuming particular distributions. The difference in this case, which is also the main advantage of Discernibility, is that it makes no assumptions and examines the structure of the dataset, based on how little its classes overlap, through the individual assessment of each one of its patterns. The introduction of this concept, along with the two indexes that measure is are the main contribution is the introduction of this research project.

Regarding particular applications of this concept, the Discernibility measure does enhance the function of the kNN classifier in the two variations where it was used (D-kNN and W-kNN), with a slight compromise in the classification speed. Yet, the proposed kNN extensions depicted clearly how the Discernibility concept can be a promising way of evaluating particular patterns or particular features, during the classification process, thereby enhancing it, for a number of different datasets. This, along with the three other classifiers introduced in this thesis (V-kNN, CB-kNN and MSTC), is another contribution of this research.

In addition, this concept exhibited a quite promising behaviour in the Discernibility-based feature selection methods introduced in this work. These methods, not only improved the classification speed significantly, but they also improved the accuracy rate and the Net Reliability in most of the cases. Furthermore, the two methods put forward in this research demonstrated how the Discernibility concept can be applied in different ways for this particular application, without any significant change in its function. This is an important contribution of this project, considering the efficiency of these two methods in the complex datasets used for the experiments.

As regards the data reduction techniques, based on Discernibility and distance, they yielded promising results as well, even though the overhead is not insignificant. Yet, the reduced datasets exhibit the same accuracy rate for most of the classifiers and in some cases the accuracy rate is increased. One of the advantages of this approach is that even though Discernibility has an altogether different philosophy in the evaluation of the significance of the patterns of a dataset, it can easily be used in combination with other criteria, such as distance in this case, yielding quite promising results. This technique is another useful contribution, which opens us a new perspective to the Data Reduction area.

The Reject Option based on the reliability measure (which is primarily centred on the Discernibility measure) yielded promising results as well. Not only did it provide a quite high accuracy rate for elites of patterns, in a number of datasets, but it also yielded a justification why certain classifications were rejected (low reliability). In addition, this was done without increasing the computational cost of the classification process much. In addition, by encompassing the change of Accuracy Rate and the change of Degree of Certainty, this method was enhanced. Also, through its fine-tuning, using a grid-based optimisation algorithm, this method yielded even better results, as it became more adaptable to the specific problem it was applied on, as well as to the particular classifier used. This method, along with its supplementary techniques, is one more contribution of this thesis.

Regarding the ensembles, the quite promising results of the experiments verified the initial hypothesis that Discernibility can be a useful asset in the creation of balanced subsets of the feature space. Moreover, it became evident that even if Discernibility was the only criterion for the creation of balanced subsets, the results are still better than those using the criterion of diversity of errors. Yet, by using both criteria, the accuracy rate of the ensemble can be enhanced. This can be explained as follows. By applying the diversity of errors criterion one makes sure that the different feature subsets of the dataset yield a different generalisation for a given classifier. As regards the Discernibility criterion, this ensures that the subsets will be of the same classification quality, thereby yielding a better generalisation since the differently trained classifiers will be characterised by a better classification quality as well. Therefore, by combining both criteria, the feature subsets will not only be supplementary to each other, but also of more or less the same standard. This alternative approach to diverse ensembles is another contribution of this research.

Moreover, the auxiliary measures introduced in this project – the Degree of Certainty (which is a generalisation of the certainty factor, applicable to most classifier types) and the Net Reliability (a measure which evaluates how reliable the certainty of a classifier is, based on its accuracy – can be seen as one more contribution of this thesis.

8.2 Limitations

Promising as they may be, the measures expressing the Discernibility concept have some limitations that need to be addressed. First of all, both the Spherical and the Harmonic Indices of Discernibility may be time-consuming when it comes to very large datasets. Therefore, an option has to be introduced that allows the user to use these indexes to evaluate a dataset (or a feature of it) based on a sample of its patterns. This would speed up the whole process, yet it would come at the price of not yielding the discernibilities of the individual patterns. Therefore, this version of the Index of Discernibility would be applicable for large-scale evaluation, in the relevant applications (e.g. feature selection, partitioning of feature space for diverse ensembles, evaluation of a dataset, etc.).

Another limitation that we have discovered is that it is not a continuous metric. If therefore there would be a way to evaluate any point of the dataset feature space, even where there are no patterns, this would render it more robust as it would offer an insight to the dataset as a continuum and not as merely a group of patterns.

Finally, the Discernibility Indexes assume equal importance to all of the classes of the dataset, something that in some cases may not be valid. This is more noticeable in cases where the misclassification costs of each class are very different, such as in the case of medical datasets (the class denoting the presence of an illness is more important as misclassifying it may lead to delayed treatment of the patient).

These limitations show that the Discernibility concept, though versatile and practical in theory, has plenty of room for improvement as regards its implementations. This however renders it a fruitful topic for future research, promising even better results.

8.3 Future Work

It is our expectation that the Discernibility concept will be able to find other applications as well, which are however beyond the scope of this work. These can be categorised as short-term and long-term, depending on the research workload involved.

Among the short-term ones is the use of Discernibility as the basis for feature generation. Such a pre-processing application will be aiming at enhancing the quality of a dataset by generating and evaluating new features, resulting to more accurate and/or faster classification (since the same or even better classification potential could be reached with less and more robust features).

Another short-term application involves the proposed ensembles in Chapter 7. These can be further enhanced and new ensembles or hybrid classification systems can be developed based on the Discernibility concept in combination with the diversity of errors. In addition, the possibility of other ensemble setups can be investigated.

One other short-term application has to do with the fine-tuning and addressing the limitations of the proposed methods. Particularly, the Indexes of Discernibility can be modified so as to take into account the misclassification cost, something which may prove useful in certain problem domains. This can be done by employing and extending the Degree of Reliability measure, adjusting it to the various datasets, based on the misclassification costs.

Discernibility can also be applied in the short-term application of dealing with discrete data more efficiently. This could be done by changing the distance measure used and the way the data is mapped.

Regarding the long-term applications, the Discernibility concept can be used in other problem domains, such as clustering. By employing one of the Indexes of Discernibility, the clustering process can be enhanced resulting to more meaningful and more clear-cut clusters.

Another long-term application is the use of Discernibility to explore which classifiers can be more suitable for which particular types of datasets based on its evaluation capability. This however is bound to be a complex investigation as it would require experiments with a number of different classifiers to prove the hypothesis.

One more long-term application is the use of Discernibility as a part of the function of Fuzzy classifiers. This could be possible by incorporating the Discernibility scores of the various features in the corresponding membership functions, in an automated or semi-automated way. This however would require a lot of experiments in order to determine the optimum parameters involved.

The spawning of artificial patterns in a sparse dataset could be another long-term application of Discernibility. Using Discernibility, it could be possible to filter the most appropriate patterns generated, so that the new dataset maintains the same structure and classification quality but with more patterns. This translates into a better accuracy rate, although there are many factors that need to be taken into account in order to preserve the integrity of the dataset.

Discernibility can also be applied in the long-term application of time-series prediction. This could be possible by introducing new features so that each dynamic variable is replaced by static ones. Then Discernibility could be used to assess the classification quality of these variables and this way provide the optimum "time window" to be used.

Overall, as shown from this research, the Discernibility concept is a quite promising notion and its implementations are quite versatile and practical in many ways. Its great potential is not thwarted by its limitations, which may be seen as good opportunities for future avenues of research that may gradually facilitate and enhance the process of classification.

References

Aarts, E. and Korst, J., 1990. *Simulated Annealing & Boltzmann Machines*. John Wiley & Sons.

Abidin, T. and Perrizo, W., 2006. SMART-TV: a fast and scalable nearest neighbor based classifier for data mining. *Proceedings of ACM SAC-06*, Dijon, France. ACM Press, New York, NY, 536-540.

Aeberhard, S., Coomans, D. and de Vel, O., 1992. Comparison of Classifiers in High Dimensional Settings, Tech. Rep. 92-02, Dept. of Computer Science and Dept. of Mathematics and Statistics, James Cook University of North Queensland.

Ahmadi, A., Omatu, A., Fujinaka, T., and Kosaka, T., 2004. Improvement of reliability in banknote classification using reject option and local PCA. *Information Sciences*, vol. 168 (1-4), 277-293.

Alexe, G., Bhanot, G., and Venkataraghavan, B., 2005. A robust meta-classification strategy for cancer diagnosis from gene expression data. *IEEE Computational System Bioinformatics Conference*, 322-325.

Alimoglu, F., 1996. *Combining Multiple Classifiers for Pen-Based Handwritten Digit Recognition*. MSc Thesis, Institute of Graduate Studies in Science and Engineering, Bogazici University.

Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, D. and Levine, A. J., 1999. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays, *Natl. Acad. Sci. USA*, vol. 96, 6745-6750.

Arlandis, J., Perez-Cortes, J.C., and Cano, J., 2002. Rejection strategies and confidence measures for a k-NN classifier in an OCR task. *Proceedings of 16th International Conference on Pattern Recognition ICPR-2002*, vol. 1, Québeq (Canada), 576-579.

Aydin, T. and Guvenir, H. A., 2006. Modeling interestingness of streaming classification rules as a classification problem. *14th Turkish Symposium on Artificial Intelligence and Artificial Neural Networks*, 168-176.

Ayrulu, B. and Barshan, B., 2002. Reliability measure assignment to sonar for robust target differentiation. *Pattern Recognition*, vol. 35 (6), 403-419.

Baase, S. and Van Gelder, A., 1999. *Computer Algorithms: Introduction to Design and Analysis*, 3rd edition. Addison-Wesley.

Babuska R., 1998. *Fuzzy Modeling for Control.* Kluwer Academic Publishers, Boston, USA.

Bailey, D.G., 2004. An efficient Euclidean distance transform. *Proceedings of IWCIA04*, 394-408.

Baram, Y., 1998. Partial Classification: the benefit of deferred decision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20 (8), 769-776.

Bermejo, S. and Cabestany, J., 2000. Adaptive soft k-nearest neighbour classifiers. *Pattern Recognition,* vol. 33, 1999-2005.

Bertoni, A., Folgieri, R., and Valentini, G., 2005. Bio-molecular cancer prediction with random subspace ensembles of support vector machines. *Neurocomputing,* vol. 63, 535-539.

Bezdek, J. C., Tsao, E. C.-K. and Pal, N. R., 1992. Fuzzy Kohonen Clustering Networks. *Proceedings of IEEE Int. Conf. on Fuzzy Systems 1992 (San Diego)*, 1035-1043.

Blayvas, I. and Kimmel, R., 2002. Efficient classification via multiresolution training set approximation. *CS Dept. Technical Report* CIS-2002-01.

Bors A. G., 2001. Introduction of the Radial Basis Function (RBF) Networks. *Online symposium of Electronic Engineering*, vol. 1 (1), 1-7.

Breiman, L., 1994. Bagging predictors. Technical Report no. 421, Department of Statistics, University of California, Berkeley.

Breiman, L., 1996. Bagging predictors. *Machine Learning*, vol. 24, 123–140.

Bresnahan, T. F., 1986. Measuring the Spillovers from Technical Advance: Mainframe Computers in Financial Services, *The American Economic Review*, vol. 76 (4), 742-755.

Brown, G., 2004. *Diversity in neural network ensembles*. PhD Thesis, The University of Birmingham.

Brown, G., Wyatt, J., Harris, R., and Yao, X., 2005. Diversity creation methods: a survey and categorisation. *Information Fusion,* vol. 6, 5-20.

Carpenter, G. A. and Grossberg, S., 1991. *Pattern Recognition by Self-Organizing Neural Networks*. MIT Press, London, England.

Castellano, G., Fanelli, A.M., and Mencar, C., 2004. An empirical risk functional to improve learning in a neuro-fuzzy classifier. *IEEE Transactions on Systems, Man, and Cybernetics*, Part B, vol. 34 (1), 725 – 731.

Chao, S. and Lihui, C., 2005. Feature dimension reduction for microarray data analysis using locally linear embedding. *3rd Asia-Pacific Bioinformatics Conference*, 211-217.

Chiang, J-H. and Chen, Y-H., 2002. Incorporating fuzzy operators in the decision network to improve classification reliability. *Computers & Electrical Engineering*, vol. 28 (6), 547-560.

Clerc, M., 2006. *Particle Swarm Optimization*, ISTE Publishing Company.

Cordella, L. P., De Stefano, C., Tortorella F., Vento, M., 1995. A method for improving classification reliability of multilayer perceptrons. *IEEE Transactions on Neural Networks*, vol. 6 (5), 1140-1147.

Cover, T. M. and Hart, P. E., 1967. Nearest neighbor pattern classification. *IEEE Trans. Inform. Theory*, vol. IT-13(1), 21–27.

Cristianini, N., and Shawe-Taylor, J., 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

Cung, V. D., Danjean, V., Dumas, J.-G., Gautier, T., Huard, G., Raddin, B., Rapine, C., Roch, J.-L., and Trystram, D., 2006. Adaptive and hybrid algorithms: classification and illustration on triangular system solving. *Proceedings of Transgressive Computing,* Granada, Spain, April 2006, 131-148.

Cunningham, P., 2000. *Overfitting and diversity in classification ensembles based on feature selection*. Dublin, Trinity College Dublin, Department of Computer Science, TCD-CS-2000-07.

Cybenko, G. V., 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems.* vol. 2, 303-314.

Dash, M. and Liu, H., 1997. Feature selection for Classification. *Intelligent Data Analysis,* vol. 1, 131-156.

Denoeux, T., 1995. A k-nearest neighbor classification rule based on Dempster-Shafer. *IEEE Transactions on Theory, Systems, Man and Cybernetics*, vol. 25 (5), 804-813.

Denoeux, T., 2000. A neural network classifier based on Dempster-Shafer theory. *IEEE Transactions on Systems, Man, and Cybernetics*, Part A, vol. 30, 131-150.

Deterding, D. H., 1989. *Speaker Normalisation for Automatic Speech Recognition*, University of Cambridge, PhD thesis.

Domeniconi, C. and Yan, B., 2005. On error correlation and accuracy of nearest neighbor ensemble classifiers. *Proceedings of the SIAM International Conference on Data Mining*, Newport Beach, California, 217-226.

Dorigo, M., Maniezzo, V., and Colorni, A., 1991. *Positive feedback as a search strategy*. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy.

Dorigo, M. and Blum, C, 2005. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344 (2–3), 243–278.

Dorigo, M. and Stützle, T., 2004. *Ant Colony Optimization*. MIT Press, Cambridge, MA.

Douglas, K. M., 2007. *General Method for Estimating the Classification Reliability of Complex Decisions Based on Configural Combinations of Multiple Assessment Scores*. PhD thesis, University of Maryland, USA.

Duda, R. O., Hart, P. E., and Stork, D. G., 2001. *Pattern Classification* (2nd ed.). John Wiley and Sons, University of Michigan.

Dvorak, J. and Savicky, P., 2007. Softening splits in decision trees using simulated annealing. Proceedings of ICANNGA 2007, Warsaw, (Ed.: Beliczynski et. al), Part I, LNCS 4431, 721-729.

Eggermont, J., Kok, J. N., and Kosters, W. A., 2004. Genetic programming for data classification: partitioning the search space. *Proceeding of Symposium of Applied Computing*, 1001-1005.

ELENA Project Artificial Databases, available on-line at http://www.dice.ucl.ac.be/neural-nets/Research/Projects/ELENA/databases/ARTIFICIAL/clouds/ (last accessed March 2008).

Enas, G. G. and Choi S. C., 1986. Choice of the smoothing parameter and efficiency of k-nearest neighbor classification. *Computers & Mathematics with Applications,* vol. 12 (2), 235-244.

Evgeniou, T., Pontil, M., and Elisseeff, A., 2004. Leave-one-out error, stability, and generalization of voting combination of classifiers. *Machine Learning*, vol. 55(1), 71-97.

Fard, M. M., 2006. Ensemble Learning with Local Experts. *IEEE Computer Society ezine "Looking.Forward" student magazine 14th edition*. Available online at:

http://www.computer.org/portal/cms_docs_ieeecs/ieeecs/communities/students/lookin g/2006fall/05.pdf (last accessed: August 2008).

Fawcett, T., 2006. An introduction to ROC analysis. *Pattern Recognition Letters*, vol. 27, 861-874.

Fidler, S. and Leonardis, A., 2003. Robust LDA classification by subsampling. *Proceedings of Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, vol. 8, 97-104.

Fisher, R. A., 1938. The statistical utilization of multiple measurements. *Annals of Eugenics*, vol. 8, 376-386.

Frohlich, H. and Chapelle, O., 2003. Feature selection for support vector machines by means of genetic algorithms. *15th IEEE International Conference on Tools with Artificial Intelligence*, 142-148.

Freund Y. and Schapire R. E., 1996, Experiments with a new boosting algorithm. *Proceedings of the Thirteenth International Conference Machine Learning*, 148–156.

Fumera, G., Roli, F., and Giacinto G., 2000. Reject option with multiple thresholds. *Pattern Recognition*, vol. 33 (12), 2099-2101.

Fung, G., Dundar, M. M., Bi, J., and Rao, B., 2004. A fast iterative algorithm for fisher discriminant using heterogeneous kernels. *Proceedings of the 21st International Conference on Machine Learning*. Available online at http://portal.acm.org/citation.cfm?id=1015409 (last accessed: May 2008).

Gao, Q-B. and Wang, Z-Z., 2007. Centre-based nearest neighbor classifier. *Pattern Recognition,* vol. 40, 346-349.

Getz, E., Levine, E., and Domany, E., 2001. Coupled Two-Way Clustering Analysis of Gene Microarray Data (2001). Found online at: arXiv:physics/0004009v1 (last accessed December 2008).

Ghosh, A. K., 2006. On optimum choice of k in nearest neighbour classification. *Computational Statistics & Data Analysis,* vol. 50, 3113-3123.

Giusti, N., Masulli, F., and Sperduti, A., 2002. Theoretical and experimental analysis of a two-stage system for classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24 (7), 893-904.

Glackin, C., 2008. Classification using a Fuzzy Spiking Neural Network. *Proceedings of UKCI Conference*, 117-122.

Graham, R. L. and Hell, P., 1985. On the history of the minimum spanning tree problem. *Annals of the History of Computing*, vol. 7(1), 43–57.

Hall, M., 1999. *Correlation-based Feature Selection for Machine Learning*. PhD thesis from the University of Waikato, Hamilton, New Zealand.

Hamers, B. and Suykens, J. A. K, 2003. Coupled transductive ensemble learning of kernel models. *Internal Report 03-172*, ESAT-SISTA, K. U. Leuven.

Hand, D. J. and Vinciotti, V., 2003. Choosing k for two-class nearest neighbour classifiers with unbalanced classes. *Pattern Recognition Letters,* vol. 24, 1555-1562.

Hattori, K. and Takahashi, M., 2000. A new edited k-nearest neighbor rule in the pattern recognition problem. *Pattern Recognition,* vol. 33, 521-528.

Hendrickx, I. and Van den Bosch, A., 2004. Maximum-entropy parameter estimation for the k-NN modified value-difference kernel. *Proceedings of the 16th Belgian-Dutch Conference on Artificial Intelligence*, Groningen, The Netherlands, 19-26.

Hochreiter, S. and Obermay, K., 2003. Feature selection and classification on matrix data: from large margins to small covering numbers. Chapter 16 in *Advances in Neural Information Processing Systems*. MIT Press.

Holden, N. and Fietas, A. A., 2007. A hybrid PSO/ACO algorithm for classification. *Proceedings of GECCO'07,* London, England, United Kingdom, 2745-2750.

Hopfield, J. J., 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the USA*, vol. 79 (8), 2554-2558.

Horton, P. and Nakai, K., 1996. A probablistic classification system for predicting the cellular localization sites of proteins. *Proceedings of Intelligent Systems in Molecular Biology*, St. Louis, USA, 109-115.

Inza, I., Larrañaga, P., Sierra, B., Etxeberria, R., Lozano, J. A., and Peña, J. M., 1999. Representing the behaviour of supervised classification learning algorithms by Bayesian networks. *Pattern Recognition Letters,* vol. 20, 1201-1209.

Jain, A. K., Duin, R. P. W., Mao, J., 2000. Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22 (1), 4-38.

Jensen, R., and Shen, Q., 2005. Fuzzy-rough data reduction with ant colony optimization. *Fuzzy Sets and Systems*, 149 (1), 5-20.

Jeon , B. H., Lee, S. U., and Lee, K. M., 2002. Face detection using the 1st-order RCE classifier. *Proceedings of International Conference on Image Processing*, vol. 2, 125-128.

Jiang, Y. and Zhou, Z.-H., 2004. Editing training data for kNN classifiers with neural network ensemble. *Lecture Notes in Computer Science*, vol. 3173, 356-361.

Karaboga.D, 2005. An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department.

Kaufman, L. and Rousseeuw, P. J., 1990. *Finding groups in data*. Wiley Interscience Publications, New York, 83-85.

Kawahara, K. and Shibata, T., 2005. A new distance measure employing element-significance factors for robust image classification. *Proceedings of 13th European Signal Processing Conference (EUSIPCO 2005)*, Antalya, Turkey.

Kaylani, T. and Dasgupta, S., 1994. A new method for initializing radial basis function classifiers systems. *IEEE International Conference on Man, and Cybernetics*, vol. 3, 2584-2587.

Keller J. M., Gray M. R., and Givens J. A., Jr., 1985. A fuzzy k-nearest neighbor algorithm. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 15 (4), 580-584.

Kennedy, J. and Eberhart, R., 1995. Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks*, 1942-1948.

Kent Ridge Bio-medical Data Set Repository. Available online at http://sdmc.lit.org.sg/GEDatasets/Datasets.html (last accessed January 2008).

Khan, M., Ding, Q. and Perrizo, W., 2002. K-nearest neighbors classification of spatial data streams using P-trees. *Proceedings of the PAKDD*, 517-528.

Kohonen, T., 1982. Self-organized formation of topologically correct feature maps. *Biological Cybernetics,* vol. 43, 59-69.

Kohonen, T., 2001. *Self-Organizing Maps* (Third Extended Edition). Springer, Berlin, Heidelberg, New York.

Komorowski J., Pawlak Z., Polkowski L., and Skowron A., 1998. Rough sets: a tutorial. In Pal, S.K. & Skowron, A. (Eds.): *Rough-Fuzzy Hybridization: A New Trend in Decision-Making*. Springer-Verlag, Singapore. Available online at http://folli.loria.fr/cds/1999/library/pdf/skowron.pdf (last accessed July 2008).

Kononenko, I., Simec, E., and Robnik-Sikonja, M., 1997. Overcoming the myopia of inductive learning algorithms with RELIEFF. *Applied Intelligence,* vol. 7, 39-55.J. B.

Kruskal, J. B. Jr., 1956. On the shortest spanning subtree of a graph and the travelling salesman problem. *Proceedings of the American Mathematical Society*, vol. 7(1), 48–50.

Kukar, M. and Kononenko, I., 2002. Reliable Classifications with Machine Learning. *13th European Conference on Machine Learning*, 219-231.

Kuncheva, L. I., 2000. *Fuzzy Classifier Design*. Springer-Verlag, Heidelberg.

Kuncheva, L. I and Whitaker, C. J., 2003. Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy. *Machine Learning*, vol. 51, 181-207.

Lai, J. Z. C., Liaw, Y-C., and Liu, J., 2007. Fast k-nearest-neighbor search based on projection and triangular inequality. *Pattern Recognition,* vol. 40, 351-359.

Lam, L. and Suen, S. Y., 1997. Application of majority voting to pattern recognition: An analysis of its behavior and performance. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 27, 553–568.

Lecocke, M. and Hess, K., 2005. An empirical study of univariate and GA-based feature selection in binary classification with microarray data. *UT MD Anderson Cancer Center Department of Biostatistics Working Paper Series*, Working Paper 5. Available online at http://www.bepress.com/mdandersonbiostat/paper5 (last accessed February 2008).

Ledward, A. Ensemble classifiers for machine learning. Available online at http://www.ee.hawaii.edu/~kuh/ee645.f99/lectures.html (last accessed: April 2008).

Lemeni, I. and Tepus, N., 2008. A SOM method for classes overlap degree evaluation. *Proceedings of 3rd International Multi-Conference on Computing in the Global Information Technology*, 171-176.

Li, X. B., and Jacob, V. S., 2008. Adaptive data reduction for large-scale transaction data. *European Journal of Operational Research*, vol. 188 (3), 910-924.

Liu, H., Li, J., and Wong, L., 2002.A comparative study on feature selection and classification methods Using gene expression profiles and proteomic patterns. *Genome Informatics,* vol. 13, 51-60.

Mainar-Ruiz, G. and Pérez-Cortes, J. S., 2006. Approximate nearest neighbor search using a single space-filling curve and multiple representations of the data points. *Proceedings of 18th International Conference on Pattern Recognition (ICPR 2006)*, Hong Kong, China, 502-505.

Manocha, S. and Girolami, M. A., 2007. An empirical analysis of the probabilistic K-nearest neighbour classifier. *Pattern Recognition Letters,* vol. 28, 1818-1824.

McCulloch, W. and Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, vol. 7, 115 - 133.

McKenzie, D. and Low, L. H., 1992. The construction of computerized classification systems using machine learning algorithms: an overview. *Computers in Human Behavior*, vol. 8, 155-167.

Michalak K., and Kwa/Snicka, H., 2006. Correlation-based feature selection strategy in classification systems. *International Journal of Applied Mathematics and Computer Science*, vol. 16 (4), 503–511.

Michalewicz, Z., 1995. A survey of constraint handling techniques in evolutionary computation methods. *Proceedings of the Fourth Annual Conference on EP*, 135-155.

Michalewicz, Z and Schoenauer, M., 1996. Evolutionary algorithms for constrained parameter optimization problems, *Evolutionary Computation*, vol. 4, 1-32.

Mitchell, T., 1997. *Machine Learning*. The McGraw-Hill Companies Inc.

Mitchell, S. W., Remmel, T. K., Csillag, F. and Wulder, M., 2008. Distance to second cluster as a measure of classification confidence. *Remote Sensing of Environment,* vol. 112, 2615-2626.

Mitchie, D., Spiegelhalter, D. J., and Taylor, C. C., 1994. *Machine learning, neural and statistical classification.* Ellis Horwood Limited.

Montgomery, D., Swinnen, G., and Vanhoof, K., 1997. Comparison of some AI and statistical classification methods for a marketing case. *European Journal of Operational Research*, vol. 103 (2), 312-325.

Moreno-Seco, F., Mico, L. and Oncina, J., 2003. A modification of the LAESA algorithm for approximated k-NN classification. *Pattern Recognition Letters,* vol. 24, 47-53.

Mukkamala, S., Liu, Q., Veeraghattam, R., and Sung, A. H., 2005. Computational intelligent techniques for tumor classification (using microarray gene expression data). *International Journal of Lateral Computing*, vol. 2 (1), 38-45.

Musavi, M. T, Ahmed, W., Chan, K. H., Faris, K. B., and Hummels, D. M., 1992. On the training of radial basis function classifiers. *Neural Networks,* vol. 5, 595-603.

Nakashima, A. and Ogawa, H., 2000. Noise suppression in training examples for improving generalization capability. *Neural Networks*, vol. 14, 459-469.

Niranjan, M. and Fallside, F., 1990. Neural networks and radial basis functions in classifying static speech patterns. *Computer Speech and Language,* vol. 4, 275-289.

Okun, O. and Priisalu, H., 2007. Unsupervised data reduction. *Signal Processing,* vol. 87 (9), 2260-2267.

Papadopoulos, H., 2004. *Qualified Predictions for Large Data Sets*. PhD thesis, Royal Holloway, University of London.

Papadopoulos, H., 2008. Inductive conformal prediction: theory and application to neural networks. *Tools in Artificial Intelligence*, P. Fritzsche, Ed. Croatia: I-Tech, 315-330.

Papadopoulos, H., Vovk, V., and Gammerman, A., 2007. Conformal prediction with neural networks, *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence* (ICTAI'07), vol. 2, 388-395, Patras, Greece, October 2007, IEEE Computer Society, Los Alamitos, CA.

Pham, D.T., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., and Zaidi, M., 2006. The Bees Algorithm - A Novel Tool for Complex Optimisation Problems. *Proceedings of IPROMS 2006 Conference*, 454-461.

Polat, K. and Güneş, S., 2007. A novel data reduction method: distance based data reduction and its application to classification of epileptiform EEG signals. *Applied Mathematics and Computation.* In Press, Corrected Proof, Available online at: www.sciencedirect.com, (last accessed March 2008).

Poli, R., Kennedy, J., Blackwell, and T., Freitas, A. (editors), 2008. *Particle Swarms: The Second Decade*, Hindawi Publishing Corp US SR.

Pomeroy, S.L, et al, 2002**.** Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature* vol. 415, 436-442.

Ray, S. and Page, D., 2003. Skewing: an efficient alternative to look ahead for decision tree induction. *Proceedings of the International Joint Conference on Artificial Intelligence*, 601-612.

Richiardi, J., Prodanov, P. and Drygajlo, A., 2005. A probabilistic measure of modality reliability in speaker verification. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, 709-712, Philadelphia, USA.

Roy, A., Govil, S., and Miranda, R., 1995. An algorithm to generate radial basis function (RBF)-like nets for classification problems, *Neural networks*, vol. 8 (2), 179-201.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J., 1986. Learning representations by back-propagating errors. *Nature*, vol. 323, 533-536.

Russell, S. and Yoon, V., 2008. Applications of wavelet data reduction in a recommender system. *Expert Systems with Applications,* vol. 34 (4), 2316-2325.

Ruta, D. and Gabrys, B., 2003. Physical field models for pattern classification. *Soft Computing Journal*, vol. 8 (2), 126-141.

Ruta, D. and Gabrys, B., 2005. Classifier selection for majority voting. *Information Fusion,* 6, 63–81.

Safavian, S. R. and Landgrebe, D., 1991. A Survey of Decision Tree Classifier Methodology. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21 (3), 660-674.

Sansone, C., Tortorella, F. and Vento, M., 2001. A classification reliability driven reject rule for multi-expert systems. *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 15 (6), 1-19.

Santos-Pereira, C. M. and Pires, A. M., 2005. On optimal reject rules and ROC curves. *Pattern Recognition Letters*, vol. 26 (7), 943-952.

Sawa, T. and Ohno-Machado, L., 2003. A neural network-based similarity index for clustering DNA microarray data. *Computers in Biology and Medicine*, vol. 33, 1-15.

Schapire, R. E., 2003. The boosting approach to machine learning: an overview. *Nonlinear Estimation and Classification*, Springer.

Shanin, M. A., Tollner, E. W. and McClendon, R. W., 2001. Artificial intelligence classifiers for sorting apples based on watercore. *Journal of Agricultural Engineering Resources*, vol. 79 (3), 265-274.

Shang, C., and Shen, Q., 2006. Aiding classification of gene expression data with feature selection: a comparative study. *Computational Intelligence Research*, 1(1), 68-76.

Sharkey, A. J. C., Sharkey, N. E., Gerecke, U. and Chandroth, G. O., 2000. The "Test and Select" Approach to Ensemble Combination. *Multiple Classifier Systems,* 30-44.

Shen, Q., and Jensen, R., 2008. Approximation-based feature selection and application for algae population estimation. *Applied Intelligence*, 28(2), 167-181.

Shipp, C. A. and Kuncheva, L. I., 2002. Relationships between combination methods and measures of diversity in combining classifiers. *Information Fusion,* 3, 135–148

Siler, W., and Buckley, J. J., 2005. *Fuzzy Expert Systems and Fuzzy Reasoning*. John Wiley and sons, inc.

Skyt, J., Jensen, C. S. and Pedersen, T. B., 2008. Specification-based data reduction in dimensional data warehouses. *Information Systems,* vol. 33 (1), 36-63.

Song, H. and Feng, H. Y., 2007. A global clustering approach to point cloud simplification with a specified data reduction ratio. *Computer-Aided Design*, in Press, Corrected Proof, Available online at: www.sciencedirect.com (last accessed March 2008).

Sotoca, J. M., Sanchez J. S. and Pla F., 2003. Estimating feature weights for distance-based classification. *Pattern Recognition in Information Systems PRIS2003*, Angers, France, 156-166, Ed. ICEIS PRESS, ISBN: 972-98816-3-4.

Specht, D. F., 1990. Probabilistic neural networks. *Neural Networks*, vol. 3 (1), 109-118.

Stork, D. G. and Yom-Tov, E., 2004. *Computer manual in Matlab to accompany Pattern Classification* 2nd edition. Wiley-Interscience.

Sugiyama M. and Ogawa, H., 2000. Incremental projection learning for optimal generalization. *Neural Networks*, vol. 14, 53-66.

Sykacek, P. and Roberts, S. J., 2003. Adaptive classification by variational kalman filtering. *Neural Information Processing Systems*, vol. 15, 737-744.

Takagi, T. and Sugeno, M., 1985. Fuzzy identification of systems and its application to modeling and control. *IEEE Trans. on Syst., Man & Cybernetics,* vol. 15, 116-132.

Tan, S., Wang, Y., and Cheng, X., 2007. Text feature ranking based on rough-set theory. *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence*, 659-662.

Tang, K., Suganthan, P. N., Yao, X. and Qin, A. K., 2005. Linear dimensionality reduction using relevance weighted LDA. *Pattern Recognition*, vol. 38, 485-493.

Tasoulis, D. K., Plagianakos, V. P. and Vrahatis, M. N., 2006. Unsupervised clustering in mRNA expression profiles. *Computers in Biology and Medicine*, vol. 36, 1126-1142.

Thien, M. Ha, 1996a. An experimental study of the optimal class-selective rejection rule. Available online at: http://citeseer.ist.psu.edu/103812.html (last accessed: December 2007).

Thien, M. Ha, 1996b. Application of the optical class-based rejection rule to the detection of abnormalities in OCR databases. Available online at: http://citeseer.ist.psu.edu/135692.html (last accessed: December 2007).

Thireou, T., Guivernau, J. L. R., Atlamazoglou, V., Ledesma, M. J., Pavlopoulos, S., Santos, A., and Kontaxakis, G., 2006. Evaluation of data reduction methods for dynamic PET series based on Monte Carlo techniques and the NCAT phantom. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment,* vol. 569 (2), 389-393.

Toplak, W., 2008. A comparison of SOM and ALEV for data reduction purposes in transport telematics. *Proc. of 26th IASTED International Conference on Artificial Intelligence and Applications*, CD Proceedings ISBN: 978-0-88986-710-9, 181-186.

Torra, V., Abowd, J. M., and Domingo-Ferrer, J., 2006. Using Mahalanobis distance-based record linkage for disclosure risk assessment. *Lecture Notes in Computer Science*, 4302, 233-242.

Tsymbal, A. and Puuronen, S., 2000. Bagging and boosting with dynamic integration of classifiers. *Principles of Data Mining and Knowledge Discovery*, 116-125.

UCI Repository: http://archive.ics.uci.edu/ml/datasets.html (last accessed: February 2008).

Valafar, F. and Ersoy, O., 1994. *Parallel, Probabilistic, Self-Organising, Hierarchical Neural Networks*. Purdue University School of Electrical Engineering.

Van den Bosch, A., 2004. Feature transformation through rule induction: a case study with the k-NN classifier. J. Fürnkrantz (Ed.), *Proceedings of the ECML/PKDD 2004 Workshop on Advances in Inductive Rule Learning*, Pisa, Italy, 1-16.

Vanherzeele, J., Guillaume, P., Vanlanduit, S., and Verboven, P., 2006. Data reduction using a generalized regressive discrete Fourier series. *Journal of Sound and Vibration* vol. 298 (1-2, 22), 1-11.

Vanlanduit, S., Cauberghe, B., Guillaume, P., Verboven, P., and Parloo, E., 2006. Reduction of large frequency response function data sets using a robust singular value decomposition. *Computers & Structures*, vol. 84 (12), 808-822.

Veal, B., 2008. Similarity measures for classification of binary data. Proceedings of UKCI Conference, 71-76.

Vapnik, V. N., 2000a. *The Nature of Statistical Learning Theory*, Second Ed. Springer-Verlag.

Vapnik, V., 2000b. *Support Vector Machines and Other Kernel-based Learning Methods*. John Shawe-Taylor & Nello Cristianini, Cambridge University Press.

Voulgaris, Z. and Magoulas, G., 2008a. Extensions of the k nearest neighbour methods for classification problems. *Proceedings of the 26th IASTED International Conference on Artificial Intelligence and Applications*, 23-28.

Voulgaris Z. and Magoulas G. D., 2008b, A discernibility-based approach to feature selection for microarray data. *CD Proceedings of the IEEE International Conference of Intelligent Systems*, Varna, Bulgaria, Sept. 2008, IEEE Press.

Voulgaris, Z. and Magoulas, G. D., 2008c. Dimensionality reduction for feature and pattern selection in classification problems. *Proceeding of The Third International Multi-Conference on Computing in the Global Information Technology*, Athens, Greece, July 2008, 160-165.

Voulgaris, Z. and Magoulas, G. D., 2008d. Discernibility-based approach for creating ensembles in pattern classification applications. *Proceedings of the UKCI Conference*, Leicester U.K., Sept. 2008, 195-199.

Voulgaris, Z. and Mirkin, B., 2008a. Choosing a discernibility measure for reject-option at a set of classifiers. *Pattern Recognition Letters*, under revision.

Voulgaris, Z. and Mirkin, B., 2008b. Optimising a reliability measure for classification. *Proceedings of the UKCI Conference*, Leicester U.K., Sept. 2008, 43-46.

Wang, H. and Bell, D., 2004. Extended k-nearest neighbours based on evidence theory. *The Computer Journal*, vol. 47 (6), 662-672.

Wang, J., Nesboric, P., and Cooper, L. N., 2006. Neighborhood size selection in the k-nearest-neighbor rule using statistical confidence. *Pattern Recognition*, vol. 39, 417-423.

Wang, J., Nesboric, P., and Cooper, L. N., 2007. Improving nearest neighbor rule with a simple adaptive distance measure. *Pattern Recognition Letters*, vol. 28, 207-213.

Warfield, S., 1996. Fast k-NN classification for multichannel image data. *Pattern Recognition Letters,* vol. 17, 713-721.

Weston, J. , Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., and Vapnik, V., 2001. Feature selection for SVMs. *Advances in Neural Information Processing Systems,* vol. 13, 668-674

Winston, R., 1969. *A heuristic program that constructs decision trees*. MIT project MAC, Memo #173.

Winston, P., 1992. Learning by building identification trees. *Artificial Intelligence*, Addison-Wesley Publishing Company.

Wu, Y., Ianakiev, K., and Gornidaraju, V., 2002. Improved k-nearest neighbor classification. *Pattern Recognition,* vol. 35, 2311-2318.

Yu, K. and Ji, L., 2002. Karyotyping of comparative genomic hybridization human metaphases using kernel nearest-neighbor algorithm. *Cytometry*, vol. 48, 202–208.

Yu, S., De Backer, S., Scheunders, P., 2002. Genetic features selection combined with fuzzy nearest neighbour classifiers for hyperspectral satellite imagery. *Pattern Recognition Letters*, vol. 23, 183-190.

Zadeh, L. A., 1965. Fuzzy sets. *Information and Control,* vol. 8, 338-353.

Zhong, P. and Fukushima, M., 2007. A regularized nonsmooth Newton method for multi-class support vector machines. *Optimization Methods and Software,* vol. 22, 225-236.

Zhou, Z.-H. and Jiang, Y., 2004. NeC4.5: neural ensemble based C4.5. *IEEE Transactions of Knowledge Data Engineering,* vol. 16, 770-773.

Zhou, C. Y. and Chen, Y. Q., 2006. Improving nearest neighbor classification with cam weighted distance. *Pattern Recognition,* vol. 39, 635-645.

Zio, E., Baraldi, P., and Gola, G., 2007. Feature-based classifier ensembles for diagnosing multiple faults in rotating machinery. *Applied Soft Computing*, in press.

Appendix A – Dataset Description

Bupa Liver: this is a dataset that has been used extensively in the literature (Inza et. al., 1999; Jiang & Zhou, 2004; Zhou & Jiang, 2004). The first five variables are all blood tests, which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption. The sixth variable is the number of half-pint equivalents of alcoholic beverages drunk per day. There are two classes, denoting where a patient is diagnosed with liver disorder or not. The dataset was created by BUPA Medical Research Ltd.

Pima Indians: a medical dataset comprising of various cases of female patients of the Pima Indian heritage living near Phoenix, Arizona, USA. The disease involved is diabetes and it is predicted using various medical measures, as well as other features. This dataset was owned by National Institute of Diabetes and Digestive and Kidney Diseases and has been used in several studies (Blayvas & Kimmer, 2002; Eggermont et. al., 2003; Sykacek & Roberts, 2003).

Breast Cancer Wisconsin: This breast cancer database was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. It involves benign or malignant cases of breast cancer (class attribute) for 683 patients, over a period of three years. Various researchers such as Glackin (2008) and Veal (2008) have made use of it in classification applications.

Statlog Heart Disease: a medical dataset involving patients examined for heart disease from Statlog that is considered a typical benchmark (Brown, 2004; Kononenko et al., 1997; Ray & Page, 2003; Tang et al., 2005). It includes 13 attributes, some of which are categorical while others are real numbers. There are two classes, relating to either presence or absence of heart disease.

Statlog Vehicle Silhouettes (used in used in Denoeux, 2000; Kononenko et. al., 1997; Ray & Page, 2003; Tang et al., 2005, Zhong & Fukushima, 2007): This dataset comprises of data related to 2D silhouettes of four different types of vehicles (as seen from one of many different angles). Four “Corgie2” model vehicles were used for the experiment: double decker bus, Cheverolet van, Saab 9000 and Opel Manta 400. This choice was made so that the different types of vehicles can be more easily distinguishable (compared to different types of cars).

Boston Housing (used in Brown, 2004; Fung et. al., 2004; Hamers & Suykens, 2003): This dataset concerns housing values in suburbs of Boston. Originally the class attribute (median value of owner-occupied homes in $1000's) is continuous but in our experiments we use a discrete version of it, where all cases are characterised as easily affordable, medium valued and expensive. The 13 features involved for the prediction of the class describe the borough of each house with various demographics and characteristics of each house.

Wine (used in Aeberhard et al, 1992): this dataset has to do with the chemical analysis of wines, from three different cultivars of a particular region of Italy. The 13 features it comprises of represent some of the constituents found in each of the 3 types of wines (class labels).

Glass (used in Zhong & Fukushima, 2005): This dataset involves a study of classification of 7 different types of glass, based on 9 characteristics, most of which have to do with particular chemical elements found in them. The relevant research which brought about the creation of this dataset was motivated by criminological investigation: at the scene of the crime, the glass left can be used as evidence, given it is correctly identified.

Vowel (used in Deterding, 1989): This dataset consists of a three dimensional array: voweldata [speaker, vowel, input]. The speakers are indexed by integers 0-89. (Actually, there are fifteen individual speakers, each saying each vowel six times.) The vowels are indexed by integers 0-10. For each utterance, there are ten floating-point input values, with array indexes 0-9 (features). There are 528 patterns in the dataset.

DLBCL has to do with Diffuse Large B-Cell Lymphomas (DLBCL) and follicular lymphomas (FL) . These are two B-cell lineage malignancies having very different clinical presentations, natural histories and response to therapy. However, FLs frequently evolve over time and acquire the morphologic and clinical features of DLBCLs and some subsets of DLBCLs have chromosomal translocations characteristic of FLs. The gene-expression based classification model was built to distinguish between these two lymphomas which have 58 and 19 samples in the dataset respectively. The number of features of the dataset is 7070, each one representing a particular gene.

Colon cancer (used in Getz et al, 2001): a medical dataset containing expression levels of 2000 genes taken in 62 different samples. For each sample it is indicated whether it came from a tumor biopsy or not. Numbers and descriptions for the different genes are also given. This dataset is used in many different research papers on gene expression data.

Leukaemia (used in Getz et al, 2001): this dataset contains expression levels of 7129 genes taken over 72 samples. Labels indicate which of two variants of leukaemia is present in the sample (AML, 25 samples, or ALL, 47 samples). This dataset is of the same type as the colon cancer dataset and is usually used for the same kind of experiments.

CNS (used in Pomeroy et al, 2002): This dataset involves patients outcome prediction for central nervous system embryonal tumor. *Survivors* are patients who are alive after treatment whiles the *failures* are those who succumbed to their disease. The data set contains 60 patient samples, 21 are survivors (labelled as "Class 1") and 39 are failures (labelled as "Class 0"). There are 7129 genes (features) in the dataset.

Pendigits (used in Alimoglu, 1996). This is a dataset for character recognition, based on a collection of 250 samples from 44 writers (yielding a total of about 11000 patterns). For its creation a WACOM PL-100V pressure sensitive tablet with an integrated LCD display and a cordless stylus were used. The writers were asked to write 250 digits in random order inside boxes of 500 by 500 tablet pixel resolution. There are 15 integer features and a class label array comprising of 10 different classes (one for each digit).

Magic Gamma Telescope (used in Dvorak & Savicky, 2007). This is a large dataset regarding the MC generated to simulate registration of high energy gamma particles in a ground-based atmospheric Cherenkov gamma telescope using the imaging technique. It comprises of 19020 patterns and 10 continuous features. A relatively balanced dataset, it has two classes denoting signal or background.

Iris (used in Zhong & Fukushima, 2005): Probably the best known dataset in the Pattern Recognition literature. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2, which are not linearly separable from each other.

Statlog **Heart** (used in Brown, 2004): This dataset is a heart disease database similar to the substantially larger Heart Disease database but in a slightly different form. It has 270 cases (patterns) of patients some of whom have heart disease (class label). There are 13 features in the datasets, quite diverse in nature, while about half of them are real numbers. They involve certain medical measurements and characteristics, as well as other attributes such as sex and age.

E.coli (used in Horton & Nakai, 1996): this is a medical dataset involving the e.coli virus. It comprises of 336 patterns and 7 features (most of which are real numbers, although there are two binary ones as well). There are two classes, representing the presence or absence of the e.coli disease.

Appendix B – MATLAB Code

Spherical Index of Discernibility (Algorithm 3.1)

```
function [z Z] = IDND5(I,O)

% Yet another IDND alternative

Z1 = [];
[N na] = size(I);

a = length(O);
if a ~= N
    error('I and O sizes do not match!')
end

D(N,2) = 0; ZZ(N) = 0;
for i = 1:N
    D = [sum((rm(I(i,:),N,1) - I).^2,2) O];
    D(i,:) = [];
    f = (D(:,2) == O(i));
    lf = sum(f);
    r = sum(D(f,1))/lf;
    f2 = (D(:,1) <= r);
    lf2 = sum(f2);
    c = sum(D(f2,2)==O(i));
    ZZ(i) = c / lf2;
end

Z = ZZ;
z = sum(Z >= .5)/N;
```

Note: *swap* is a function that interchanges the contents of two variables a and b
rm is a light version of the build-in *repmat* function.

Harmonic Index of Discernibility (Algorithm 3.2)

```
function [z Z] = IDND6(I,O,q,Q)

% Yet another IDND alternative

N = size(I,1);
a = length(O);
if a ~= N
    error(‘I and O sizes do not match!’)
end

if nargin < 3
    [q Q] = nc(O);
end

class{q} = []; NN(q) = 0;
for I = 1:q
    class{I} = find(O==Q(i));
    NN(i) = length(class{I});
end

d(N,N) = Inf;
```

```
for I = 1:(N-1)
    X = ones((N-I),1)*I(I,:);
    d((i+1):N,i) = sum( (X - I((i+1):N,:)).^2,2 ) ;
    d(i,i) = Inf;
end

D = d + d'; % A compact form of the distance matrix
ZZ(N) = 0;

for i = 1:N
    f = (O(i) == Q);
    otherclasses = 1:N;
    otherclasses(class{f}) = [];
    z1 = sqrt(NN(f) / (sum(1./(d(class{f}) + eps)) - eps));
    z2 = sqrt(( N - NN(f) ) / sum(1./(d(otherclasses) + eps)- eps));
    ZZ(i) = max( ((z2 - z1) / (z1 + z2)) , 0);
end

Z = ZZ;
z = sum(Z>0.5)/N;
```

Discernibility k Nearest Neighbour Classifier (Algorithm 4.1)

```
function [y Dcy cpu] = DNNK(P,T,PT,K)

if nargin < 4
    K = 5;
end

[N na] = size(P);
n = size(PT,1);
[q, Q, cs] = nc(T);

t = cputime;

class{q} = [];
for I = 1:q
    class{I} = P((T==Q(i)),:);
end
[z Z] = IDND5(P,T);


cpu = cputime – t;

% N = number of elements in training set
% n = number of elements in testing set
% na = number of attributes (=features)
% q = number of classes
% Q = classes vector

Y(n) = 0; DCY(n) = 0;
for I = 1:n % Do this for every element to be classified
    z = zeros(q,1);
    X = PT(I,:);
    d(1:N,1:2) = 0; % Squared distance vector
    D(:,1) = sum(((rm(X,N,1) – P).^2),2);
    D(:,2) = T;
    D(:,3) = Z;
    DK = ksr(D,K);
    for j = 1:q
```

```
        f = (DK(:,2) == Q(j));
        z(j) = sum(DK(f,3)./(DK(f,1)+eps));
    end

    [M ind] = max(z);
    Y(i) = Q(ind);
    DCY(i) = M / sum(z);
end % of FOR I

y = Y;
Dcy = DCY;
```

Weighted k Nearest Neighbour Classifier (Algorithm 4.2)

```
function [y DCy cpu] = WKNN(P,T,PT,K)

%  WkNN - Weighted Nearest Neighbour K
%
%  A variation of NNK, using the independence vetors of each
attribute, provided
%  by IDND5, as weights for the distance vectors used for the
classification.

if nargin < 4
    K = 5;
end

[N na] = size(P);
n = size(PT,1);
[q, Q] = nc(T);

t = cputime;

Z = zeros(N,na);
for i = 1:na
    [z Z(:,i)] = IDND5(P(:,i),T);
end

cpu = cputime - t;

for i = 1:n
    d = sum(((repmat(PT(i,:),N,1) - P).^2),2);
    [m ind] = min(d);
    w = Z(ind,:)';
    d(:,1) = ((repmat(PT(i,:),N,1) - P).^2)*w;
    d(:,2) = T;
    ds = sortrows(d); % Sorted distances
    D = ds(1:K,:); % K nearest neighbours
    for j = 1:q
        c(j) = length(find(D(:,2)==Q(j))); % Class counter
    end
    [M ind] = max(c);
    DCY(i) = M/(sum(c));
    Y(i) = Q(ind);
end

y = Y;
DCy = DCY;
```

Variable k Nearest Neighbour Classifier (Algorithm 4.3)

```
function [Y DCY K cpu] = VKNN(P,T,PT)

% NNVK - Nearest Neighbour with Variable K parameter


% Some fundamental variables defined here
N = size(P,1);
n = size(PT,1);
[q, Q] = nc(T);

% N = number of elements in training set
% n = number of elements in testing set
% na = number of attributes (=features)
% q = number of classes
% Q = classes vector

t = cputime;

D = dmatrix(P); % the distance matrix of the training set
BK(N,2) = 0; % Best K list
minN = min((N-1),20);
tempDC(minN) = 0;
for i = 1:N
    temp = D(i,:)';
    [st(:,1) ind] = sort(temp);
    st(:,2) = T(ind);
    for j = 1:minN
        temp = st(1:j,:);
        a = sum(temp(:,2)==T(i));
        tempDC(j) = a/j;
    end
    [M ind] = max(tempDC);
    BK(i,:) = [i ind];
end
clear D temp
cpu = cputime - t;
k(n) = 0; Y(n) = 0; DCY(n) = 0;

for i = 1:n
    x = PT(i,:);
    temp(:,1) = sum((rm(x,N,1) - P).^2,2)';
    temp(:,2) = 1:N;
    temp(:,3) = T;
    [TEMP IND] = sort(temp(:,1));
    ind = IND(1);

    k(i) = BK(temp(ind,2),2); % this is the optimum k of the nearest
neighbour

    st = [TEMP temp(IND,2:3)];
    knn = st(1:k(i),3);
    for j = 1:q
        a(j) = sum(knn==Q(j));
    end
    [M ind] = max(a);
    Y(i) = Q(ind);
    DCY(i) = M / sum(a);
end
```

```
K = sum(k)/n;
```

Class-Based k Nearest Neighbour Classifier (Algorithm 4.2)

```
function [y DCy cpu] = CBKNN(P,T,PT)

[N na] = size(P);
n = size(PT,1);
[q, Q] = nc(T);

t = cputime;

for i = 1:q
    f = find(T==Q(i));
    class{i} = P(f,:);
    cs(i) = length(f); % Class Size
end

cputemp = cputime - t;

% N = number of elements in training set
% n = number of elements in testing set
% na = number of attributes (=features)
% q = number of classes
% Q = classes vector

for i = 1:n % Do this for every element to be classified
    X = PT(i,:);
    t = cputime;
    for k = 1:( min( min(cs),12 ) )
        for j = 1:q
            d(1:N,1:2) = 0; % Squared distance vector
            d = sum(((repmat(X,cs(j),1) - class{j}).^2),2);
            ds = sort(d); % Sorted distances
            D(j) = harmean(ds(1:k),1,.0001); % Average distance of K
nearest neighbours of class
        end
        [m ind] = min(D);
        Ytemp(k) = Q(ind);
        if m == 0
            DCYtemp(k) = 1;
        else
            DCYtemp(k) = 1/(m*sum(1./D));
        end
    end % of FOR k
    cputemp = cputemp + cputime - t;
    [M ind] = max(DCYtemp);
    Y(i) = Ytemp(ind);
    DCY(i) = DCYtemp(ind);
end % of FOR i

y = Y;
DCy = DCY;
cpu = cputemp;
```

Net Reliability (Equation 3.4)

```
function z = NR(y, TT, DCy)
```

```
%  Net Reliability
%
%  This function computes the Net Reliability of a classifier, based
on a
%  given classification (y) and its degree of certainty for that
(DCy).
%
%  (c) by Zack Voulgaris, Crete 25/8/'06

n1 = length(y); % The number of elements classified
n2 = length(TT);
n3 = length(DCy);
if (n1 ~= n2)|(n1 ~= n3)
    error('Input vectors do not match in length!');
end

v = (y == TT'); % Vector of correctly classified elements
% n = sum(v); % Number of correctly classified elements

z = sum((2*v - 1).*DCy)/n1;
```

Individual Feature Filter (Figure 5.1)

```
function [ITr2 ITe2 NFS] = IFF(P, PT, T, th)

% Feature Selector function 3 (irrelevant to 1 & 2).
% Copyright by Zack Voulgaris, London, March - September 2007

sT = size(T);
[N ind] = max(sT); % Number of elements
[r na] = size(P);

if r ~= N
    error('Dimensionality of Input and Output do not match!')
end

[r c] = size(PT);
if c ~= na
    error('Dimensionality of Input and Output do not match!')
end

if sT(3-ind) ~= 1
    error('Target Training must be a vector')
end

p(na) = 0;
for i = 1:na
    p(i) = IDND5(P(:,i),T);
end

if nargin < 4
    th = harmean(p,2,eps);
end

NFS = find(p>=th);
ITr2 = P(:,NFS);
ITe2 = PT(:,NFS);
```

Note: harmean is a function calculating the harmonic mean of an array p.

Group Feature Selector (Algorithm 5.1, Figure 5.2)

```
function [ITr2 ITe2 NFS] = GFS(P, PT, T)

% Feature Selector function 10 (irrelevant to 1, 2 & 3)
% A variation of Feature Selector 6. Here features are added and
pruned at
% the same time, until the optimum feature set is found.

% Copyright by Zack Voulgaris, London, April-September 2007

sT = size(T);
[N ind] = max(sT); % Number of elements
[r na] = size(P);
if r ~= N
    error('Dimensionality of Input and Output do not match!')
end

[r c] = size(PT);
if c ~= na
    error('Dimensionality of Input and Output do not match!')
end

if sT(3-ind) ~= 1
    error('Target Training must be a vector')
end

[q Q] = nc(T);

% ID0 = IDND5(P, T); % Original ID
% RR = sprintf ('%0.5g', 100*ID0);
% ['Original IDND: ' RR '%']


% Building up starts here

Itemp = P(:,1); FS = 1;
% IDtemp = IDND5(Itemp,T);
AF = 2:na; % Available Features
naf = na - 1; % Number of Available Features
ff = 1;
flag = 1;
tempID = zeros(1,naf);

for i = 1:naf
    temp = [Itemp P(:,AF(i))];
    tempID(i) = IDND5(temp,T,q,Q);
end
[M ind] = max(tempID);
FS = [FS AF(ind)]; ff = ff + 1; naf = naf - 1;
Itemp = [Itemp P(:,AF(ind))];
IDtemp = M;
AF(ind) = [];
clear temp

tempID = zeros(1,ff);
```

```
for i = 1:ff
    temp{i} = Itemp;
    temp{i}(:,i) = [];
    tempID(i) = IDND5(temp{i},T,q,Q);
end
[M ind] = max(tempID);
if M >= IDtemp
    ff = ff - 1;
    naf = naf + 1;
    AF = [AF FS(ind)];
    FS(ind) = [];
    Itemp(:,ind) = [];
    IDtemp = M;
end
clear temp

while flag == 1
    for i = 1:naf
        temp = [Itemp P(:,AF(i))];
        tempID(i) = IDND5(temp,T,q,Q);
    end
    [M ind] = max(tempID);
    if M > IDtemp
        FS = [FS AF(ind)]; ff = ff + 1; naf = naf - 1;
        Itemp = [Itemp P(:,AF(ind))];
        AF(ind) = [];
        IDtemp = M;
    else
        flag = 2;
        break
    end

    for i = 1:naf
        temp = [Itemp P(:,AF(i))];
        tempID(i) = IDND5(temp,T,q,Q);
    end
    [M ind] = max(tempID);
    if M > IDtemp
        FS = [FS AF(ind)]; ff = ff + 1; naf = naf - 1;
        Itemp = [Itemp P(:,AF(ind))];
        AF(ind) = [];
        IDtemp = M;
    else
        flag = 0;
    end
    clear temp

    tempID = zeros(1,ff);
    for i = 1:ff
        temp{i} = Itemp;
        temp{i}(:,i) = [];
        tempID(i) = IDND5(temp{i},T,q,Q);
    end
    [M ind] = max(tempID);
    if M >= IDtemp
        ff = ff - 1;
        naf = naf + 1;
        AF = [AF FS(ind)];
        FS(ind) = [];
        Itemp(:,ind) = [];
        IDtemp = M;
    end
end

if flag == 2
    clear temp
    tempID = zeros(1,ff);
    for i = 1:ff
        temp{i} = Itemp;
        temp{i}(:,i) = [];
        tempID(i) = IDND5(temp{i},T,q,Q);
    end
    [M ind] = max(tempID);
    if M >= IDtemp
        ff = ff - 1;
        naf = naf + 1;
        AF = [AF FS(ind)];
        FS(ind) = [];
        Itemp(:,ind) = [];
        IDtemp = M;
    end
end

NFS = FS;
ITr2 = Itemp;
ITe2 = PT(:,NFS);
% RR = sprintf ('%0.5g', 100*IDtemp);
% ['Reduced Feature Set IDND: ' RR '%']
```

Data (Pattern) Reducer (Figure 5.3)

```
function [P2 T2 A] = PR(P,T,th)

% Pattern Reduction

if nargin < 3
    th = .25;
end

% R = 10000000; % repeating ID calculation parameter
[N na] = size(P);
n = round(N * th);

[T ind] = sort(T);
P = P(ind,:);
Q = [T(1) nan nan nan nan nan nan nan nan nan];
q = 1;
class{20} = []; c(20) = 0; cc(20) = 0;
temp = T;

for i = 1:inf
    ff = (1:N)'.*(temp == Q(q));
    f = ff(ff>0);
    c(q) = length(f); cc(q) = cc(q) + c(q);
    class{q} = [class{q};f];
    temp(f) = NaN;
    if sum(isnan(temp))==N
        break
    end
    q = q + 1;
    Q(q) = temp(cc(q-1)+1);
    cc(q) = cc(q-1) + cc(q);
end

Q = Q(1:q);
a = [1 (cc(1:(q-1))+1)];
A(n,2) = 0; % Matrix showing patterns to be discarded
D(N,N) = 0; % Distance matrix of dataset
ad(q) = 0; % Average distance of class

for i = 1:q
    d = dmatrix3(P(class{i},:));
    ad(i) = sum(sum(d))/(c(i)*(c(i)-1));
    D(a(i):cc(i),a(i):cc(i)) = d / ad(i);
end

[z Z] = IDND5(P,T);
DD(q,N) = 0; % Minimum distance

for i = 1:q
    [M ind] = max(Z(class{i}));
    A(i,1) = ind + a(i) - 1;
    A(i,2) = i;
    Z(A(i,1)) = 0;
    DD(i,:) = D(A(i,1),:);
end

v(q,N) = 0;

for i = (q+1):n
%     p = round(100*i/n);
%     if mod(i,R) == 0
%         clc,disp(p)
%         temp = A((A(:,1)>0),1);
%         P1 = P; T1 = T;
%         P1(temp,:) = repmat(inf,length(temp),na);
%         [z Z] = IDND5(P1,T1);
%         Z(temp) = 0;
%     end

    for j = 1:q
        v(j,:) = DD(j,:).*Z;
    end

    [M r c] = mmax(v); % r = class, c = pattern
    Z(c) = 0;
    A(i,:) = [c r];
    DD(r,:) = min(DD(r,:),D(A(i,1),:));
end

% p = 100;clc,disp(p)

P(A(:,1),:) = []; T(A(:,1)) = [];
P2 = P; T2 = T;
```

Degree of Reliability (Section 3.4)

```
function [DRy elite] = DR(P,T,PT,yy,classifier)

if nargin < 5
    classifier = 'NNKlite';
end

[q, Q, NN, class] = nc(T); % Class information
[N na] = size(P);
n = size(PT,1);

% Create Validation set (V matrix and TV vector)

[ITR ITE TTR TTE] = KFCV(P,T,4);

th(4) = 0;
for k = 1:4

    P1 = ITR{k}; T1 = TTR{k};
    V = ITE{k}; TV = TTE{k};
    N1 = length(T1); m = length(TV);
    for i = 1:q
        temp = (1:m)'.*(TV==Q(i));
        classV{i} = temp(temp>0);
        LC(i) = length(classV{i});
    end

    DRY(m) = 0;
    [y DCy] = feval(classifier,P1,T1,P1);
    DCY = sum(DCy);
    AR = sum(y==T1'); % Self-assessment of classifier
    for i = 1:m
        X = V(i,:);
        CX = (yy(i)==Q);
        P2 = [P1; X];
        T2 = [T1; Q(CX)];
        [y DCy] = feval(classifier,P2,T2,P1);
        d = sum((repmat(X,(N-m),1) - P1).^2,2);
        classi = classV{CX};
        otherclasses = 1:N1;
        otherclasses(classi) = [];
        z1 = sqrt(LC(CX) / (sum(1./(d(classi) + eps)) - eps));
        z2 = sqrt(( N1 - LC(CX) ) / sum(1./(d(otherclasses) + eps) -
eps));
        Z = max( ((z2 - z1) / (z1 + z2)) , 0);
        DRY(i) = ((sum(y==T1')/AR)*sum(DCy)/DCY * Z).^(0.5);
    end

    TH = 1.05; M = 0;
    for i = 1:20
        TH = TH - 0.05;
        temp = (1:m).*(DRY>=TH);
        ELITE = temp(temp>0);
        if isempty(ELITE)
            Z(i) = NaN;
        else
            Z(i) = (sum(y(ELITE)==TV(ELITE)')) / length(ELITE);
            if Z(i) > M
                M = Z(i);
                ind2 = i;
            end
        end
    end

    [M ind1] = max(Z);
    th(k) = 1.05 - 0.025*(ind1 + ind2);
end % of k loop

th = median(th)
DRY(1:n) = 0;
[y DCy] = feval(classifier,P,T,P);
DCY = sum(DCy);
AR = sum(y==T'); % Self-assessment of classifier
for i = 1:n
    X = PT(i,:);
    CX = (yy(i)==Q);
    P2 = [P; X];
    T2 = [T; Q(CX)];
    [y DCy] = feval(classifier,P2,T2,P);
    d = sum((repmat(X,N,1) - P).^2,2);
    classi = class{CX};
    otherclasses = 1:N;
    otherclasses(classi) = [];
    z1 = sqrt(NN(CX) / (sum(1./(d(classi) + eps)) - eps));
    z2 = sqrt(( N - NN(CX) ) / sum(1./(d(otherclasses) + eps) -
eps));
    Z = max( ((z2 - z1) / (z1 + z2)) , 0);
    DRY(i) = ((sum(y==T')/AR)*sum(DCy)/DCY * Z).^(0.5);
end

f = (DRY >= th);
temp = (1:n).*f;
elite = temp(temp>0);
DRy = min(DRY,1);
```

Note: NNKlite is a light version of the classic kNN classifier. This is selected as the default classifier, in case the user doesn’t define a classifier to be used.

Minimum Spanning Tree Classifier (Algorithm 4.3)

```
function [y DCy cpu] = MSTC(P,T,PT)

t = cputime;

[q Q] = nc(T);
[N na] = size(P);
N2 = length(T);
[n na2] = size(PT);

if (N ~= N2)||(na ~= na2)
    error('Dimensionality mismatch!')
end

C{q} = []; TB{q} = []; N(q) = 0; p(q) = 0; Y(n) = 0; DCY(n) = 0;
for c = 1:q
    f = (T == Q(c));
    C{c} = P(f,:); % Class cell
    DM = dmatrix(C{c});
    [temp TB{c}] = MST(DM);
    N(c) = sum(f);
end

cpu = cputime - t;

for i = 1:n
    X = PT(i,:); % Test element to be classified
    for j = 1:q
        D = sum( (ones(N(j),1)*X - C{j}).^2 , 2);
        [d1 ind1] = min(D);

        d1 = sqrt(d1);

        f1 = find(TB{j}(:,1) == ind1);
        f2 = find(TB{j}(:,2) == ind1);
        lf = length(f1) + length(f2);
        TEMP = [C{j}(TB{j}(f1,2) , :); C{j}(TB{j}(f2,1) , :)];
        dtemp = sum( (TEMP - ones(lf,1)*X).^2 , 2);
        [d2 ind2] = min(dtemp);
        d2 = sqrt(d2);
        d3 = sqrt( sum( (TEMP(ind2,:) - C{j}(ind1,:)).^2 , 2 ) );
        clear dtemp
        if d2 >= sqrt(d1^2 + d3^2)
            d = d1; % d = shortest distance to the branch
        else
            t = (d1 + d2 + d3) / 2;
            area = sqrt(t*(t-d1)*(t-d2)*(t-d3));
            d = 2*area / d3;
        end
        p(j) = 1 / (d + eps);
    end
    [M ind] = max(p);
    Y(i) = Q(ind);
    DCY(i) = M / sum(p);
end

y = Y;
DCy = DCY;
```

Minimum Spanning Tree function (auxiliary program for MSTC)

```
function [y C] = MST(P)
%  MST  Minimum Spanning Tree (using Prim Algorithm variation)
%
%  D = distance matrix
%  C = connections matrix for MST
%  y = minimum distance
%
%  Created by Zack Voulgaris, London 29/9/'06

n = size(P,1);

d(n,n) = Inf;
for i = 1:(n-1)
    X = ones((n-i),1)*P(i,:);
    d((i+1):n,i) = sum((X - P((i+1):n,:)).^2,2);
    d(i,i) = Inf;
end

D = d + d';
```

```
CC = [];
yy = 0;
c = 1;

[m f] = min(D(1,:));
CC = [CC;1 f];
yy = yy + m;
c = c + 1;
D(1,f) = Inf; D(f,1) = Inf;
x(1) = 1; x(2) = f;

while c < n
    temp = D(x,:);
    [m1 ind1] = min(temp,[],2);
    [m ind2] = min(m1);
    fx = ind1(ind2);
    fy = x(ind2);

    CC = [CC; fy fx];
    yy = yy + m;
    c = c + 1;
    x(c) = fx;
    for i = 1:(c-1)
        D(x(i),fx) = Inf;
        D(fx,x(i)) = Inf;
    end
end

y = yy;
C = CC;
```

Distance Matrix (auxiliary program for MST)

```
function D = dmatrix(P)

N = size(P,1);

d = zeros(N);
for I = 1:(N-1)
    X = ones((N-I),1)*P(I,:);
    d((i+1):N,i) = sum((X - P((i+1):N,:)).^2,2);
end

D = sqrt(d + d');
```

Feature Subset function (auxiliary program)

```
function [FS IND] = fss(I, k, na)

% FSS – Feature SubSets
%
% Creates k feature subsets of original dataset (I, O) containing na
features.
% The subsets are not overlapping.

IND{k} = []; % Feature Subset
m = floor(na / k); % Number of features in every feature partition
af = 1:na; % Available features
naf = na; % Number of available features
```

```
for j = 1:m
    for I = 1:(k-1)
        a = ceil(naf*rand);
        IND{I} = [IND{I} af(a)];
        af(a) = [];
        naf = naf – 1;
    end
end

IND{k} = af;

for I = 1:k
    IND{I} = sort(IND{I});
    FS{I} = I(:,IND{I});
end
```

Feature Subsets – Balanced (Section 7.2.1)

```
function [FS IND] = fssb(I, O, k, N, na)

% FSSB - Feature SubSets that are Balanced (in terms of
Discernibility)
%
% Creates k feature subsets of original dataset (I, O) containing na
features.
% The subsets are not overlapping.

th = 0.6; NI = 200; % Termination conditions (0.6 = 0.8 x 0.75)

IND{k} = []; % Feature subSet
m = floor(na / k); % Number of features in every feature partition
D(k) = 0;

d = disc(I, O, N);

AF = 1:na;
IND{k} = []; FS = IND;

c = 1;
v = 0; V = 0;
while (v < th)
    c = c + 1;
    af = AF; % Available features
    naf = na; % Number of available features
    ind = IND;
    for i = 1:(k-1)
        for j = 1:m
            a = ceil(naf*rand);
            ind{i} = [ind{i} af(a)];
            af(a) = [];
            naf = naf - 1;
        end
        D(i) = disc(I(:,ind{i}), O, N);
    end
    ind{k} = [IND{k} af];
    D(k) = disc(I(:,ind{k}), O, N);
    v = (min(D))^2 / (d * max(D));
    if v > V
        V = v;
        indB = ind;
    end
    if c == NI
        ind = indB;
        break
    end
end

for i = 1:k
    IND{i} = sort(ind{i});
    FS{i} = I(:,ind{i});
end

return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function z = disc(I, O, N)

ZZ(N) = 0;
for i = 1:N
    D = [sum((repmat(I(i,:),N,1) - I).^2,2) O];
    D(i,:) = [];
    f = (D(:,2) == O(i));
    lf = sum(f);
    r = sum(D(f,1))/lf;
    f2 = (D(:,1) <= r);
    lf2 = sum(f2);
    c = sum(D(f2,2)==O(i));
    ZZ(i) = c / lf2;
end

z = sum(ZZ >= .5)/N;
```

Note: the *disc* function within the *fssb* function is the Spherical Index of Discernibility function.

Feature Subsets – Diverse and Balanced (Section 7.2.1)

```
function [FS IND] = fssdb(P, T, PV, TV, C, N, na)

% FSSDB - Feature SubSets that are Diverse (in terms of errors)
%         and Balanced (in terms of Discernibility)
%
% Creates k feature subsets of original dataset (P, V) containing na
features.
% The subsets are not overlapping and are as diverse as possible.
% C = classifier used
% PV = Validation set Input values
% TV = Validation set Target values

K = 20; % swarm size
NI = 250; % Total number of iterations


p = 3; % number of partitions

n = length(TV);
x(K) = 0; d(K) = 0;
```

```
clear X temp V
combs = p*(p-1)/2;
X{K,p} = []; temp(p) = 0; V(n,p) = 0; temp2(combs) = 0;

for j = 1:K % swarm particle
    af = 1:na; % Available Features
    naf = na;
    for i = 1:p
        r = ceil(naf*rand);
        X{j,i} = [X{j,i} af(r)]; % Partition i
        af(r) = [];
        naf = naf - 1;
    end

    for i = 1:naf
        r = ceil(p*rand);
        X{j,r} = [X{j,r} af(i)];
    end

    for i = 1:p
        X{j,i} = sort(X{j,i});
        temp(i) = disc(P(:,X{j,i}),T,N); % discernibilies of subsets
        y = feval(C,P(:,X{j,i}),T,PV(:,X{j,i}));
        V(:,i) = (y' == TV);
    end

    c = 0;
    for i = 1:(p-1)
        for ii = (i+1):p
            c = c + 1;
            temp2(c) = div(V(:,i),V(:,ii),N);
        end
    end

    x(j) = mean(temp); % average discernibility
    d(j) = mean(temp2); % average diversity
    f = x.*d; % function to be maximised
end

[M ind] = max(f);
stop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function z = disc(I, O, N)

ZZ(N) = 0;
for i = 1:N
    D = [sum((repmat(I(i,:),N,1) - I).^2,2) O];
    D(i,:) = [];
    f = (D(:,2) == O(i));
    lf = sum(f);
    r = sum(D(f,1))/lf;
    f2 = (D(:,1) <= r);
    lf2 = sum(f2);
    c = sum(D(f2,2)==O(i));
    ZZ(i) = c / lf2;
end

z = sum(ZZ >= .5)/N;
```

Note: the *disc* function within the *fssdb* function is the Spherical Index of Discernibility function.

Appendix C – List of Articles Published or Awaiting Publication

Articles Published in Referred Proceedings of International Conferences

- Voulgaris, Z., Magoulas, G., 2008. Extensions of the k nearest neighbour methods for classification problems. *Proceedings of the 26th IASTED International Conference on Artificial Intelligence and Applications*, 23-28.
- Voulgaris Z., Magoulas G. D., 2008, A discernibility-based approach to feature selection for microarray data. *CD Proceedings of the IEEE International Conference of Intelligent Systems*, Varna, Bulgaria, Sept. 2008, IEEE Press.
- Voulgaris, Z., Magoulas, G. D., 2008. Dimensionality reduction for feature and pattern selection in classification problems. *Proceeding of The Third International Multi-Conference on Computing in the Global Information Technology*, Athens, Greece, July 2008, 160-165.
- Voulgaris, Z., Magoulas, G. D., 2008. Discernibility-based approach for creating ensembles in pattern classification applications. *Proceedings of the UKCI Conference*, Leicester U.K., Sept. 2008, 195-199.
- Voulgaris, Z., Mirkin, B., 2008. Optimising a reliability measure for classification. *Proceedings of the UKCI Conference*, Leicester U.K., Sept. 2008, 43-46.

Submitted Journal Articles

- Voulgaris, Z., Mirkin, B., 2008. Choosing a discernibility measure for reject-option at a set of classifiers. *Pattern Recognition Letters*, under revision.
- Voulgaris, Z., Magoulas, G., 2008. Discernibility-based extensions of the k nearest neighbour rule for pattern classification. *Pattern Recognition*, under review.
- Voulgaris, Z., Magoulas, G., 2008. Nearest Neighbour Rules with Self-determined k for Classification. *Pattern Recognition Letters*, under review.