CLASSIFIER COMBINATION METHODS IN PATTERN RECOGNITION

by

Alper BAYKUT

B.S. in I.E., Boğaziçi University, 1992

B.S. in CMPE., Boğaziçi University, 1992

M.S. in I.E., Boğaziçi University, 1994

Submitted to the Institute for Graduate Studies in Science and Engineering in partial fulfillment of the requirements for the degree of

Doctor

of

Philosophy

Boğaziçi University

2002

ACKNOWLEDGEMENTS

I want to express my acknowledgments especially to Prof. Aytül ERÇİL for her valuable guidance, encouragement and understanding throughout the thesis.

I am very grateful to Refik ÜREYEN, and Mustafa TÜZÜNGÜÇ for their encouragement and help in equipment support needed for this research. I would also wish to express my gratitude to my colleagues in the Research and Technology Development Department for their intelligent suggestions to the problems I faced throughout the study.

I wish to express my gratitude to Prof. Gülay BARBARAOSOĞLU, Prof. Yaman BARLAS, Assoc. Prof. Taner BİLGİÇ and Assist. Prof. Berrin YANIKOĞLU for their participation in my thesis committee.

Special thanks to my mother, Cavide BAYKUT, for her understanding and encouragement throughout this study.

ABSTRACT

CLASSIFIER COMBINATION METHODS IN PATTERN RECOGNITION

This thesis studies methodologies to combine multiple classifiers to improve classification accuracy. Different classifiers, training methods and combination algorithms are covered throughout this study. The classifiers are extended to produce class probability estimates besides their class assignments to be able to combine them more efficiently. They are integrated in a framework to provide a toolbox for classifier combination. The leave-one-out training method is used and the results are combined using proposed weighted combination algorithms. The weights of the classifiers for the weighted classifier combination are determined based on the performance of the classifiers on the training phase. The classifiers and combination algorithms are evaluated using classical and proposed performance measures. It is found that the integration of the proposed reliability measure, improves the performance of classification. A sensitivity analysis shows that the proposed polynomial weight assignment applied with probability based combination is robust to choose classifiers for the classifier set and indicates a typical one to three per cent consistent improvement compared to a single best classifier of the same set.

ÖZET

ÖRÜNTÜ TANIMADA SINIFLANDIRICI BİRLEŞTİRME YÖNTEMLERİ

Bu tezde sınıflandırma doğruluğunun arttırılması amacıyla, sınıflandırıcı kararlarının birleştirilmesinin methodları üzerinde çalışılmıştır. Bu çalışmada çeşitli sınıflandırıcılar, eğitim metodları ve birleştirme yöntemleri üzerinde durulmuştur. Sınıflandırıcı birleştirme yöntemlerini daha etkin kılmak için kullanılan tüm sınıflandırıcıların sınıf sonuçlarının yanında olasılık değerleri de üretmeleri için çeşitli değişiklikler yapılmıştır. Tüm sınflandırıcı ve birleştirme yöntemleri bütünleşik bir yazılım yapısı altında gerçeklenmiştir. Her seferinde bir adet verinin dışarıda bırakıldığı eğitim metoduyla eğitilen sınıflandırıcıların sonuçları, önerilen ağırlıklı birleştirme yöntemleri ile birleştirilmiştir. Birleştirme ağırlıkları sınıflandırıcının eğitim seti üzerindeki performansına göre belirlenmiştir. Sınıflandırıcı ve birleştirme yöntemleri, klasik ve önerilen yeni performans ölçütleri ile değerlendirilmiştir. Sınıflandırıcıların güvenilirliklerinin birleştimede kullanılmasının performansı arttırdığı gözlemlenmiştir. Sınıflandırıcı kümesinin, sınıflandırıcılara duyarlılık analizi sonucunda, önerilen çok terimli ağırlık atama metodunun uygulandığı olasılık tabanlı birleştirme yönteminin sınıflandırıcı setindeki sınıflandırıcılara göre oldukça gürbüz olduğu bulunmuştur. Aynı analiz sonucunda sette bulunan en yüksek performanslı sınıflandırıcıdan yüzde bir ila üç arasında daha iyi performans gözlemlenmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

A	Cardinality of the ranked class label set
a	Rank in the set
C	Number of classes
D	Number of features
d	Index for dimension of feature vector
f_n	Feature vector of the n^{th} observation/sample test pattern with D features
f_{nd}	Single feature value of the d^{th} dimension of n^{th} observation/sample test pattern
K	Number of classifiers
k	Index of the classifier
m_k	Classifier-k
N	Number of observations in the training set
n	Index for the observation
p_{knc}	Posterior probability as classification result of classifier m_k for observation x_n for class c
w_c	Classification result as a class label
W_k	Weight of the classifier m_k when combining
w_{kn}	Classification result of classifier m_k for observation x_n as class labels
x_n	n^{th} observation of data set as D dimensional vector
x_{nk}	A subset of D dimensional feature vector used by classifier m_k for n^{th} observation
T	Number of sample test patterns
t	Index for the sample test pattern
Δ_{cnk}	1 if $P(w_c \mid x_{nk}) = \max_{i=1..C} P(w_i \mid x_{nk})$ and 0 otherwise.

1. INTRODUCTION

Broadly speaking, pattern recognition is the science that is concerned with the description or classification/recognition of measurements. Pattern recognition classifies patterns into appropriate classes, depending on their measured features. There is little doubt that pattern recognition is an important, useful, and rapidly developing technology with cross-disciplinary interest and participation.

Pattern recognition techniques are often an important component of intelligent systems, used both for data preprocessing and for decision-making. Pattern recognition is not comprised of one approach, but rather is a broad body of often loosely related knowledge and techniques. Historically, the two major approaches to pattern recognition have been the statistical/decision theoretic and the syntactic/structural approaches. The emerging technology of artificial neural networks has provided a third approach, although there is a relationship between statistical and neural approaches.

No single technology is the optimum solution for all pattern recognition problems, hence we have to incorporate all available and relevant information in a structured fashion to formulate a solution. This includes the development or modification of models that incorporate structural and a priori information. Therefore, the examination of pattern recognition approaches in this thesis will be guided by these questions:

- Can the pattern recognition problem be solved using pattern recognition techniques? If so which pattern recognition techniques suitable, or even applicable to the problem at hand?
- Can we develop or modify useful models for the situation and if necessary, determine the model parameters?
- If so, are there formal tools and heuristics that may be applied, and does a computationally practical solution procedure exist?

The ultimate goal of designing pattern recognition systems is to achieve the best possible classification performance for the task at hand. This objective usually leads to the

development of different classification schemes for the pattern recognition problem to be solved. The results of an experimental assessment of the different designs would then be used as the basis for choosing one of the classifiers as a final solution to the problem. It had been observed in such design studies, that although one of the designs would yield the best performance, the sets of patterns misclassified by the different classifiers would not necessarily overlap. This suggests that different classifier designs potentially offer complementary information about the patterns to be classified, which could be harnessed to improve the performance of the selected classifier.

These observations motivated this study on combining classifiers. The idea is not to rely on a single decision making scheme, but to use all or some subset of the classifiers to derive a consensus decision by combining their individual opinions. This is similar to real-life example of combining expert decisions on a particular problem. In this thesis, we aim to build a robust classifier combination system given a classifier set. For this purpose, the current trends in classifier combination are studied and various classifier combination schemes have been devised. To study classifier combination techniques, 10 classical classifiers are gathered to form a classifier set:

- K-means clustering based classifier,
- Self organizing map clustering based classifier,
- Fuzzy neural network classifier,
- Artificial neural network classifier,
- K-means classifier,
- Parzen classifier,
- K-nearest neighbour classifier,
- Piecewise quadratic distance classifier,
- Piecewise linear distance classifier,
- Support vector machine classifier

In this thesis classifiers and combination techniques are integrated in a framework including a complete software package for pattern recognition is designed and developed. The developed software enables to test the proposed combination schemes providing a

useful pattern recognition toolbox. To analyze the data, some data visualizing techniques are also implemented in this framework. This enables us to see the behavior of the whole training or test data in two dimensions by dimensional reduction of features. Some of the visualization techniques of the framework will be demonstrated on some of the data sets used in the experiments. The characteristics of the data set can also be visualized using the graphical feature versus feature or box plot techniques, which enables a detailed study on the data sets and their class characteristics. The 10 classical classifiers are modified to use them in classifier combination effectively. Different classifier combination schemes are proposed and realized in this framework. All classifiers and combination schemes are evaluated using a variety of performance measures.

When combining different classifiers their decisions may be weighted affecting their influence level on the final decision. Alternatives of weight assignments are proposed. The weights of classifiers are basically based on their performances in the training phase, assuming they will achieve almost the same performance for the test samples. So it is very critical to use objective performance measures while evaluating the classifiers, although there are no agreed objective criteria by which to judge algorithms. Hence different performance measures are also proposed and used to evaluate the classifiers and combination schemes. Any comparative study that does not include the majority of the algorithms would be incomplete. This thesis is aimed to include a wide range of classifiers and combination methods for comparison.

Basic concepts of pattern recognition are detailed in Section 2. In Section 3 an overview of classification techniques and algorithms are given. Section 4 describes the definition and need for combination. Section 5 presents the proposed performance measures and weight assignments for classifier combination and the developed integrated framework. The results of applying the classifier collection with different combination methods on data sets are given in Section 6, where a detailed sensitivity analysis for the classifier sets is done. The final sections cover the summary of findings and the conclusion, and give a design guideline for classifier combination.

2. PATTERN RECOGNITION

The aim of pattern recognition is the design of a classifier m_k, a mechanism which takes d features f_t of sample test pattern x_n as its input and which results in a class wc_{kt} to indicate which class the pattern belongs to. Pattern recognition techniques overlap with other areas, such as: signal processing and systems, artificial intelligence, neural modeling, optimization/estimation theory, automata theory, fuzzy sets, structural modeling, formal languages.

Pattern recognition applications include: image processing, image segmentation and analysis, seismic analysis, radar signal classification/analysis, face recognition, speech recognition/understanding, fingerprint identification, character, letter or number recognition, handwriting analysis, electrocardiographic signal analysis/understanding, medical diagnosis, etc....

Pattern recognition applications come in many forms. In some instances, there is an underlying and quantifiable statistical basis for the generation of patterns. In other instances, the underlying structure of the pattern provides the information fundamental for pattern recognition. In still others, neither of the above cases hold true, but we are able to develop and train an architecture to correctly associate input patterns with desired responses. Thus, a given problem may allow one or more of these different solution approaches.

For different applications we may have different feature sets, different training sets, different classification methods or different training sessions, all resulting in a set of classifiers whose outputs may be combined, with the hope of improving the overall classification accuracy. If this set of classifiers is fixed, the problem focuses on the combination function. It is also possible to use a fixed combiner and optimize the set of input classifiers. In this thesis the set of classifiers and their parameters are fixed; they are not optimized for the application at hand, different combination schemes are developed hoping to reach a reasonable classification accuracy that is independent of the characteristic of the application.

There are many books on pattern recognition, however, the coverage is usually partitioned along statistical, syntactic, and neural pattern recognition. Statistical pattern recognition is explored in depth in books by (Duda and Hart, 1973), (Fukunaga, 1972), (Jain and Dubes, 1988). Coverage of syntactic pattern recognition approaches are presented in (Fu, 1982), (Pavlidis, 1977). Fundamental neural network architecture and application book references are (Kohonen, 1984), (Bishop, 1995). Various aspects of pattern recognition are covered also in journals: Pattern Recognition letters journal, IEEE Transactions on Systems, Man and Cybemetics, IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Transactions on Acoustics, Speech and Signal Processing, IEEE Transactions on Information Theory, IEEE Transactions on Geoscience and Remote Sensing, IEEE Transactions on Industrial Electronics, IEEE Transactions on Neural Networks, Optical Engineering (SPIE).

2.1. General Pattern Recognition Systems

The boundaries between statistical pattern recognition, syntactic pattern recognition, and neural pattern recognition are fuzzy and fading (Schalkoff, 1990b). They share common features and common goals. Often, given a specific pattern recognition problem, we choose one approach over another based on analysis of underlying statistical components (statistical pattern recognition), underlying grammatical structure (syntactic pattern recognition), as well as suitability of a neural network solution and training ability, and perhaps lack of suitable statistical or structural models.

In realistic applications, the design of an autonomous pattern recognition system is a complex, usually iterative and interactive task. Although it is impossible to provide an all-inclusive algorithmic procedure, the highly interrelated, skeletal steps outlined here reflect typical efforts and concerns.

- Study the classes of patterns under consideration to develop possible characterizations. This includes assessments of (quantifiable) pattern structure and probabilistic characterizations, as well as exploration of possible within-class and interclass similarity/dissimilarity measures. In addition, possible pattern

deformations or invariant properties and characterization of noise sources should be considered at this point.

- Determine the availability of feature/measurement data.
- Consider constraints on desired system performance and computational resources (e.g., parts/minute, classification accuracy).
- Consider the availability of training data.
- Consider the availability of suitable and known pattern recognition techniques (e.g., statistical pattern recognition, syntactic pattern recognition, and clustering) an overall pattern recognition system structure.
- Develop a pattern recognition system simulation. This may involve choosing models, grammars, or network structures.
- Train the system.
- Simulate system performance.
- Iterate among the above steps until desired performance is achieved.

This study will cover the known techniques of pattern recognition for the different approaches, and combination methods will be applied to develop an automated hybrid system for pattern recognition.

2.2. Hybrid Systems for Pattern Recognition

Much of statistical, syntactic and neural pattern recognition are based on the concept of pattern similarity. Universally applicable similarity measures that enable good classification are both desirable and elusive. Hybrid systems try to combine statistical, syntactical and neural approaches to increase the performance of classification or to come out a robust classification performance.

The aim of pattern recognition is the design of a classifier m_k, a mechanism which takes d features f_t of sample test pattern x_n as its input and which results in a class w_{kn} to indicate which class the pattern belongs to. The hybrid system tries to combine the results of different statistical, syntactical and neural approaches in a classifier set to increase the performance of classification. In fact these results can be more informative than the single

class assignment decision, when for example the posterior probabilities of the decision are also given. The results may also include the posterior probabilities for all classes for the sample. Especially for the aim of classifier combination, results should also be generated in one of the following forms to indicate to which class the pattern belongs.

- A single class label with the highest posterior probability may be assigned to the sample pattern $w_{kn}=w_c$ where w_{kn} is an integer class label value 1...C and $P(w_c \mid x_n, m_k) = \max_{i=1..C} P(w_i \mid x_n, m_k)$
- A ranked class label set based on the posterior probabilities with a predefined cardinality A may be used $w_{kn}=\{w_a\}=\{w_1,\ldots,w_A\}$, where $P(w_a \mid x_n, m_k) \geq P(w_{(a+1)} \mid x_n, m_k)$ and $A \leq C$. If we set the cardinality A=1, we result with the first form of single class labels.
- A pair of class label with its posterior probability may be given as result of classification $p_{knc}=(w_c, P(w_c \mid x_n, m_k))$, where $P(w_c \mid x_n, m_k) = \max_{i=1..C} P(w_i \mid x_n, m_k)$ is the maximum posterior probability for a class assignment w_{kn}, given the sample test pattern x_n and classifier m_k.
- A set of pairs of class labels with their respective posterior probabilities for all possible classes may be given as result $p_{kni}=\{w_i, P(w_i \mid x_n, m_k)\}$, where i=1...C.

The ultimate goal of designing pattern recognition systems is to achieve the best possible classification performance for the task at hand. This objective traditionally led to the development of different classification schemes for the pattern recognition problem to be solved. The results of an experimental assessment of the different designs would then be the basis for choosing one of the classifiers as a final solution to the problem. It had been observed in such design studies, that although one of the designs would yield the best performance, the sets of patterns misclassified by the different classifiers would not necessarily overlap. This suggested that different classifier designs potentially offered complementary information about the patterns to be classified, which could be harnessed to improve the performance of the selected classifier.

These observations motivated the relatively recent interest in combining classifiers. The idea is not to rely on a single decision making scheme. Instead, all the designs, or their subset, are used for decision making by combining their individual opinions to derive a consensus decision. Various classifier combination schemes have been devised and experimentally demonstrated that some of them consistently outperform a single best classifier. There are several reasons for combining multiple classifiers to solve a given classification problem. Some of them are listed below:

- A designer may have access to a number of different classifiers, each developed in a different context and for an entirely different representation/description of the same problem.
- Sometimes more than a single training set is available, each collected at a different time or in a different environment. These training sets may even use different features.
- Different classifiers trained on the same data may not only differ in their global performances, but they also may show strong local differences. Each may have its own region in the feature space in which it performs the best.
- Some classifiers such as neural networks show different results with different initializations due to the randomness inherent in the training procedure. Instead of selecting the best network and discarding the others, one can combine various networks, thereby taking advantage of all the attempts to learn from the data.

In summary, we may have different feature sets, different training sets, different classification methods or different training sessions, all resulting in a set of classifiers whose outputs may be combined, with the hope of improving the overall classification accuracy.

3. CLASSIFIERS

A classifier partitions the feature space into class labeled decisions regions. In order to use decision regions for a possible and unique class assignment, these regions must cover the feature space and be nonoverlapping. An exception to the last constraint is the notion of fuzzy sets (Zadeh, 1965), where decision regions may overlap. The border of each decision region is a decision boundary/discriminant function.

There are many different ways to represent pattern classifiers. One way is in terms of a set of discriminant functions $g_c(x)$, c=1,...,C. The classifier assigns a sample x_n using discriminant functions as follows:

$$\text{Assign } x_n \rightarrow w_c \text{ if } g_c(x_n) > g_i(x_n) \text{ for all } i \neq c. \tag{3.1}$$

The problem of finding a linear discriminant function will be formulated as a problem of minimizing a criterion function. The obvious criterion function for classification purposes is the sample risk, or training error, the average loss incurred in classifying the set of training samples (Duda and Hart, 1973). It is difficult to derive the minimum risk linear discriminant hence several related criterion functions that are analytically more tractable are also used.

To use discriminant functions as classifiers we assume that we know the proper forms for the discriminant functions and use the samples to estimate the values of parameters of the classifier. There are various procedures for determining discriminant functions, some of which are statistical and some of which are not. None of them, however, requires knowledge of the forms of underlying probability distributions, and in this limited sense they can be said to be nonparametric.

In case of linear discriminant functions, they are either linear in the components of x, or linear in some given set of functions of x. Linear discriminant functions have a variety of analytical properties.

A discriminant function is a linear combination of components of x, that can be written as $g(x)=\omega^t x+\omega_0$, where ω is the weight vector and ω_0 is the bias or threshold weight. For the general case there will be c such discriminant functions, one for each of c classes. For a discriminant function of this form, a two class classifier implements the following decision rule: Assign $x_n \rightarrow w_1$ if $g(x_n)>0$ and w_2 if $g(x_n)<0$. In other words x_n is assigned to w_1 if the inner product $\omega^t x_n$ exceeds the threshold $-\omega_0$ and to w_2 otherwise. If $g(x_n)=0$, x_n can be ordinarily assigned to either class (Duda and Hart, 1973).

In all two-class cases, such discriminants lead to hyperplane decision boundaries, either in the feature space itself or in a space where the features have been mapped by a nonlinear function. In support vector machines, discussed in Section 3.7, the input is mapped by a nonlinear function to a highdimensional space, and the optimal hyperplane is the one that has the largest margin. The support vectors are those transformed patterns that determine the margin. They are informally the hardest patterns to classify, and the most informative ones for designing the classifier. An upper bound on expected error rate of the classifier depends linearly upon the expected number of support vectors.

For multiclass problems, the linear machines create decision boundaries consisting of sections of such hyperplanes. One can prove convergence of multiclass algorithms by first converting them to two class algorithms and using the two class proofs. The simplex algorithm finds the optimum of a linear function subject to constraints, and it can be used for training linear classifiers.

There is more than one way to devise multiclass classifiers employing linear discriminant functions. The problem may be reduced to c-1 two class problems, where the i^{th} problem is solved by a linear discriminant function that separates points assigned to w_i from those not assigned to w_i. A more costly approach would be to use $\frac{c(c-1)}{2}$ linear discriminants, one for every pair of classes. Both of these approaches can lead to regions in which the classification is undefined (Hand, 1981).

For multiclass case linear discriminant functions are defined as $g_i(x)=\omega^t x_i+\omega_{i0}$ where i=1..c and the decision rule becomes: Assign $x_n \rightarrow w_i$ if $g_i(x_n)>g_j(x_n)$ for all $j \neq i$, and in case

of ties, the classification is left undefined. This resulting classifier is called a linear machine. A linear machine divides the feature space into c decision regions, with $g_i(x)$ being the largest discriminant if x is in region R_i. If R_i and R_j are contiguous, the boundary between them is a portion of the hyperplane H_{ij} defined by $g_i(x)=g_j(x)$ or $(\omega_i-\omega_j)^t x+(\omega_{i0}-\omega_{j0})=0$.

It follows at once that $(\omega_i-\omega_j)$ is normal to H_{ij} and the signed distance from x to H_{ij} is given by $\frac{g_i(x)-g_j(x)}{\|\omega_i-\omega_j\|}$. Thus, with the linear machine it is not the weight vectors themselves but their differences that are important. While there are $\frac{c(c-1)}{2}$ pairs of regions, they need not all be contiguous, and the total number of hyperplane segments appearing in the decision surfaces is often fewer than $\frac{c(c-1)}{2}$. It is easy to show that the decision regions for a linear machine are convex, and this restriction surely limits the flexibility and accuracy of the classifier. In particular, every decision region is singly connected and this tends to make the linear machine most suitable for problems for which the conditional densities $p(x|w_i)$ are unimodal.

The linear discriminant function g(x) can be written as $g(x)=\omega_0+\sum_{i=1}^{d}\omega_i x_i$ where the coefficients ω_i are the components of the weight vector ω. By adding additional terms involving the products of pairs of components of x, the quadratic discriminant function is obtained: $g(x)=\omega_0+\sum_{i=1}^{d}\omega_i x_i+\sum_{i=1}^{d}\sum_{j=1}^{d}\omega_{ij}x_i x_j$. This procedure can be generalized obtaining the class of polynomial discriminant functions. Linear discriminants, while useful, are not sufficiently general for arbitrary challenging pattern recognition problems unless an appropriate nonlinear mapping can be found.

Clearly, the choice of discriminant functions is not unique. The discriminant functions can be multiplied by a positive constant or biased by an additive constant without influencing the decision. More generally, if g(x) is replaced by f(g(x)), where f is a monotonically increasing function the resulting classification is unchanged (Duda and Hart,

1973). This observation can lead to significant analytical and computational simplifications.

$$g_i(x) = P(\omega_i \mid x)$$
$$g_i(x) = \frac{P(x \mid \omega_i)P(\omega_i)}{\sum_{j=1}^{C} P(x \mid \omega_j)P(\omega_j)} \qquad (3.2)$$
$$g_i(x) = P(x \mid \omega_i)P(\omega_i)$$

3.1. Bayes Classifier

Bayes decision theory is a fundamental statistical approach to the problem of pattern recognition. This approach is based on the assumption that the decision problem is posed in probabilistic terms, and that all of the relevant probability values are known. The Bayes classifier is a mechanism which minimizes the classification error. In order to do this, it needs the underlying probability density functions of the features. The Bayes classifier labels an object with the label for which the probability density function multiplied by the a priori probability is highest. The minimal error this classifier makes, the Bayes error, therefore is a theoretical minimum to the error any classifier can make (Duda and Hart, 1973). The Bayes rule is given by:

$$P(w_c \mid x_n) = \frac{P(x_n \mid w_c)P(w_c)}{P(x_n)} \text{ where } P(x_n) = \sum_{c=1}^{C} P(x_n \mid w_c)P(w_c)$$

In order to implement the ideal Bayes classifier, one needs the knowledge of the exact distributions of the classes. In practical problems an infinite number of learning samples are never available. For that reason, the true probability density functions and a priori probability information is not available. Classifier construction is then based on the assumption that the learning samples available represent the true probability density function, i.e. that they are realizations of stochastic variables having the true distribution function. In this case we either assume the parametric form of the distribution and estimate the parameters or estimate the distribution nonparametrically. The Bayesian approach assumes that the unknown parameter vector is a random variable. Any information about

w_c prior to observing the samples is assumed to be contained in a known apriori density $P(w_c)$. Observation of the samples converts this to an aposteriori density $P(w_c|x_n)$, which, hopefully, is sharply peaked about the true value of w_c. Therefore, classifiers will be built that minimize the Bayes error on the training set in the hope that this will also minimize the error made on test set. However, in practice we may not know the true form of the underlying probability density, hence we are going to estimate the distributions nonparametrically from a finite number of samples available. There are several types of nonparametric methods of interest in pattern recognition:

- Estimating the density functions $P(x_n|w_c)$, which can be substituted for the true densities when designing the classifier.
- Estimating the posterior probabilities $P(w_c|x_n)$, which bypass probability estimation and used directly in decision functions.

The most fundamental techniques of estimating an unknown probability density function rely on the fact that the probability P that a vector x will fall in a region R is given by $P=\int_R p(x`)dx`$. Thus P is a smoothed or averaged version of the density function p(x), and this smoothed value of p can be estimated by estimating the probability P. Suppose that n observations $x_1,\ldots,x_n$ are made independently and identically according to the density of p(x). Clearly, the probability that k of these n are in R is given by the binomial law: $P_k = \binom{n}{k} P^k (1-P)^{n-k}$, and the expected value for k is E(k)=nP.

For large n the ratio k/n is a very good estimate for the probability P. If we now assume that p(x) is continuous and that the region R with enclosed volume V is so small that p does not vary much within it, we can write $P=\int_R p(x`)dx`=p(x)V$ and reach to the estimate of p(x) as (k/n)/V.

From a practical standpoint, the number of samples is always limited. Thus, the volume V cannot be allowed to become arbitrarily small. If this kind of estimate is to be used, one will have to accept a certain amount of variance in the ratio k/n and a certain amount of averaging of the density p(x). To estimate the density at x, we form a sequence

of regions $R_1,R_2,\ldots$ containing x, where R_n is a region with n sample. Let V_n be the volume of region R_n, k_n be the number of samples in R_n, and $p_n(x)=(k_n/n)/V_n$ be the n^{th} estimate for p(x). There are two leading methods for estimating the density at a point:

- The first method is to specify k_n as some function of n, such as $k_n=\sqrt{n}$ and so the volume V_n is grown until it encloses k_n neighbours of x. This is the k-nearest neighbour estimation method.
- The second method is to start with a large volume centered on the test point and shrink it according to a function such as $V_n=1/\sqrt{n}$. This is the Parzen window method.

3.2. K-Nearest Neighbour

If a number of classes are represented by example points in a feature vector space and it is required to classify an unknown vector x_n, this can be done by finding the closest point to it and assigning the closest point's class label to x_n. This would be called the nearest neighbour classification technique.

$$\text{Assign } x_n \rightarrow w_c \text{ if } d(x_n,(x_t,w_c)) = \min_{i=1..N} \left[d(x_n,(x_i,w_j)) \right] \tag{3.3}$$

The k-nearest neighbour classifier is similar, except that the K nearest points to x_n are found, and it is assigned to the class represented by the class with the largest number of the neighbouring points. The k-nearest neighbour classifier labels an unknown pattern x_n with the label of the majority of the K nearest neighbours. Both methods produce nonlinear decision surfaces.

If we calculate, sort and build the set of K smallest distances $\{d_1(x_n,(x_{(1)},w_1)),\ldots, d_K(x_n,(x_{(K)},w_K))\}$ where $x_{(i)}$ is the i^{th} training sample sorted by increasing distance to the test sample , $w_{(i)}$ its class label and define $\Delta_{ck}=1$ if $d_k(x_n,(x_k,w_c))$ is in the set and 0 otherwise.

$$\text{Assign } x_n \rightarrow w_c \text{ if } \sum_{k=1}^{K} \Delta_{ck} = \max_{i=1..C} \sum_{k=1}^{K} \Delta_{ik} \tag{3.4}$$

The k-nearest neighbour classifier is critically dependent on the method of distance measure in the feature space. Euclidean Distance $EucD(x_i, x_j) = \sqrt{\sum_{d=1}^{D}(x_{id} - x_{jd})^2}$ is the most widely used distance formula for that purpose. If the feature space is transformed or scaled, the Euclidean distance measure in the transformed space can be very different from the original distance relationships, even though the transformation merely amounts to a different choice of units for the features. Such scale changes can have major impact on k-nearest neighbour classifier.

Commonly used distance measures (Wilson and Martinez, 1997) can be summarized as follows:

Euclidean Distance $$EucD(x_i, x_j) = \sqrt{\sum_{d=1}^{D}(x_{id} - x_{jd})^2}$$

Mahalanobis Distance $$MahD(x_i, x_j) = (x_i - x_j)^T \sum^{-1} (x_i - x_j)$$

where Σ^{-1} is the covariance matrix of the training set.

Minkowsky Distance $$L_zD(x_i, x_j) = \left[\sum_{d=1}^{D} | x_{id} - x_{jd} |^z\right]^{1/z}$$

(L_1 is Manhattan Distance, L_2 is Euclidean Distance)

Chebychev Distance $$ChebvD(x_i, x_j) = \max_{d=1..D} | x_{id} - x_{jd} |$$

Camberra Distance $$CamD(x_i, x_j) = \sum_{d=1}^{D}\left[\frac{| x_{id} - x_{jd} |}{| x_{id} + x_{jd} |}\right]$$

The discrimination function implemented by this classifier will in general be a jagged, piecewise linear function since it is influenced by each observation available in the training set. A disadvantage of this method is its large computing power requirement, since for classifying an object its distance to all the objects in the learning set has to be calculated.

One modification to the typical k-nearest neighbour classifier is performed with the addition of conflict resolution strategy. The model consists of two stages. In the first stage, if we find that a given class has the majority of training samples closer to the test pattern, then we declare this class as an outright winner and allocate the test pattern to this class. However, if we find that there are an equal number of nearest neighbours for two or more classes surrounding the test pattern, and then we perform the second stage called conflict resolution. At this stage, the class whose distance from the test data, averaged over all its training samples within the hypersphere, is found to be the smallest is declared the winner.

If we calculate, sort and build the set of K smallest distances $\{d_1(x_n,(x_{(1)},w_{(1)})),\ldots, d_K(x_n,(x_{(K)},w_{(K)}))\}$ where $x_{(i)}$ is the i^{th} training sample sorted by increasing distance to the test sample, $w_{(i)}$ its class label and define $\Delta_{ck}=1$ if $d_k(x_n,(x_{(i)},w_c))$ is in the set and 0 otherwise. Define $\Delta_k=1$ if $\sum_{k=1}^{K}\Delta_{ck}=\max_{i=1..C}\sum_{k=1}^{K}\Delta_{ik}$ and 0 otherwise.

$$\text{Assign } x_n\rightarrow w_c \text{ if } \sum_{k=1}^{K}\left[\Delta_k d_k(x_t,(x_{(i)},w_c))\right]=\min_{j=1..C,\Delta k>0}\sum_{k=1}^{K}\left[\Delta_k d_k(x_t,(x_{(i)},w_j))\right] \qquad (3.5)$$

Average distance k-nearest neighbour classifier system ignores the quantity of nearest neighbours found for the given test pattern but considers the average distance between classes and the test pattern to decide by selecting the class to which it's average distance is the smallest. The test pattern is allocated to the winning class. These average distances are based on the distance from the test data of all training patterns found within the hypesphere. Define $\Delta_{ij}=1$ if $d_i(x_n,(x_i,w_c))$ and j=c, and 0 otherwise.

$$\text{Assign } x_n\rightarrow w_c \text{ if } \sum_{i=1}^{N}\left[\frac{\Delta_{ic}d_i(x_n,(x_i,w_c))}{\sum_{k=1}^{N}\Delta_{kc}}\right]=\min_{j=1..C}\sum_{i=1}^{N}\left[\frac{\Delta_{ij}d_i(x_i,w_j))}{\sum_{k=1}^{N}\Delta_{kj}}\right] \qquad (3.6)$$

To increase performance of classification, after a test pattern has been classified, the classification result is verified using the test patterns' true class. If the classification assignment is correct then the pattern is added to the training set. For subsequent classifications, the incremented training data will be used. The results have shown that such updates of the training set play a crucial role when the training data is small in amount at the start of the experiment (Singh *et al.*, 1999). The incremental learning process also allows the classifier to improve its learning abilities with time and improves its confidence on unseen data labeling.

3.3. Parzen Window Approach

One convenient method to estimate the probability density function of a random variable from a set of given samples is Parzen window approach. The Parzen window approach assumes that the region R_n is a d-dimensional hypercube. If h_n is the length of an edge of that hypercube, then its volume is given by $V_n=h_n^d$. We can obtain an analytic expression for k_n, the number of samples falling in the hypercube, by defining the following window function: $\varphi(u)=1$ if $|u_i|=1/2$; $i=1,...,d$ and 0 otherwise. Thus, $\varphi(u)$ defines a unit hypercube centered at the origin. It follows that $\varphi(\frac{x-x_i}{h_n})$ is equal to unity if x_i falls within the hypercube of volume V_n centered at x, and is zero otherwise. The number of samples in this hypercube is therefore given by $k_n = \sum_{i=1}^{n} \varphi(\frac{x-x_i}{h_n})$ (Duda and Hart, 1973). Thus we obtain the estimate $p_n(x) = \frac{k_n/n}{V_n} = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{V_n} \varphi(\frac{x-x_i}{h_n})$. This equation suggests a more general approach to estimating the density functions.

Rather than limiting the window function to a hypercube other kernels can be substituted. Parzen window approach is mostly used with Gaussian kernels $\varphi(u) = \frac{1}{\sqrt{2\pi}} e^{\frac{-u^2}{2}}$. Hence, given a set of observations x_n, the estimate of the probability density function of a specific class c becomes $p_c(x_n) = \frac{1}{n_c}\sum_{i=1}^{nc} \frac{1}{h_{nc}} \varphi(\frac{x_n - x_{ci}}{h_{nc}})$ with

$h_{nc} = \frac{h_1}{\sqrt{n_c}}$ or $h_{nc} = \frac{h_1}{\ln(n_c)}$. Using the prior probabilities $P(w_c)$, the Bayes classifier can be formulated using nonparametric probability density function estimates as

$$\text{Assign } x_n \rightarrow w_c \text{ if } P(w_c)p_c(x_n) = \max_{i=1..C}(P(w_i)p_i(x_n)) \tag{3.7}$$

3.4. Fisher's Linear Discriminant Function

The Fisher approach (Fisher, 1936), is based on projection of D-dimensional data onto a line. For any data set, the dimensionality can be reduced from D dimensions to one dimension, if the D-dimensional data is projected onto a line. Of course, even if the samples formed well-separated, compact clusters in D-space, projection onto an arbitrary line will usually produce a confused mixture of samples from all the classes and thus produce poor classification performance. However, by moving the line around, it might be possible to find an orientation for which the projected samples are well separated. This is exactly the goal of discriminant analysis.

Suppose that we have a set of n D-dimensional samples $x_1,..,x_n$ If we form a linear combination of the components of x, we obtain $y=A^tx$ and a corresponding set of n samples $y_1,\ldots,y_n$ Geometrically, if $\|A\|=1$, each y_i is the projection of the corresponding x_i onto a line in the direction of A. Actually, the magnitude of A is of no real significance since it merely scales y. The direction of A is important, however, if we imagine that the samples labeled w_1 fall more or less into one cluster while those labeled w_2 fall in another, we want the projections falling onto the line to be well separated, not thoroughly intermingled (Duda and Hart, 1973).

It is tried to find the best such direction A to be used in the classification. A measure of the separation between the projected points is the difference of the sample means. If m_c is the D-dimensional sample mean given by $m_c = \frac{1}{n_c}\sum_{i=1}^{n_c} x_n$.

Then the sample mean for the projected points is given by $m_c' = \frac{1}{n_c}\sum_{i=1}^{n_c} y_i = \frac{1}{n_c}\sum_{i=1}^{n_c} A_i^t x_i = A^t m_c$ and is simply the projection of m_c.

It follows that for the two class case, the distance between the projected means is $|m_1' - m_2'| = |A^t(m_1 - m_2)|$ and that we can make this difference as large as we wish merely by scaling A. To obtain good separation of the projected data, we really want the difference between the means to be large relative to some measure of the standard deviations for each class. We define the scatter for projected samples labeled w_i by $S_c^2 = \sum_{i=1}^{n_c}(y - m_i')^2$.

Thus, $\frac{1}{n}(S_1^2 + S_2^2)$ is an estimate of the variance of the pooled data, and $(S_1^2 + S_2^2)$ is called the total within-class scatter of the projected samples. The Fisher linear discriminant employs that linear function $A^t x$ for which the criterion function $J(A) = \frac{|m_1' - m_2'|^2}{S_1^2 + S_2^2}$ is maximum. It can be shown that J(A)can be written as: $J(A) = \frac{A^t S_B A}{A^t S_W A}$ where $S_B=(m_1-m_2)(m_1-m_2)^t$ and $S_w=S_1+S_2$ with $S_c = \sum_{i=1}^{n_c}(x - m_c)(x - m_c)^t$.

This expression is well known in mathematical physics as the generalized Rayleigh quotient and it can be shown that a vector w that maximizes J(A) must satisfy $S_BA=\lambda^t S_WA$ and the solution for the A that optimizes J(A) is $A=S_W^{-1}(m_1-m_2)$.

Thus the classification has been converted from a D-dimensional problem to a hopefully more manageable one-dimensional one. This mapping is many-to-one, and in theory it cannot possibly reduce the minimum achievable error rate, if we have a very large training set.

For the C-class problem, the natural generalization of Fisher's linear discriminant involves C-1 discriminant functions. Thus, the projection is from a C-dimensional space to a (C-1) dimensional space. The generalization of the above formulas result in finding a rectangular matrix A that maximizes:

$$J(A) = \frac{\left|A^t S_B A\right|}{\left|A^t S_W A\right|} \text{ where } S_B = \sum_{i=1}^{C} n_i (m_i - m)(m_i - m)^t \text{ , } S_W = \sum_{i=1}^{C} S_i \text{ and } m = \frac{1}{n}\sum_{i=1}^{C} n_i m_i \text{ .}$$

3.5. Artificial Neural Networks

Many neural network models are similar or identical to popular statistical techniques such as generalized linear models, polynomial regression, nonparametric regression and discriminant analysis, projection pursuit regression, principal components, and cluster analysis, especially where the emphasis is on prediction of complicated phenomena rather than on explanation (Warren, 1994). There are also a few neural network models, such as counterpropagation, learning vector quantization, and self-organizing maps that have no precise statistical equivalent for data analysis.

A brief historical review by Jain and Mao (Jain and Mao, 1996) for neural network research has noted three periods of extensive activity. The first peak in the 1940s was due to McCulloch and Pitts pioneering work (McCulloch and Pitts, 1943). The second occurred in the 1960s with Rosenblatt's perceptron convergence theorem (Rosenblatt, 1962) and Minsky and Papert's work showing the limitations of a simple perceptron (Minsky and Papert, 1969). Minsky and Papert's results dampened the enthusiasm of most researchers, especially those in the computer science community. The resulting hull in neural network research lasted almost 20 years. Since the early 1980s, neural networks have received considerable renewed interest. The major developments behind this resurgence include Hopfield's energy approach in 1982 (Hopfield, 1982) and the backpropagation learning algorithm for multilayer feedforward networks first proposed by Werbos (Werbos, 1974), reinvented several times, and then popularized by Rumelhart and McClelland in 1986 (Rumelhartand and McClelland, 1986). Anderson and Rosenfeld

(Anderson and Rosenfeld, 1988) provide a detailed historical account of neural network developments.

Neural networks can be viewed as weighted directed graphs in which artificial neurons are nodes and directed edges with weights are connections between neuron outputs and neuron inputs. Neural networks provide an emerging paradigm for pattern recognition implementations that involve large interconnected networks of relatively simple and typically nonlinear units so-called neurons. Although a number of artificial neural structures exist, and more continue to appear as research continues, many of these structures have common topological properties, unit characteristics, and training approaches. Basically, three entities characterize a neural network:

- The network topology, or interconnection of neural units;
- The characteristics of individual units or artificial neurons; and
- The strategy for pattern learning or training.

Based on the connection pattern, neural networks can be grouped into two categories:

- feed-forward networks, in which graphs have no loops,
- recurrent or feedback networks, in which loops occur because of feedback connections.

The neuron computes a linear combination of the inputs possibly with an intercept or bias term, then a possibly nonlinear activation function is applied to produce the output. An activation function maps any real input into a usually bounded range, often 0 to 1 or -1 to 1. Some common activation functions are: (Warren, 1994)

- Linear or identity: $f(z)=z$
- Hyperbolic tangent: $f(z)=\tanh(z)$
- Logistic: $f(z)=(1+e^{-x})^{-1}=(\tanh(z/2)+1)/2$
- Sigmoid: $f(z)=(1+e^{-\beta x})^{-1}$
- Threshold: $f(z)=0$ if $z<0$, 1 otherwise

- Gaussian: $f(z)=e^{-z2/2}$

A neuron can have one or more outputs. Each output has a separate bias and set of weights. Usually the same activation function is used for each output, although it is possible to use different activation functions.

Multi layer perceptrons are general purpose, flexible, nonlinear models that, given enough hidden neurons and enough data, can approximate virtually any function to any desired degree of accuracy. In other words, multi layer perceptrons are universal approximators (White, 1992). With a small number of hidden neurons a multi layer perceptron is a parametric model that provides a useful alternative to polynomial regression. With a moderate number of hidden neurons, a multi layer perceptron can be considered a quasi-parametric model similar to projection pursuit regression (Friedman and Stuetzle, 1981). A multi layer perceptron with one hidden layer is essentially the same as the projection pursuit regression model except that a multi layer perceptron uses a predetermined functional form for the activation function in the hidden layer, whereas projection pursuit uses a flexible nonlinear smoother. If the number of hidden neurons is allowed to increase with the sample size, a multi layer perceptron becomes a nonparametric sieve that provides a useful alternative to methods such as kernel regression and smoothing splines (Haerdle, 1990). Multi layer perceptrons are especially valuable because you can vary the complexity of the model from a simple parametric model to a highly flexible nonparametric model.

Feature mapping is a form of nonlinear dimensionality reduction that has no statistical analog. There are several varieties of feature mapping, of which Kohonen's (Kohonen, 1989) self-organizing map is the best known. Methods such as principal components and multidimensional scaling can be used to map from a continuous highdimensional space to a continuous lowdimensional space. Self-organizing map maps from a continuous space to a discrete space. Self-organizing map is a special type of competitive learning network that defines a spatial neighbourhood for each output unit. The shape of the local neighbourhood can be square, rectangular, or circular. Initial neighbourhood size is often set to one half to two thirds of the network size and shrinks

over time according to a function. During competitive learning, all the weight vectors associated with the winner and its neighbouring units are updated.

Learning vector quantization (Kohonen, 1989) has both supervised and unsupervised aspects, although it is not a hybrid network in the strict sense of having separate supervised and unsupervised layers. Learning vector quantization is a variation of nearest neighbour discriminant analysis. Rather than finding the nearest neighbour in the entire training set to classify an input vector, learning vector quantization finds the nearest point in a set of prototype vectors, with several prototypes for each class. Learning vector quantization differs from edited and condensed k-nearest neighbour methods (Hand, 1981) in that the prototypes are not members of the training set but are computed using algorithms similar to adaptive vector quantization. A somewhat similar method proceeds by clustering each class separately and then using the cluster centers as prototypes. The clustering approach is better if you want to estimate posterior membership probabilities, but learning vector quantization may be more effective if the goal is simply classification (Oehler and Gray, 1995).

3.6. Clustering Based Classifiers

Clustering is the unsupervised classification of patterns into clusters based on similarity. Intuitively, patterns within a valid cluster are more similar to each other than they are to a pattern belonging to a different cluster. The variety of techniques for the three steps of clustering: feature extraction and representing data; measuring similarity between samples; and grouping them has produced a rich and often confusing assortment of clustering methods.

The most challenging step in clustering, as in the case of preparing features of the training data set for a classifier, is feature extraction or pattern representation. Pattern recognition researchers conveniently avoid this step by assuming that the pattern representations are available as input to the clustering algorithm. But it is not possible to use a large set of measurements directly in clustering because of computational costs. So several feature extraction and selection approaches have been designed to obtain linear or nonlinear combinations of these measurements, which can be used to represent patterns.

The second step in clustering is similarity computation. A variety of schemes have been used to compute similarity between two patterns. They use knowledge either implicitly or explicitly. Most of the knowledge based clustering algorithms use explicit knowledge in similarity computation. However, if patterns are not represented using proper features, then it is not possible to get a meaningful partition irrespective of the quality and quantity of knowledge used in similarity computation. There is no universally acceptable scheme for computing similarity between patterns represented using a mixture of both qualitative and quantitative features. Dissimilarity between a pair of patterns is represented using a distance measure that may or may not be a metric.

Pattern similarity is usually measured by a distance function defined on pairs of patterns. A variety of distance measures are in use in the various communities (Anderberg, 1973), (Jain and Dubes, 1988), (Diday and Simon, 1976). A simple distance measure like Euclidean distance, which a special case of the Minkowski metric, can often be used for continuous features to reflect dissimilarity between two patterns, whereas other similarity measures can be used to characterize the conceptual similarity between patterns (Michalski and Stepp, 1983). Direct use of the Minkowski metric has the drawback of the tendency of the largest scaled feature to dominate the others. Solutions to this problem include normalization of the continuous features or other weighting schemes. Linear correlation among features can also distort distance measures, this distortion can be alleviated by applying a whitening transformation to the data or by using the squared Mahalanobis distance. Since similarity is fundamental to the definition of a cluster, a measure of the similarity between two patterns drawn from the same feature space is essential to most clustering procedures. Because of the variety of feature types and scales, the distance measure must be chosen carefully. It is most common to calculate the dissimilarity between two patterns using a distance measure defined on the feature space. The computation of distances between patterns with some or all features being noncontinuous is also a problematic issue. Wilson and Martinez (Wilson and Martinez, 1997) proposes a combination of a modified Minkowski metric for continuous features and a distance based on population for nominal attributes. A variety of other metrics have been reported in (Diday and Simon, 1976) and (Ichino and Yaguchi, 1994) for computing the similarity between patterns represented using quantitative as well as qualitative features.

There are some distance measures reported in the literature (Gowda and Krishna, 1977), (Jarvis and Patrick, 1973) that take into account the effect of surrounding or neighbouring points. These surrounding points are called context in (Michalski and Stepp, 1983). Since similarity plays a key role in our intuitive notion of a cluster, nearest neighbour distances can serve as the basis of clustering procedures. An iterative procedure was proposed in (Lu and Fu, 1978), it assigns each unlabeled pattern to the cluster of its nearest labeled neighbour pattern, provided the distance to that labeled neighbour is below a threshold. The k-means is the simplest and most commonly used algorithm employing a squared error criterion (McQueen, 1967).

The final step in clustering is the grouping step. There are broadly two grouping schemes: hierarchical and partitional schemes. The hierarchical schemes are more versatile, and the partitional schemes are less expensive. The partitional algorithms aim at maximizing the squared error criterion function.

Hierarchical clustering algorithms produce a nested series of partitions based on a criterion for merging or splitting clusters based on similarity. Partitional clustering algorithms identify the partition that optimizes a clustering criterion. Additional techniques for the grouping operation include probabilistic (Brailovsky, 1991) and graph theoretic (Zahn, 1971) clustering methods. Most hierarchical clustering algorithms are variants of the singlelink (Sneath and Sokal, 1973), completelink (King, 1967), and minimum variance (Ward, 1963), (Murtagh, 1984) algorithms. Of these, the singlelink and completelink algorithms are most popular. These two algorithms differ in the way they characterize the similarity between a pair of clusters. In the singlelink method, the distance between two clusters is the minimum of the distances all pairs of patterns drawn from the two clusters. In the completelink algorithm, the distance between two clusters is the maximum of all pairwise distances between patterns in the two clusters. In either case, two clusters are merged to form a larger cluster based on minimum distance criteria

Partitional clustering algorithms obtain a single partition of the data instead of a clustering structure, such as dendrogram produced by a hierarchical technique. Partitional methods are advantageous in applications involving large data sets for which the

construction of a dendrogram is computationally prohibitive. A problem accompanying the use of a partitional algorithm is the choice of the number of desired output clusters.

The grouping step can be performed in a number of ways. The output clustering can be hard or fuzzy, where each pattern has a variable degree of membership in each of the output clusters. Hence, the clusters in a hard clustering are disjoint. Fuzzy clustering extends this notion to associate each pattern with every cluster using a membership function (Zadeh, 1965). The output of such algorithms is a clustering, but not a partition.

Competitive neural networks (Jain and Mao, 1996) are often used to cluster input data. In competitive learning, similar patterns are grouped by the network and represented by a neuron. This grouping is done automatically based on data correlations. Well-known examples of neural networks used for clustering include Kohonen's learning vector quantization and self-organizing map (Kohonen, 1984), and adaptive resonance theory models (Carpenter and Grossberg, 1990).

The best-known graph theoretic divisive clustering algorithm is based on construction of the minimal spanning tree of the data (Zahn, 1971), and then deleting the minimal spanning tree edges with the largest lengths to generate clusters.

A number of books on clustering have been published (Jain and Dubes, 1988), (Anderberg, 1973), (Hartigan, 1975), (Spath, 1980), (Duran and Odell, 1974), (Everitt, 1993), (Backer, 1995), in addition to some useful and influential review papers. Cluster analysis was surveyed in (Dubes and Jain, 1980), (Jain *et al.*, 1986), (Jain *et al.*, 1999). A comparison of various clustering algorithms for constructing the minimal spanning tree and the short spanning path was given in (Lee, 1981). A review of image segmentation by clustering was reported in (Jain and Flynn, 1996). Comparisons of various combinatorial optimization schemes, based on experiments, have been reported in (Mishra and Raghavan, 1994) and (Al-Sultan and Khan, 1996). There is no clustering technique that is universally applicable in uncovering the variety of structures present in multidimensional data sets. Even though there is an increasing interest in the use of clustering methods in pattern recognition (Anderberg, 1973), image processing (Jain and Flynn, 1996) and information retrieval (Rasmussen, 1992), (Salton, 1991), clustering has a rich history in other

disciplines (Jain and Dubes, 1988) such as biology, psychiatry, psychology, archaeology, geology, geography, and marketing. Other terms more or less synonymous with clustering include unsupervised learning, numerical taxonomy (Sneath and Sokal, 1973) and learning by observation (Michalski and Stepp, 1983). The field of spatial analysis of point patterns (Ripley, 1989) is also related to cluster analysis.

In this thesis the final step of clustering is extended by solving a one-to-one assignment problem of the cluster groups to classes using the class info of the training data, to be able use them as classifiers. The optimization problem tries to minimize the total wrong class assignment.

3.7. Support Vector Machines

The aim of Support Vector classification is to devise a computationally efficient way of learning separating hyperplanes in a high dimensional feature space. The hyperplanes try to optimize the generalization bounds like, the maximal margin, the number of support vectors, the margin distribution, etc... Support Vector Machines are a system for efficiently training the linear learning machines in the kernel-induced feature spaces (Christianini and Taylor 2000). An important feature of these systems is that, while enforcing the learning biases suggested by the generalization theory, they also produce sparse dual representations of the hypothesis, resulting in extremely efficient algorithms. This is due to the Karush-Kuhn-Tucker conditions, which hold for the solution and play a crucial role in the practical implementation and analysis of these machines. Another important feature of the Support Vector approach is that due to Mercers conditions on the kernels the corresponding optimization problems are convex and hence have no local minima. This fact and the reduced number of non-zero parameters, mark a clear distinction between these system and other pattern recognition algorithms, such as neural networks (Cortes and Vapnik, 1995). There are many studies on Support Vector Machines and optimization of it (Schölkopf *et al.*, 1998), (Burges and Schölkopf, 1997), (Osuna *et al.*, 1997), (Schölkopf *et al.*, 1998), the use of Support Vector Machines for regression (Smola and Schölkopf, 1998), density estimation (Weston *et al.*, 1997) and anova decomposition (Stitson *et al.*, 1997) has also been studied.

For a separable data set, $\{x_i,w_i\}$ i=1..n, $x_i \in R^d$, $w_i \in \{-1,1\}$, suppose we have a separating hyperplane H. The points x_i which lie on the hyperplane H satisfy $A \cdot x_i + b = 0$, where W is normal to the hyperplane, $|b|/\|A\|$ is the perpendicular distance from the hyperplane to the origin, and $\|A\|$ is the Euclidean norm of A. Let $d_+(d_-)$be the shortest distance from the separating hyperplane to the closest positive (negative) sample data. Define the margin of a separating hyperplane to be $d_+ + d_-$. For the linearly separable case, the support vector algorithm simply looks for the separating hyperplane with largest margin (Boser *et al.*, 1992). This can be formulated as follows, suppose that all the training data satisfy the following constraints:

$$x_i \cdot A + b > +1 \text{ for } w_i = +1 \tag{3.8}$$

$$x_i \cdot A + b < -1 \text{ for } w_i = -1 \tag{3.9}$$

These can be combined into one set of inequalities:

$$w_i(x_i \cdot A + b) - 1 \geq 0, \forall I \tag{3.10}$$

These points lie on the hyperplane H_1: $x_i \cdot A + b = 1$, with normal A and perpendicular distance from the origin $|1-b|/\|A\|$. Similarly, the points for which the second equality holds lie on the hyperplane H_2: $x_i \cdot A + b = -1$, with normal again A, and perpendicular distance from the origin $|-1-b|/\|A\|$. Hence $d_+ = d_- = 1/\|A\|$ and the margin is simply $2/\|A\|$. Note that H_1 and H_2 are parallel and that no training points fall between them. Thus we can find the pair of hyperplanes which gives the maximum margin by minimizing $1/\|A\|^2$, subject to the combined constraint of inequalities.

Using the Lagrangian formulation of the problem benefits two points. The first is that the constraints will be replaced by constraints on the Lagrange multipliers themselves, which will be much easier to handle. The second is that in this reformulation of the problem, the training data will only appear in the form of dot products between vectors. This is a crucial property, which will allow us to generalize the procedure to the nonlinear case.

$$L_{Primal} = \frac{1}{2}\|A\|^2 - \sum_{i=1}^{n} \alpha_i w_i (x_i \cdot A + b) + \sum_{i=1}^{n} \alpha_i \tag{3.11}$$

We must now minimize L_{Primal} with respect to A, b, and simultaneously require that the derivatives of L_{Primal} with respect to all the α_i vanish, all subject to the constraints $\alpha_i \geq 0$. Now this is a convex quadratic programming problem, since the objective function is itself convex, and those points, which satisfy the constraints, also form a convex set. This means that we can equivalently solve the following dual problem: maximize L_{Primal}, subject to the constraints that the gradient of L_{Primal} with respect to A and b vanish, and subject also to the constraints that the $\alpha_i \geq 0$ (Kaufman, 1998). This particular dual formulation of the problem is called the Wolfe dual (Fletcher, 1987).

Introducing positive slack variables s_i, i=1,..,n to the constraints, $\sum_{i=1}^{n} s_i$ is an upper bound on the number of training errors. Hence a natural way to assign an extra cost for errors is to change the objective function to be minimized from $\frac{\|A\|^2}{2}$ to $\frac{\|A\|^2}{2} + C\left(\sum_{i=1}^{n} s_i\right)^d$, where C is a parameter to be chosen by the user, a larger C corresponding to assigning a higher penalty to errors. As it stands, this is a convex programming problem for any positive integer d; for d=2 and d=1 it is also a quadratic programming problem, and the choice d=1 has the further advantage that neither the si, nor their Lagrange multipliers, appear in the Wolfe dual problem, which becomes:

$$\begin{aligned} &\text{maximize } L_{Dual} = \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j w_i w_j x_i x_j \\ &\text{subject to } 0 \leq \alpha_i \leq C \text{ and } \sum_i \alpha_i w_i = 0 \end{aligned} \tag{3.12}$$

The solution is given by $A = \sum_{i=1}^{N_s} \alpha_i w_i x_i$, where N_s is the number of support vectors. Thus the only difference from the optimal hyperplane case is that the α_i now has an upper bound of C.

In order to learn non-linear relations with a linear machine, we need to select a set of non-linear features and to rewrite the data in the new representation. This is equivalent to applying a fixed non-linear mapping of the data to a feature space, in which the linear machine can be used. Hence, the set of hypotheses we consider will be functions of the type

$$f(x) = \sum_{i=1}^{N} a_i \Phi_i(x) + b \tag{3.13}$$

where Φ: X→F is a non-linear map from the input space to some feature space. This means that we will build non-linear machines in two steps: first a fixed non-linear mapping transforms the data into a feature space F, and then a linear machine is used to classify them in the feature space. One important property of linear learning machines is that they can be expressed in a dual representation. This means that the hypothesis can be expressed as a linear combination of the training points, so that the decision rule can be evaluated using just inner products between the test point and the training points:

$$f(x) = \sum_{i=1}^{l} a_i w_i (\Phi(x_i) \cdot \Phi(x)) + b \tag{3.14}$$

If we have a way of computing the inner product $(\Phi(x_n) \cdot \Phi(x))$ in feature space directly as a function of the original input points, it becomes possible to merge the two steps needed to build a non-linear learning machine. We call such a direct computation method a kernel function. The use of kernels makes it possible to map the data implicitly into a feature space and to train a linear machine in such a space, potentially side-stepping the computational problems inherent in evaluating the feature map.

The key to this approach is finding a kernel function that can be evaluated efficiently. Once we have such a function the decision rule can be evaluated by at most one evaluation of the kernel:

$$f(x) = \sum_{i=1}^{l} a_i w_i K(x_i, x) + b \tag{3.15}$$

One of the curious facts about using a kernel is that we do not need to know the underlying feature map in order to be able to learn in the feature space. The use of a kernel function is an attractive computational short-cut. If we wish to use this approach, there appears to be a need to first create a complicated feature space, then work out what the inner product in that space would be, and finally find a direct method of computing that value in terms of the original inputs. In practice the approach taken is to define a kernel function directly, hence implicitly defining the feature space. In this way, we avoid the feature space not only in the computation of inner products, but also in the design of the learning machine itself. We will argue that defining a kernel function for an input space is frequently more natural than creating a complicated feature space. Before we can follow this route, however, we must first determine what properties of a function K(x,z) are necessary to ensure that it is a kernel for a some feature space. Clearly, the function must be symmetric, and satisfy the inequalities that follow from the Cauchy-Schwarz inequality. These conditions are, however, not sufficient to guarantee the existence of a feature space and Mercer's theorem should be also satisfied.

The first kernels investigated for the pattern recognition problem were the following:

- Linear: $K(x,z)=x\cdot z'$
- Polynomial: $K(x,z)=(x\cdot z'+1)^p$
- Radial Basis: $K(x,z)=e^{\frac{-(x-z)\cdot(x-z)'}{2p^2}}$
- Sigmoid: $K(x,z)=\tanh(p_1\cdot x\cdot z'-p_2)$

For the radial basis case, the number of support vectors, the support vectors themselves, the weights, and the threshold are all produced automatically by the support vector machine training and give excellent results compared to classical Gaussian radial basis functions (Schölkopf *et al.*, 1997).

4. COMBINING CLASSIFIERS

There are at least three reasons why good ensembles can be constructed and why it may be difficult or impossible to find a single classifier that performs as well as the combination. To understand these reasons, we must consider the nature of machine learning algorithms. Machine learning algorithms work by searching a space of possible hypotheses H for the most accurate hypothesis, that is, the hypothesis that best approximates the unknown function f. Two important aspects of the hypothesis space H are its size and whether it contains good approximations to f.

If the hypothesis space is large, then we will need a large amount of training data to constrain the search for good approximations. Each training sample rules out or makes less plausible, all those hypotheses in H those misclassify it. In a 2-class problem, ideally each training sample can eliminate half of the hypotheses in H, so we require O(log |H|) samples to select a unique classifier from H (Dietterich, 1997).

First, the need for combination is that the training data may not provide sufficient information for choosing a single best classifier from H (Clemen, 1989). Most of the learning algorithms consider very large hypothesis spaces, so even after eliminating hypotheses that misclassify training samples, there are many hypotheses remaining. All of these hypotheses appear equally accurate with respect to the available training data. We may have reasons for preferring some of these hypotheses over others, preferring simpler hypotheses or hypotheses with higher prior probability. But nonetheless, there are typically many plausible hypotheses. From this collection of surviving hypothesis in H, we can easily construct a combination of classifiers.

Second, the need for combination is that our learning algorithms may not be able to solve the difficult search problems that we pose. For example, finding the weights for the smallest possible neural network consistent with the training samples is NP-hard. Neural network algorithms therefore employ local search methods such as gradient descent to find locally optimal weights for the network. A consequence of these imperfect search algorithms is that even if the combination of our training samples and our prior knowledge,

preferences for simple hypotheses, Bayesian priors, determines a unique best hypothesis, we may not be able to find it. Instead, we will typically find a hypothesis that is somewhat more complex or has somewhat lower posterior probability (Ali and Pazzani, 1996). If we run our search algorithms with a slightly different training sample or injected noise, we will find a different suboptimal hypothesis. Combination can be seen therefore as a way of compensating for imperfect search algorithms.

Third, the need for combination is that our hypothesis space H may not contain the true function f. Instead, H may include several equally good approximations to f. By taking weighted combinations of these approximations, we may be able to represent classifiers that lie outside of H (McQueen, 1967). One way to understand this is to visualize the decision boundaries constructed by learning algorithms. A decision boundary is a surface such that samples that lie on one side of the surface are assigned to a different class than samples that lie on the other side of the surface. Combination also provides a way of overcoming representational inadequacies in our hypothesis space.

A large number of combination methods have been proposed in the literature (Xu *et al.*, 1992), (Kittler, 1998), (Kittler *et al.*, 1998). A typical combination method consists of a set of individual classifiers and a combiner, which combines the results of the individual classifiers to make the final classification. When the individual classifiers should be invoked or how they should interact with each other is determined by the architecture of the combination method. Thus, various combination methods may differ from each other in their architectures, the characteristics of the combiner, and selection of the individual classifiers.

Various methods for combining multiple classifiers can be grouped into three main categories according to their architecture:

- Parallel
- Serial
- Hierarchical

In the parallel architecture, all the individual classifiers are invoked independently, and a combiner then combines their results. Most combination methods in the literature belong to this category. In the gated parallel variant, the outputs of individual classifiers are selected or weighted by a gating device before they are combined. In the serial architecture, individual classifiers are invoked in a linear sequence. The number of possible classes for a given pattern is gradually reduced as more classifiers in the sequence have been invoked. For the sake of efficiency, inaccurate but cheap classifiers with low computational and measurement demands, are considered first, and followed by more accurate and expensive classifiers. In the hierarchical architecture, individual classifiers are combined into a structure, which is similar to that of a decision tree classifier. The tree nodes, however, may now be associated with complex classifiers demanding a large number of features. The advantage of this architecture is the high efficiency and flexibility in exploiting the discriminant power of different types of features. Using these three basic architectures, we can build even more complicated classifier combination systems.

Consider a pattern recognition problem where sample test pattern x_n is to be assigned to one of the C possible classes. Let us assume that we have K classifiers each representing the given pattern by a distinct feature vector. Denote the feature vector used by the k^{th} classifier m_k by x_{nk}. In the measurement space each class c is modeled by the probability density function $P(x_{nk}|w_c)$ and its a priori probability of occurrence is denoted $P(w_c)$. We shall consider the models to be mutually exclusive which means that only one model can be associated with each pattern.

According to the Bayesian theory, given a specific feature vector x_{nk}, the sample test pattern x_n, should be assigned to class w_c, provided the a posterior probability of that interpretation is maximum,

$$\text{Assign } x_n \rightarrow w_c \text{ if } P(w_c \mid x_{n1},\ldots,x_{nk}) = \max_{i=1..c} P(w_i \mid x_{n1},\ldots,x_{nK}) \qquad (4.1)$$

The Bayesian decision rule states that, in order to utilize all the available information correctly to reach a decision, it is essential to compute the probabilities of the various hypotheses by considering all the classifiers simultaneously. This is, of course, a correct

statement of the classification problem but it may not be a practicable proposition. The computation of the a posterior probability functions would depend on the knowledge of high-order measurement statistics described in terms of joint probability density functions $P(x_{n1},\ldots,x_{nK}|w_c)$ that would be difficult to infer. We shall therefore attempt to simplify the above rule and express it in terms of decision support computations performed by the individual classifiers, each exploiting only the information conveyed by vector x_{nk}. We shall see that this will not only make Bayesian decision rule computationally manageable, but also it will lead to combination rules which are commonly used in practice. Moreover, this approach will provide a scope for the development of a range of efficient classifier combination strategies.

We shall commence from Bayesian decision rule and consider how it can be expressed under certain assumptions. Let us rewrite the posterior probability $P(w_c|x_{n1},\ldots,x_{nK})$ using the Bayes theorem. We have $P(w_c \mid x_{n1},\ldots,x_{nK}) = \frac{P(x_{n1},\ldots,x_{nK} \mid w_c)P(w_c)}{P(x_{n1},\ldots,x_{nK})}$, where $P(x_{n1},\ldots,x_{nK})$ is the unconditional joint probability density function of the sample pattern, which can be expressed in terms of conditional probabilities as $P(x_{n1},\ldots,x_{nK}) = \sum_{c=1}^{C} P(x_{n1},\ldots,x_{nK} \mid w_c)P(w_c)$

From the point of view of the selection of input feature vectors there are basically two classifier combination scenarios. In the first scenario, all the classifiers use the same representation of the input sample test pattern x_n. In this case, each classifier, for a given input pattern, can be considered to produce an estimate of the same posterior class probability.

In the second scenario, each classifier uses its own representation of the input sample test pattern x_{nk}. In other words, the feature vectors extracted from the sample pattern are unique to each classifier. An important application of combining classifiers in this scenario is the possibility to integrate physically different types of measurements/features. In this case, it is no longer possible to consider the computed posterior probabilities to be

estimates of the same functional value, as the classification systems operate in different measurement spaces.

An important requirement for a combiner that uses the output of the individual classifiers is that the classifiers should not be strongly correlated in their misclassification. That is, classifiers should not agree with each other when they misclassify a sample, or at least they should not assign the same incorrect class to a sample. This requirement can be satisfied to a certain extent by using different feature sets and using a different classification principle for each of the individual classifiers. Using different feature sets leads, in many cases, to a reduction in the correlation between the outputs of individual classifiers, since there is almost always less correlation between the input vectors using different representations than when using the same set of features. Different classifiers usually use different assumptions about the structure of the data and the stochastic model that generates it. This leads to a different estimate of the posterior probabilities especially around the Bayes decision boundaries.

Under different assumptions and using different approximations, we can derive the commonly used classifier combination methods such as the product rule, sum rule and their approximations as min rule, max rule and majority voting.

4.1. Product Rule

Let us assume that the classifiers are statistically independent, which will lead us to rewrite the joint probability density function $P(x_{n1},...,x_{nK} \mid w_c) = \prod_{k=1}^{K} P(x_{nk} \mid w_c)$. Substituting this in the Bayes theorem, we find $P(w_c \mid x_{n1},...,x_{nK}) = \frac{P(w_c)\prod_{k=1}^{K} P(x_{nk} \mid w_c)}{\sum_{i=1}^{C} P(w_i)\prod_{k=1}^{K} P(x_{nk} \mid w_i)}$ and using the Bayesian decision rule:

$$\text{Assign } x_n \rightarrow w_c \text{ if } P(w_c)\prod_{k=1}^{K} P(x_{nk} \mid w_c) = \max_{i=1..C}\left[P(w_i)\prod_{k=1}^{K} P(x_{nk} \mid w_i)\right] \quad (4.2)$$

In terms of the posterior probabilities the decision rule could be written as:

$$\text{Assign } x_n \rightarrow w_c \text{ if } \frac{\prod_{k=1}^{K} P(w_c \mid x_{nk})}{P(w_c)^{K-1}} = \max_{i=1..C}\left[\frac{\prod_{k=1}^{K} P(w_i \mid x_{nk})}{P(w_i)^{K-1}}\right] \quad (4.3)$$

The assumption of independence of classifiers of the product rule is a strong one, if we are not so selective on the type of the classifiers for combination, even further the x_{nk}'s are not independent. One more disadvantage is that, if we have a bad classifier m_k in terms of posterior probability $P(w_{kn}|x_{nk})\cong 0$, the overall classification result is inhibited.

4.2. Min Rule

We can approximate the product rule, by bounding the product of posterior probabilities from above: $\prod_{k=1}^{K} P(w_c \mid x_{nk}) \leq \min_{k=1..K} P(w_c \mid x_{nk})$ we obtain

$$\text{Assign } x_n \rightarrow w_c \text{ if } \frac{\min_{k=1..K} P(w_c \mid x_{nk})}{P(w_c)^{K-1}} = \max_{i=1..C}\left[\frac{\min_{k=1..K} P(w_i \mid x_{nk})}{P(w_i)^{K-1}}\right] \quad (4.4)$$

If we further assume that the prior probabilities are equal, this simplifies to

$$\text{Assign } x_n \rightarrow w_c \text{ if } \min_{k=1..K} P(w_c \mid x_{nk}) = \max_{i=1..C} \min_{k=1..K} P(w_i \mid x_{nk}) \quad (4.5)$$

4.3. Sum Rule

Let us assume that the posterior probabilities computed by the classifiers will not deviate dramatically from the prior probabilities. We can assume that the posterior probabilities can be expressed as $P(w_c|x_{nk})=P(w_c)(1+\delta_{ck})$, where δ_{ck} satisfies $|\delta_{ck}|<<1$.

$$\frac{\prod_{k=1}^{K} P(w_c \mid x_{nk})}{P(w_c)^{K-1}} = P(w_c)\prod_{k=1}^{K}(1+\delta_{ck})$$

If we expand the product and neglect any terms of second and higher order, we can approximate the right-hand side as

$$P(w_c)\prod_{k=1}^{K}(1+\delta_{ck}) = P(w_c)+P(w_c)\sum_{k=1}^{K}\delta_{ck} = P(w_c)+P(w_c)\sum_{k=1}^{K}\left[\frac{P(w_c \mid x_{nk})}{P(w_c)}-1\right]$$

This simplifies to the sum rule

$$\text{Assign } x_n \rightarrow w_c \text{ if } (1-K)P(w_c)+\sum_{k=1}^{K}P(w_c \mid x_{nk}) = \max_{i=1..C}\left[(1-R)P(w_i)\sum_{k=1}^{K}P(w_i \mid x_{nk})\right] \quad (4.6)$$

As far as the sum rule is concerned, the assumption that the posterior class probabilities do not deviate greatly from the prior probabilities will be unrealistic in most applications. When observations on a pattern convey significant discriminatory information, the sum approximation of the product will introduce approximation errors.

4.4. Max Rule

We can approximate the sum rule by the maximum of the posterior probabilities, since $(\frac{1}{K})\sum_{k=1}^{K}P(w_c \mid x_{nk}) \le \max_{k=1..K} P(w_c \mid x_{nk})$

Assign $x_n \rightarrow w_c$ if

$$(1-K)P(w_c)+K\max_{k=1..K}P(w_c \mid x_{nk}) = \max_{i=1..C}\left[(1-K)P(w_i)K\max_{k=1..K}P(w_i \mid x_{nk})\right] \quad (4.7)$$

If we further assume that the prior probabilities are equal, this simplifies to

$$\text{Assign } x_n \rightarrow w_c \text{ if } \max_{k=1..K}P(w_c \mid x_{nk}) = \max_{i=1..C}\max_{k=1..K}P(w_i \mid x_{nk}) \quad (4.8)$$

4.5. Mean/Median Rule

If we assume equal prior probabilities, the sum rule can be viewed as computing the average posterior probabilities for each class over all the classifier outputs:

$$\text{Assign } x_n \rightarrow w_c \text{ if } \left(\frac{1}{K}\right)\sum_{k=1}^{K}P(w_c \mid x_{nk}) = \max_{i=1..C}\left(\frac{1}{K}\right)\sum_{k=1}^{K}P(w_i \mid x_{nk}) \quad (4.9)$$

This rule assigns a sample pattern to that class with maximum average posterior probability. If any of the classifiers outputs a posterior probability for some class, which is an outlier, it will affect the average and this is in turn could lead to an incorrect decision. It is well known that a robust estimate of the mean is the median. It could therefore be more appropriate to base the combined decision on the median of the posterior probabilities, which leads to the following rule:

$$\text{Assign } x_n \rightarrow w_c \text{ if } \operatorname*{med}_{k=1..K}P(w_c \mid x_{nk}) = \max_{i=1..C}\operatorname*{med}_{k=1..K}P(w_i \mid x_{nk}) \quad (4.10)$$

4.6. Majority Vote Rule

Let us force the posterior probabilities to produce binary valued function Δ_{cnk} as:

$\Delta_{cnk}=1$ if $P(w_c \mid x_{nk}) = \max_{i=1..C}P(w_i \mid x_{nk})$ and 0 otherwise.

This function results in combining decision outputs to be class labels rather than posterior probabilities. If we further assume that the prior probabilities are equal we find:

$$\text{Assign } x_n \rightarrow w_c \text{ if } \sum_{k=1}^{K} \Delta_{cnk} = \max_{i=1..C} \sum_{k=1}^{K} \Delta_{ink} \tag{4.11}$$

Note that for each class w_i the sum on the right hand side is simply the count of the votes received from the individual classifiers. The class, which receives the largest number of votes, is then selected as majority decision.

4.7. Bagging

Classifier combination methods are methods for improving the accuracy of a classifier through aggregation of several similar classifiers predictions. In bagging, the training set is perturbed by bootstrapping and the results combined with a majority vote. Typically, such aggregation will arrive at greater accuracy by reducing either the bias or the variance of a single classifier (Breiman, 1996a), (Dietterich, 1997).

In bagging, an inductive learner is repeatedly applied to bootstrap samples (Efron and Tibshirani, 1993) from a master training set, where a bootstrap sample is generated by random sampling with replacement from the original training data. Once several classifiers have been trained on such bootstraps, the classifier combination is determined by majority voting among the classifiers. That is, the classifier combination evaluates test samples by querying each of the classifiers on the sample and then outputting their majority opinion. Breiman's two main arguments for baggings appropriateness and effectiveness are that running several trials on uniform samples of a population results in more significant, less variant statistical results, and deferring to the majority opinion can rid the final classifier of noise-induced errors or other mistakes that occur only in a handful of the classifiers (Bauer and Kohavi, 1999), (Breiman, 1996a).

4.8. Boosting

Boosting also operates on random subsets but is more complex, and constructs a filtered sequence of classifiers that increases the probability of selecting previously misclassified patterns followed by weighted vote combination (Freund and Schapire,

1997). Boosting, in particular, has been shown to be very effective for turning a series of weak learners into a strong learner.

Boosting improves classification accuracies by iteratively estimating classifiers using a base-learning algorithm while systematically varying the training sample. At each iteration, the training sample is modified to focus the classification algorithm on examples that are difficult to classify correctly. This modification is performed by providing a weight for each training sample. The importance of misclassified training samples is increased and the classification algorithm focuses on learning these samples. The boosted classifier's prediction is then based upon an accuracy-weighted vote across the estimated classifiers. A number of different boosting algorithms have been developed.

Recall that bagging gives all samples equal probability of being selected for the training set of each classifier and gives each resulting classifier combination equal voting power. Boosting is similar to bagging but attempts to learn more intelligently than bagging in two ways.

First, each time boosting creates a new classifier, its accuracy is verified through testing on the training data (Dietterich, 1997). The probability distribution over the samples is then re-weighted based on a measure of how hard each sample was to classify for the current classifier. The harder samples probabilities of being picked in the subsequent classifiers bootstrap are thus boosted in the distribution, such that the next classifier will focus on solving the harder to classify samples. The second way in which boosting diverges from bagging is in the voting phase of the combination. During testing, boosting weights the vote of each classifier according to how successful it was during verification.

4.9. Error Sensitivity

We assumed that posterior class probabilities $P(w_i|x_{nk})$, in terms of which the various classifier combination rules are defined, are estimated correctly. In fact, each classifier m_k

will produce only an estimate of this probability, which we shall denote $\bar{P}(w_i \mid x_{nk})$. The estimate deviates from the true probability by error e_{ink}

$$\bar{P}(w_i \mid x_{nk}) = P(w_i \mid x_{nk}) + e_{ink} \tag{4.12}$$

It is these estimated probabilities that enter the classifier combination rules rather than the true probabilities. (Kittler, 1998) addresses the issue of the sensitivity of various combination rules to estimation errors, and point out that the techniques based on the sum rule are more reliable to errors than those derived from the product rule. Let us now consider the effect of the estimation errors on the classifier combination rules.

If we further assume that $e_{ink} << P(w_i|x_{nk})$ and $P(w_i|x_{nk})>0$, we can rewrite the product and sum rules as follows:

$$\text{Assign } x_n \rightarrow w_c \text{ if } \frac{\prod_{k=1}^{K}[P(w_c \mid x_{nk}) + e_{cnk}]}{P(w_c)^{K-1}} = \max_{i=1..C}\left[\frac{\prod_{k=1}^{K}(P(w_i \mid x_{nk}) + e_{ink})}{P(w_i)^{K-1}}\right] \tag{4.13}$$

We can rewrite by rearranging terms as:

$$\prod_{k=1}^{K}[P(w_i \mid x_{nk}) + e_{ink}] = \prod_{k=1}^{K} P(w_i \mid x_{nk}) \prod_{k=1}^{K}\left[\frac{1 + e_{ink}}{P(w_i \mid x_{nk})}\right]$$ which can be linearized as:

$$\prod_{k=1}^{K}[P(w_i \mid x_{nk}) + e_{ink}] = \prod_{k=1}^{K} P(w_i \mid x_{nk})\left[1 + \sum_{k=1}^{K}\frac{e_{ink}}{P(w_i \mid x_{nk})}\right]$$

We reach to the product rule with the error of estimation in the posterior probabilities:

Assign $x_n \rightarrow w_c$ if

$$\frac{\prod_{k=1}^{K} P(w_c \mid x_{nk})}{P(w_c)^{K-1}}\left[1+\sum_{k=1}^{K}\frac{e_{cnk}}{P(w_c \mid x_{nk})}\right] = \max_{i=1..C}\left[\frac{\prod_{k=1}^{K} P(w_i \mid x_{nk})}{P(w_i)^{K-1}}\right]\left[1+\sum_{k=1}^{K}\frac{e_{ink}}{P(w_i \mid x_{nk})}\right] \quad (4.14)$$

Comparing this rule with the error free version, each term in the error free classifier combination rule is affected by the error factor.

$$\text{Product rule error factor} = \left[1+\sum_{k=1}^{K}\frac{e_{ink}}{P(w_i \mid x_{nk})}\right] \quad (4.15)$$

A similar analysis of the sum rule is as follows:

$$\text{Assign } x_n \rightarrow w_c \text{ if}$$

$$(1-K)P(w_c)+\sum_{k=1}^{K}\left[P(w_c \mid x_{nk})+e_{cnk}\right] = \max_{i=1..C}\left[(1-K)P(w_i)\sum_{k=1}^{K}(P(w_i \mid x_{nk})+e_{ink})\right] \quad (4.16)$$

Which can be rewritten as

$$\text{Assign } x_n \rightarrow w_c \text{ if}$$

$$(1-K)P(w_c)+\sum_{k=1}^{K}P(w_c \mid x_{nk})\left[1+\frac{\sum_{k=1}^{K}e_{cnk}}{\sum_{k=1}^{K}P(w_c \mid x_{nk})}\right]$$

$$= \max_{i=1..C}\left[(1-K)P(w_i)\sum_{k=1}^{K}P(w_i \mid x_{nk})\left[1+\frac{\sum_{k=1}^{K}e_{ink}}{\sum_{k=1}^{K}P(w_i \mid x_{nk})}\right]\right] \quad (4.17)$$

A comparison to the original sum rule results in the error factor.

$$\text{Sum rule error factor} = \left[1 + \frac{\sum_{k=1}^{K} e_{ink}}{\sum_{k=1}^{K} P(w_i \mid x_{nk})} \right] \tag{4.18}$$

Comparing error factors for the product and sum rules, it is obvious that the sensitivity to errors of the product is much more dramatic than that of the sum rule. Note that since the posterior probabilities are less than unity, each error e_{ink} in error factor of the product rule is amplified by $P^{-1}(w_i|x_{nk})$. The compounded effect of all these amplified errors is equivalent to their sum. In contrast, in the sum rule, the errors are not amplified. On the contrary, their compounded effect, which is also computed as a sum, is scaled by the sum of the posterior probabilities. For the most probable class, this sum is likely to be greater than one, which will result in lowering of the errors. Thus, the sum decision rule is much more reliable to estimation errors. It follows, therefore, that the sum classifier combination rule is not only a very simple and intuitive technique of improving the reliability of decision making based on different classifier opinions but it is also remarkably robust.

5. AUTOMATED CLASSIFIER COMBINATION

Different pattern classifiers trained for the same data set can come out with different results and variable reliability. They may be derived from different concepts, using different features, or operating with different classifier parameters. Their performance may be different for the same data set. Their performance also depends on the classes of the data sets; some classifiers or different classifier's parameters perform well on some classes of the data set, but not on all of them. Under these circumstances combining different pattern classifiers developed for the same task bears the promise of improving the overall performance, just like in every day life where more than one expert is consulted if a difficult case is to be settled. Since different pattern classifiers have different strengths and different weaknesses, classifier combination must be led by the goal of making the respective strengths effective and repelling the deficiencies.

Methods for combining classifiers can be subdivided into unweighted vote, weighted vote, gating networks and stacking. The simplest approach is to take an unweighted vote as is done in bagging and many other methods. While it may appear that more intelligent voting schemes should do better, the experience in the forecasting literature has been that simple, unweighted voting is very robust (Clemen, 1989).

One refinement that is proposed in the thesis on simple majority vote is based on the thought, that each classifier m_k can produce class probability estimates p_{knc} rather than a simple classification decision w_{kn}. A class probability estimate for sample x_n is the probability that the true class is w_c: $P_k(w_c|x_{nk})$ for c=1...C. For come classifiers, like artificial neural networks, this probability is already calculated if the training outputs are given as probabilities. In this case the results are in the form of posterior probabilities. For other classifiers, like k-nearest neighbour the similarity values calculated as distances should be converted to posterior probabilities. This conversion is made by normalizing the similarity measures, based on the maximum and minimum similarity values, after the leave-one-out training is performed. In the framework, which will be detailed in Section 5.1, all classifiers are modified to supply posterior probabilities for each class.

We can combine the class probabilities of all of the classifiers so that the class probability of the combination is $P(w_c \mid x_n) = \frac{1}{K}\sum_{k=1}^{K} P_k(w_c \mid x_n)$. The predicted class of x_n is then the class having the highest-class probability.

$$\text{Assign } x_n \rightarrow w_c \text{ if } P(w_c \mid x_n) = \max_{i=1..C} \frac{1}{K}\sum_{k=1}^{K} P_k(w_i \mid x_{nk}) \qquad (5.1)$$

Many different weighted voting methods have been developed for classifier combination. For regression problems, (Perrone and Cooper, 1993) and (Hashem, 1993) apply least squares regression to find weights that maximize the accuracy of the combination on the training data. They show that the weight applied to the classifier m_k should be inversely proportional to the variance of the estimates of itself. A difficulty with applying linear least squares is that the various classifiers can be very highly correlated. Methods have been described for choosing less correlated subsets of the classifier set and combining them using linear least squares.

For classification problems, weights are usually obtained by measuring the accuracy of each individual classifier on the training data and constructing weights that are proportional to those accuracies. (Ali and Pazzani, 1996) describe a method that they call likelihood combination in which they apply the Bayes algorithm to learn weights for classifiers. The weight for a classifier is computed from the accuracy measured on the training data.

The third approach to combining classifiers is to learn a gating network or a gating function that takes as input x_n and produces as output the weights W_k to be applied to compute the weighted vote of the classifiers (Jordan and Jacobs, 1994). As with any learning algorithm, there is a risk of overfitting the training data by learning the gating function in addition to learning each of the individual classifiers.

A fourth approach to combining classifiers, called stacking, is where different learning algorithms are applied to the training data at each level to teach a good combining

classifier. Wolpert proposed a scheme for learning using a form of leave-one-out cross-validation (Wolpert, 1992). Each algorithm is applied to the training data many times, leaving a training sample out each time. We can then apply each classifier to the sample to obtain the predicted class, which results in a new data set whose features are the classes predicted by each of the classifiers. Now we can apply some other learning algorithm to the new data to classify in a stacked manner. Breiman applied this approach to combining different forms of linear regression with very good results (Breiman, 1996b).

5.1. Performance Measures

To compare different classification algorithms and combination methods, performance measures should be defined. There are different performance measures which evaluates different performances of classifiers: generalization performance, learning performance, correct and wrong classification performance, real time performance, etc.... One of the measures is the number of operations, which is the count of mathematical operations performed. This measure depends on the number of samples of the data, on the length of the feature vector of the data and the classification algorithm itself.

For each classification result, a Cx(C+1) classification matrix as seen in Table 5.1 is given to show the correct, wrong and unclassified number of samples for the data set. In this matrix a value in the location (i,i) shows the number of correct classifications for class i, a value in location (i,(C+1)) is for the unclassified number of samples for class i and the other values in the matrix are for wrong classification showing a wrong classification for class i as class j.

Table 5.1. Classification matrix

Classes	**Class 1**	...	**Class i**	...	**Class C**	**Unclassified**	**Correct**	**Wrong**	**Performance**	**Reliability**
Class 1	$correct_1$	...	$wrong_{1i}$	...	$wrong_{1C}$	$unclassified_1$	$correct_1$	$wrong_1$	$performance_1$	$reliability_1$
...										
Class i	$wrong_{i1}$	...	$correct_i$	...	$wrong_{iC}$	$unclassified_i$	$correct_i$	$wrong_i$	$performance_i$	$reliability_i$
...										
Class C	$wrong_{C1}$	...	$wrong_{Ci}$	...	$correct_C$	$unclassified_C$	$correct_C$	$wrong_C$	$performance_C$	$reliability_C$

Next to the classification matrix, the total number of correct and wrong classification counts are given as vectors. There are two more columns next to them, which are the calculated results of performance and reliability for the classes. Performance is the ratio for correct classification and reliability is the probability of correct classification for that class.

$$\text{performance}_i = 100 \times \frac{\text{correct}_i}{\text{correct}_i + \text{wrong}_i + \text{unclassified}_i} \tag{5.2}$$

$$\text{reliability}_i = 100 \times \frac{\text{correct}_i}{\text{correct}_i + \sum_{j=1}^{C} \text{wrong}_{ji}} \tag{5.3}$$

$$CP = \frac{\sum_{i=1}^{C} \text{correct}_i}{\sum_{i=1}^{C} \text{samples}_i} \tag{5.4}$$

Classification performance based on classes (CP) is based on the ratio of correct classification to the sample size. Generally, only this correct classification performance measure is used in the literature. In this thesis, other measures are proposed as defined in the following paragraphs and they are used to show how the different measures differ for determining a classifier's performance.

Classification performance based on probabilities (PP) is based on distances of the posterior probabilities p'_t of the classification result and the true classification probability p_t.

$$PP = 1 - \frac{\sum_{t=1}^{N} | p'_i - p_t |}{\sum_{i=1}^{C} \text{samples}_i} \tag{5.5}$$

Overall classification performance (OP) is a measure, which combines the products of performance_i and reliability_i with the counts of samples_i at corresponding class_i as a weight.

$$OP = \frac{\sum_{i=1}^{C} (\text{performance}_i \times \text{reliability}_i \times \text{samples}_i)}{\sum_{i=1}^{C} \text{samples}_i} \tag{5.6}$$

Sum of squared errors based on probabilities (SSEP) is the sum of squared differences of posterior probabilities wp'_t of the classification result and the true classification probability wp_t.

$$\text{SSEP} = \frac{\sum_{t=1}^{N} (p'_t - p_t)^2}{\sum_{i=1}^{C} \text{samples}_i} \tag{5.7}$$

Distance of probabilities (DP) is the Euclidean distance between posterior probabilities p'_t of the classification result and the true classification probability p_t.

$$\text{DP} = \frac{\sqrt{\sum_{t=1}^{N} (p'_t - p_t)^2}}{\sum_{i=1}^{C} \text{samples}_i} \tag{5.8}$$

In this thesis the classifiers are modified to produce posterior probabilities for each class. This allows to define a region where it is unreliable to classify samples, it is better to left them unclassified. This region is calculated using special values of posterior probabilities after the leave-one-out training phase. Minimum difference for true probability (MinT) is the maximum difference between posterior probabilities of the classification result for any wrong classification assignment to its correct classification posterior probability for the same sample. Maximum difference for wrong probability

(MaxW) is the minimum difference between posterior probability of the classification result for any correct classification assignment to the wrong classification result with the highset posterior probability for the same sample. The trust value to decide for the indecisive boundary, is either the MaxW if it is smaller than the MinT, otherwise it is 0.25(MinT-MaxW)+MaxW. Using the trust value in the classification of test samples, if the difference between the highest posterior probability to its next highest posterior probability is lower than the trust value, the sample is left unclassified.

In this thesis, to compare different classification algorithms and combination methods, different performance measures are calculated and the resulting tables include all the values for these measures. For the classification algorithms, the required parameter values are also given in the resulting tables. For the combination methods, the weight of classifiers for combination is also given. The training performance of all classifiers is calculated by using the training sets with leave-one-out technique.

5.2. Weighted Combination Algorithms

This study focuses on combining the results of several different classifiers in a way that provides a coherent inference, which performs a reasonable classification performance for the data set at hand. Therefore three different weighted combination algorithms with three different weight assignment are applied on the data sets. Class based combination algorithm uses the classifiers' class assignments to combine (Alexandre *et al.* 2000). For each class, the weights of the classifier is added to the decision value of the classifier combination, if the classifier has decided on that class. The classifier combination decides the assigned class based on the maximum of these decision values (5.9). If the weights are equal, this combination algorithm is the classical majority vote (4.11). The classifiers are forced to produce binary valued function Δ_{cnk} using the posterior probabilities as: $\Delta_{cnk}=1$ if $P(w_c \mid x_{nk}) = \max_{i=1..C} P(w_i \mid x_{nk})$ and 0 otherwise.

$$\text{Assign } x_n \rightarrow w_c \text{ if } \sum_{k=1}^{K} W_k \Delta_{cnk} = \max_{i=1..C} \sum_{k=1}^{K} W_k \Delta_{ink} \tag{5.9}$$

The reliability of the classifier for the assigned class, as proposed in (5.3), is also integrated to the combination by multiplying the reliability value with the weighted class assignment. This idea is intuitive, if we handle the classifiers as experts in the decision theory. That is, each classifier has different reliabilities on deciding on different classes. For the case of classifiers, this handles the situations, where the classifier performances differ to classify the sample of different classes of the same data set. In other words, this reliability value increases the influence on the final decision, if the classifier reliability is high for deciding this class, and decreases the influence, if it is unreliable.

$$\text{Assign } x_n \rightarrow w_c \text{ if } \sum_{k=1}^{K} W_k R_{kc} \Delta_{cnk} = \max_{i=1..C} \sum_{k=1}^{K} W_k R_{ki} \Delta_{ink} \tag{5.10}$$

Probability based combination algorithm uses the posterior probabilities of classifiers to carry out the combination. As in the case of class based combination they are weighted with the classifier's weights.

$$\text{Assign } x_n \rightarrow w_c \text{ if } \sum_{k=1}^{K} W_k P(w_c \mid x_{nk}) = \max_{i=1..C} \sum_{k=1}^{K} W_k P(w_i \mid x_{nk}) \tag{5.11}$$

The reliability of the classifier for the assigned class is also integrated to the combination by multiplying the reliability value with the assignment probability.

$$\text{Assign } x_n \rightarrow w_c \text{ if } \sum_{k=1}^{K} W_k R_{kc} P(w_c \mid x_{nk}) = \max_{i=1..C} \sum_{k=1}^{K} W_k R_{ki} P(w_i \mid x_{nk}) \tag{5.12}$$

The results of class based combination algorithm and probability based algorithm are combined (5.15) by first converting the class assignments of class (5.13) and probability (5.14) based combination algorithm to posterior probabilities.

$$P_c(w_i \mid x_{nk}) = \frac{\sum_{k=1}^{K} W_k \Delta_{ink}}{\sum_{j=1}^{C}\sum_{k=1}^{K} W_k \Delta_{jnk}} \tag{5.13}$$

$$P_p(w_i \mid x_{nk}) = \frac{\sum_{k=1}^{K} W_k P(w_i \mid x_{nk})}{\sum_{j=1}^{C}\sum_{k=1}^{K} W_k P(w_j \mid x_{nk})} \tag{5.14}$$

Assign $x_n \rightarrow w_c$ if

$$\sum_{k=1}^{K} P_p(w_c \mid x_{nk}) + P_c(w_c \mid x_{nk}) = \max_{i=1..C} \sum_{k=1}^{K} P_p(w_i \mid x_{nk}) + P_c(w_i \mid x_{nk}) \tag{5.15}$$

The integration of reliability is again applied in combination of combined class and probability based combination results.

$$P_{cr}(w_i \mid x_{nk}) = \frac{\sum_{k=1}^{K} W_k R_{ki} \Delta_{ink}}{\sum_{j=1}^{C}\sum_{k=1}^{K} W_k R_{kj} \Delta_{jnk}} \tag{5.16}$$

$$P_{pr}(w_i \mid x_{nk}) = \frac{\sum_{k=1}^{K} W_k R_{ki} P(w_i \mid x_{nk})}{\sum_{j=1}^{C}\sum_{k=1}^{K} W_k R_{kj} P(w_j \mid x_{nk})} \tag{5.17}$$

Assign $x_n \rightarrow w_c$ if

$$\sum_{k=1}^{K} P_{pr}(w_c \mid x_{nk}) + P_{cr}(w_c \mid x_{nk}) = \max_{i=1..C} \sum_{k=1}^{K} P_{pr}(w_i \mid x_{nk}) + P_{cr}(w_i \mid x_{nk}) \tag{5.18}$$

Different algorithms are used to calculate the weights of the classifiers for weighted combination. The easiest way is to combine the classifiers using equal weights. This performs a good result if the classifiers in the set are independent and unbiased (Alexandre *et al.* 2000). The weight of classifier m_k is:

$$W_k = \frac{1}{K} \tag{5.19}$$

A better way for assigning weights to classifiers is, to assign their performance values in the training phase. In this thesis, it is proposed that the defined overall performance values, "OP"s, found in the leave-one-out training phase are assigned as weights. These results are also integrated with the reliability of the classifiers. The weight of a classifier m_k is:

$$W_k = \frac{OP_k}{\sum_{i=1}^{K} OP_i} \tag{5.20}$$

Another proposal is to assign weights using a linear fit on the posterior probabilities of leave-one-out results. Better and reliable performance is reached with the weight assignment where the true class probabilities of the training data set and the posterior probabilities found by the classifiers are used to find a linear regression parameters for them. Least square fit parameters for the training data set is used as weights of the classifiers in the combination. The constant term which may be in the equation is not used since the shifting will not affect the relative values for classification. The weights are the solution of the following equation:

$$\begin{bmatrix} P(w_1 \mid x_1) \\ \vdots \\ P(w_1 \mid x_N) \\ P(w_2 \mid x_1) \\ \vdots \\ P(w_C \mid x_N) \end{bmatrix} = \begin{bmatrix} W_1 \\ \vdots \\ W_k \\ \vdots \\ W_K \end{bmatrix} \begin{bmatrix} P(w_1 \mid x_1,m_1) & .. & P(w_1 \mid x_1,m_2) & .. & P(w_1 \mid x_1,m_K) \\ \vdots & & \vdots & & \vdots \\ P(w_1 \mid x_N,m_1) & .. & P(w_1 \mid x_N,m_2) & .. & P(w_1 \mid x_N,m_K) \\ P(w_2 \mid x_1,m_1) & .. & P(w_2 \mid x_1,m_2) & ... & P(w_2 \mid x_1,m_K) \\ \vdots & & \vdots & & \vdots \\ P(w_C \mid x_N,m_1) & .. & P(w_C \mid x_N,m_2) & .. & P(w_C \mid x_N,m_K) \end{bmatrix} \tag{5.21}$$

The extension of this polynomial weight assignment proposal is to integrate the reliability of the classifier for the assigned class. The weights are the solution of the following equation:

$$\begin{bmatrix} P(w_1 \mid x_1) \\ \vdots \\ P(w_1 \mid x_N) \\ P(w_2 \mid x_1) \\ \vdots \\ P(w_C \mid x_N) \end{bmatrix} = \begin{bmatrix} W_1 \\ \vdots \\ W_k \\ \vdots \\ W_K \end{bmatrix} \begin{bmatrix} R(w_1 \mid m_1)P(w_1 \mid x_1, m_1) & .. & R(w_1 \mid m_K)P(w_1 \mid x_1, m_K) \\ \vdots & & \vdots \\ R(w_1 \mid m_1)P(w_1 \mid x_N, m_1) & .. & R(w_1 \mid m_K)P(w_1 \mid x_N, m_K) \\ R(w_2 \mid m_1)P(w_2 \mid x_1, m_1) & .. & R(w_2 \mid m_K)P(w_2 \mid x_1, m_K) \\ \vdots & & \vdots \\ R(w_C \mid m_1)P(w_C \mid x_N, m_1) & .. & R(w_C \mid m_K)P(w_C \mid x_N, m_K) \end{bmatrix} \quad (5.22)$$

5.3. Integrated Framework for Classifier Combination

In this thesis an integrated framework for pattern recognition is designed and developed. All the classification algorithms and combination methods are developed and coded in Matlab. The documented classification algorithms are modified to integrate them in a framework. The classifiers' input/output types and the data set's data structures are standardized. Different mapping, classification and clustering algorithms are integrated in this structure, which enables a complete analysis of data sets and the performance of different combination algorithms are tested using the classifier set integrated in this environment. The data set features, the corresponding true class assignments and if exists their real membership values for each class are stored in different files allocating a row for each sample. Finally a Windows-based user interface as seen in Figure 5.1 is developed to test and develop new classifier combination methods.

In this framework the K-means clustering and self organizing map clustering algorithms are modified to use them as classifiers, by solving an optimization problem to assign clusters to class.

$$\text{minimize} \quad \sum_{c=1}^{C}\sum_{n=1}^{N} z_{nc}x_{nc}$$

$$\text{subject to} \quad \sum_{n=1}^{N} x_{nc} = 1 \quad c = 1...C$$

$$\sum_{c=1}^{C} x_{nc} = 1 \quad n = 1...N \tag{5.23}$$

$$\text{where} \quad x_{nc} = \begin{cases} 1 \text{ if sample n is assigned to class c} \\ 0 \text{ otherwise} \end{cases}$$

$$z_{nc} = \begin{cases} 1 \text{ if sample n is wrongly assigned to class c} \\ 0 \text{ otherwise} \end{cases}$$

In this thesis, the classifiers are modified to produce class probability estimates besides their class assignments for all classes. For that purpose, their distance measures or belief values are normalized in the training set as probabilities and applied on the test set. For some classifiers such as fuzzy neural network classifier, artificial neural network classifier and support vector machine, their belief values are converted to posterior probabilities using a normalized mapping. For K-means classifier, Parzen, K-nearest neighbour, piecewise quadratic and piecewise linear distance classifiers their implicit distance measures are explicitly calculated and converted to posterior probabilities for each class. One of the main contributions of this thesis is that all the classifiers and clustering based classifier algorithms in the classifier set are modified to produce posterior probabilities for their class assignments for all classes.

For two-class data sets, the class conditioned and standardized class conditioned principle component mapping, difference of two means with equal or different covariances, generalized clustering, optimal discriminant mapping and k-nearest neighbour mapping algorithms are implemented. The multilayer perceptron, least square and principle component mappings are implemented as general two dimensional mapping techniques. The details of the classification and clustering algorithms are given in Section 6.1. To integrate the clustering algorithms as classifiers, a one-to-one assignment algorithm is applied on the final clusters.

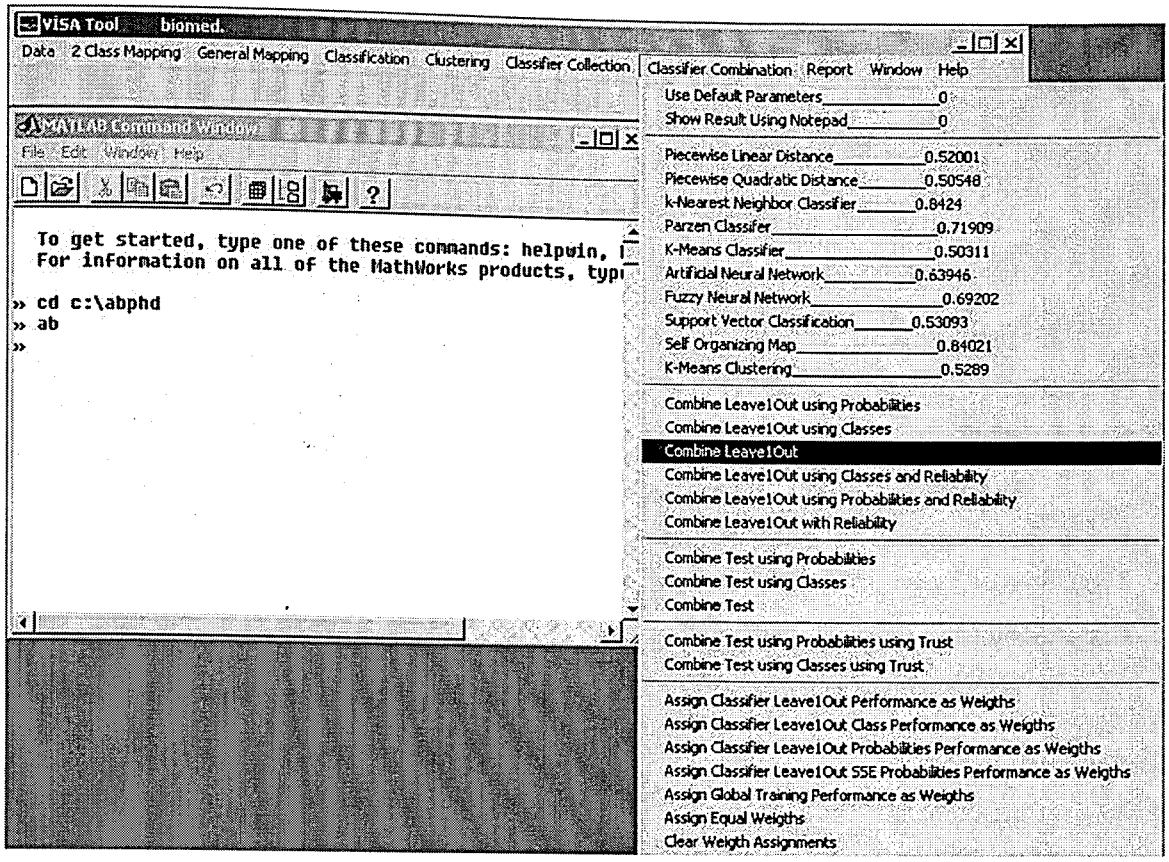


Figure 5.1. Integrated frameworks's user-interface

To analyze the data, some data visualizing techniques are also implemented and integrated in this framework. This enables us to see the behaviour of the whole training or test data in two dimensions by dimensional reduction of features. Some of the visualization techniques of the framework will be demonstrated on some of the data sets used in the experiments. For example the principle component mapping of the simulated data of two spirals can be seen in Figure 5.2.

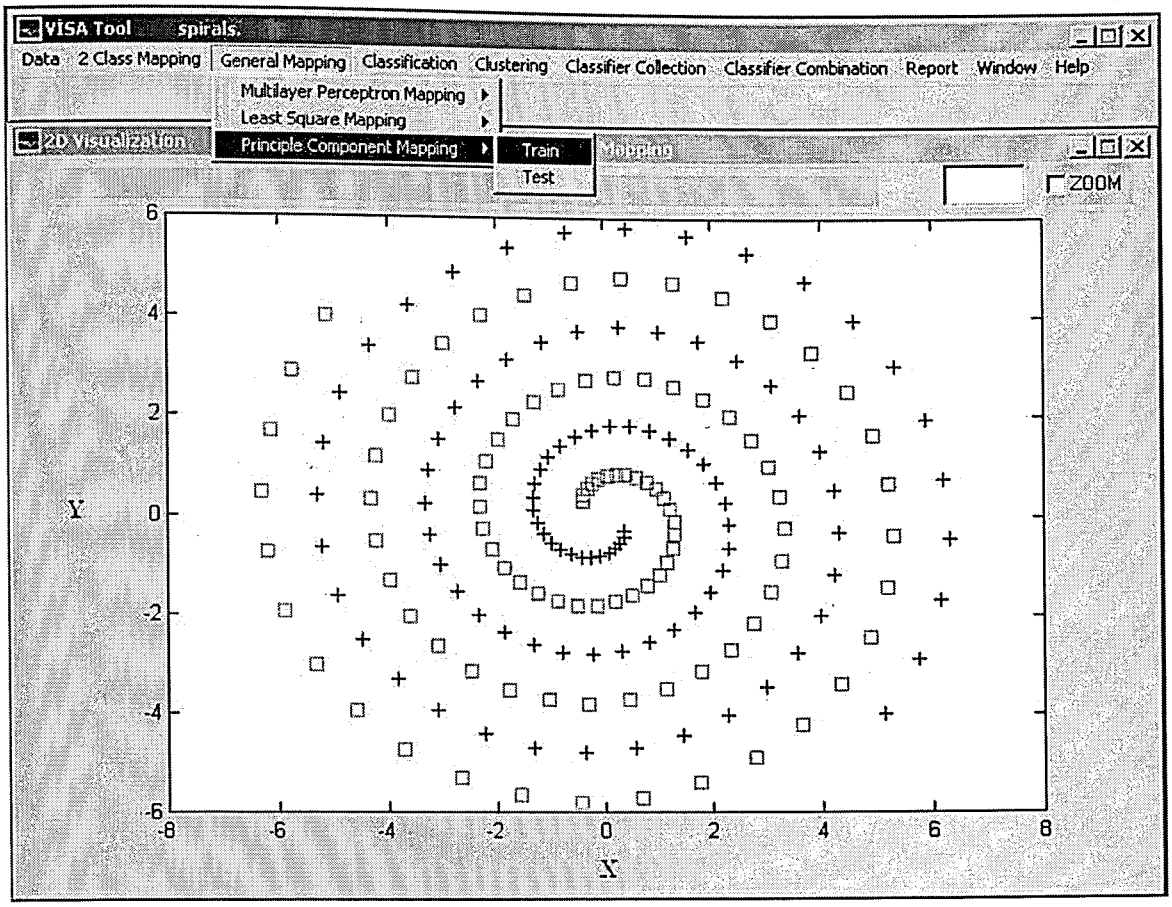


Figure 5.2. 2D Visualization of the 2SD data set

A visualization of a real-life data set is given in Figure 5.3. This is the data set used by Gorman and Sejnowski in their study of the classification of sonar signals using a neural network (Gorman and Sejnowski, 1988). The file contains 111 patterns obtained by bouncing sonar signals off a metal cylinder at various angles and under various conditions for 97 patterns obtained from rocks under similar conditions. The transmitted sonar signal is a frequency-modulated chirp, rising in frequency. The data set contains signals obtained from a variety of different aspect angles, spanning 90 degrees for the cylinder and 180 degrees for the rock. Each pattern is a set of 60 numbers in the range 0.0 to 1.0. Each number represents the energy within a particular frequency band, integrated over a certain period of time. The integration aperture for higher frequencies occur later in time, since these frequencies are transmitted later during the chirp. The label associated with each record contains the letter "R" if the object is a rock and "M" if it is a mine (metal cylinder). Figure 5.3 shows the optimal discriminant mapping of the SMR data set.

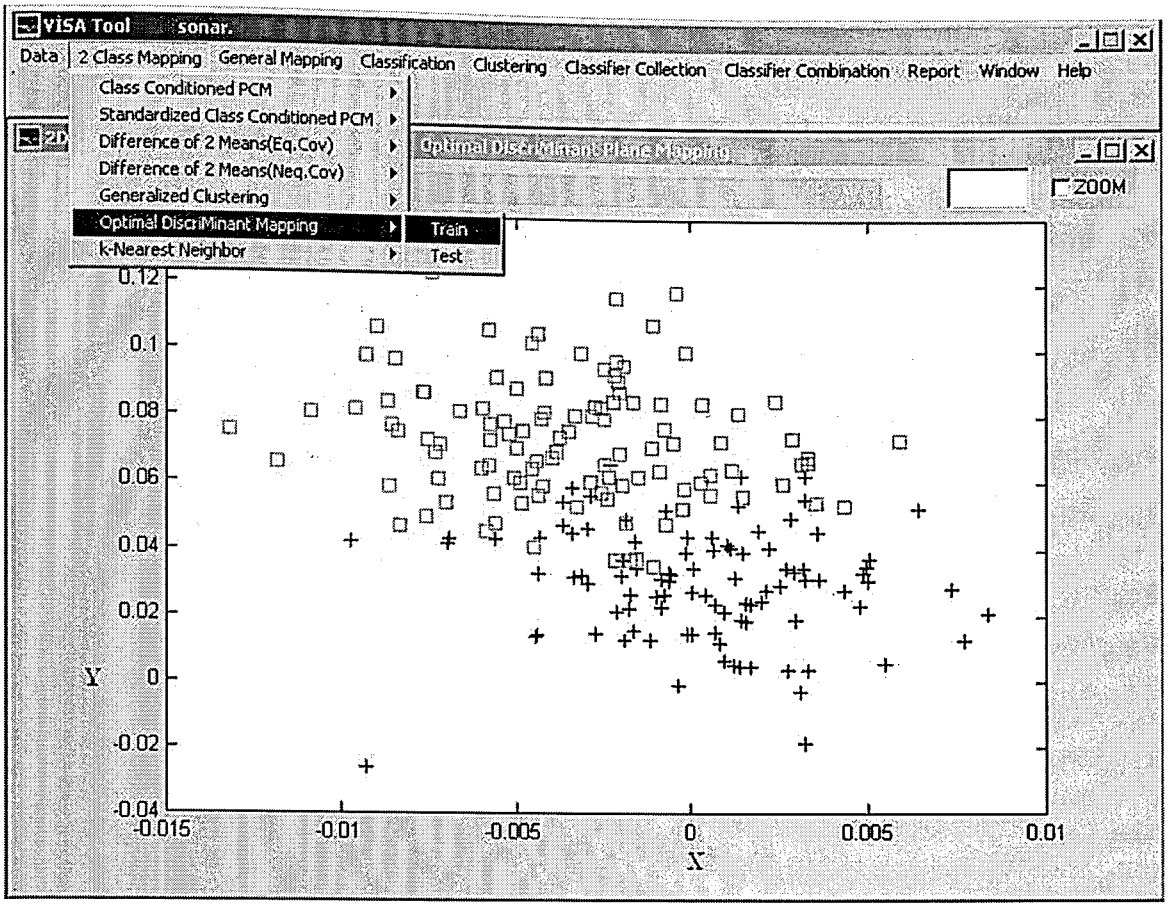


Figure 5.3. 2D Visualization of the SMR data set

The characteristics of the data set can also be visualized using the graphical feature versus feature plot techniques. The visualized feature plots in Figure 5.4 are of the GID data set, whose study of classification of types of glass was motivated by criminological investigation. At the scene of the crime, the glass left can be used as evidence, if it is correctly identified.

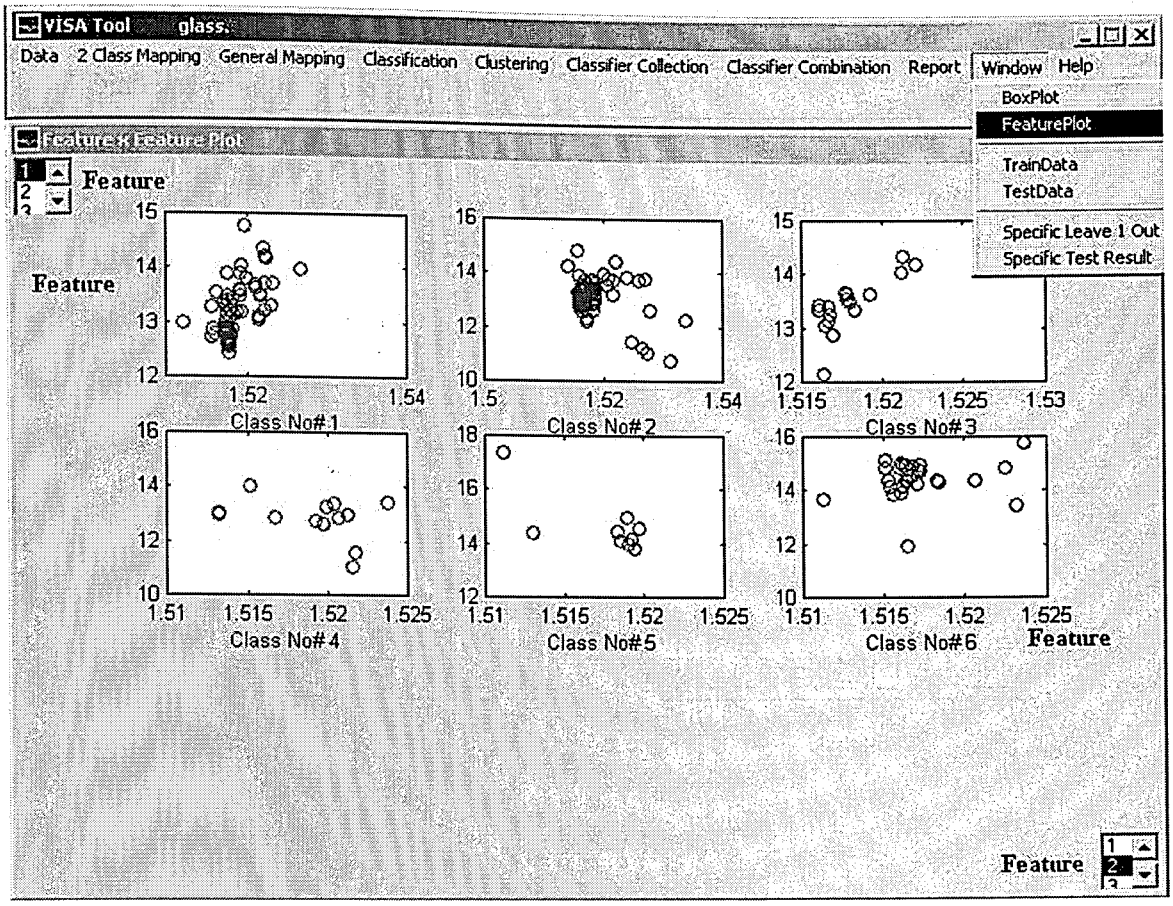


Figure 5.4. Feature x Feature plot of the GID data set

Another data set is the DIB, which is a two class data set investigating whether the patient shows signs of diabetes according to World Health Organization criteria (i.e., if the 2 hour post-load plasma glucose was at least 200 mg/dl at any survey examination or if found during routine medical care). The ADAP algorithm makes a real-valued prediction between 0 and 1 (Smith *et al.*, 1998). This was transformed into a binary decision using a cutoff of 0.448. Using 576 training instances, the correct classification performance of their algorithm was 76 per cent on the remaining 192 instances. The box plot analysis of the data set is seen in Figure 5.5. The 8 features used are defined as follows:

- Number of times pregnant
- Plasma glucose concentration a 2 hours in an oral glucose tolerance test
- Diastolic blood pressure (mm Hg)
- Triceps skin fold thickness (mm)
- 2-Hour serum insulin (mu U/ml)

- Body mass index (weight in kg/(height in m)^2)
- Diabetes pedigree function
- Age (years)

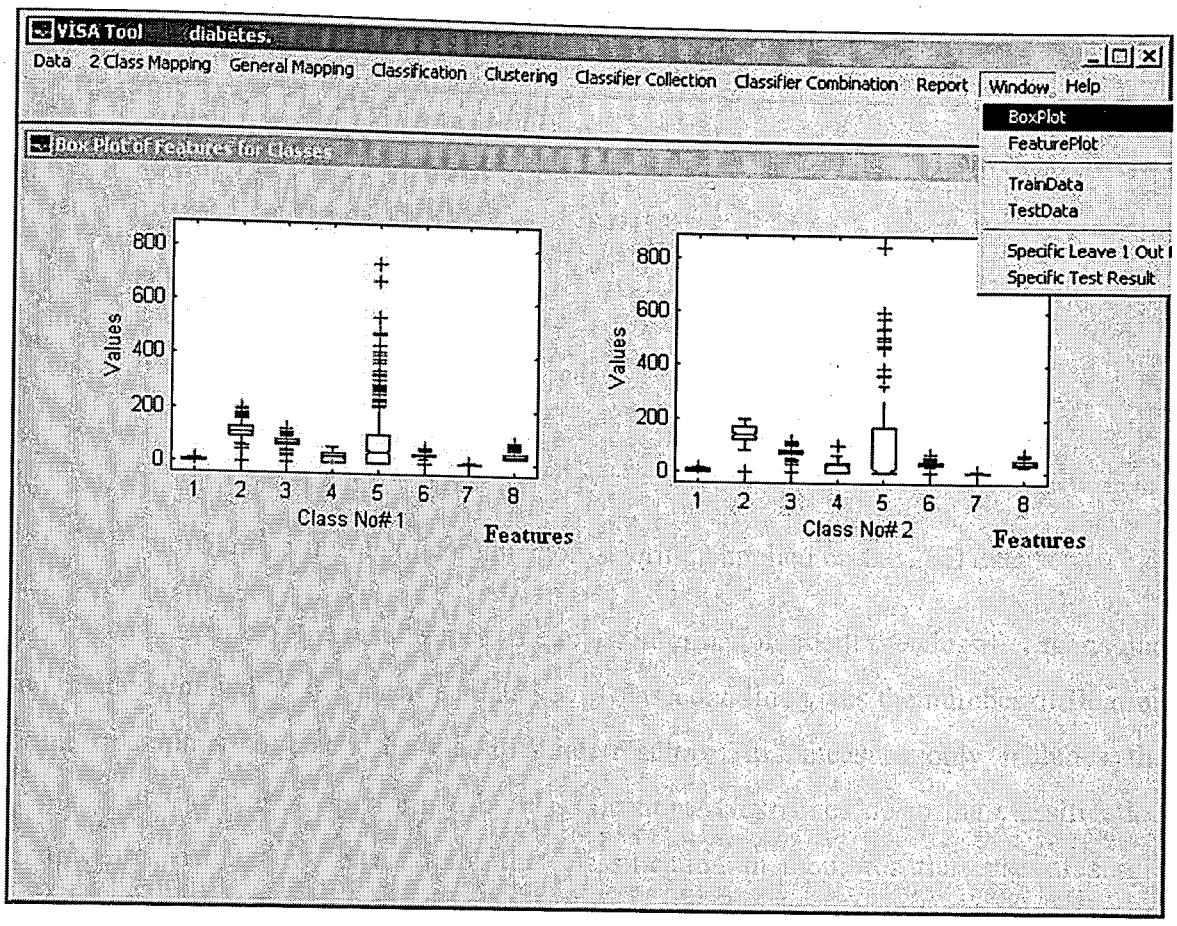


Figure 5.5. Box-plot of the DIB data set

The combination methods are collected as a separate menu item (Figure 5.1), where you can let batch long runs and later analyze the resulting reports. The results of running the algorithms including all posterior probabilities and class assignments are stored in different files. Special analysis based on the wrong classified samples is also given in separate files. All of the results are combined in a compact report for each algorithm as seen in Figure 5.6.

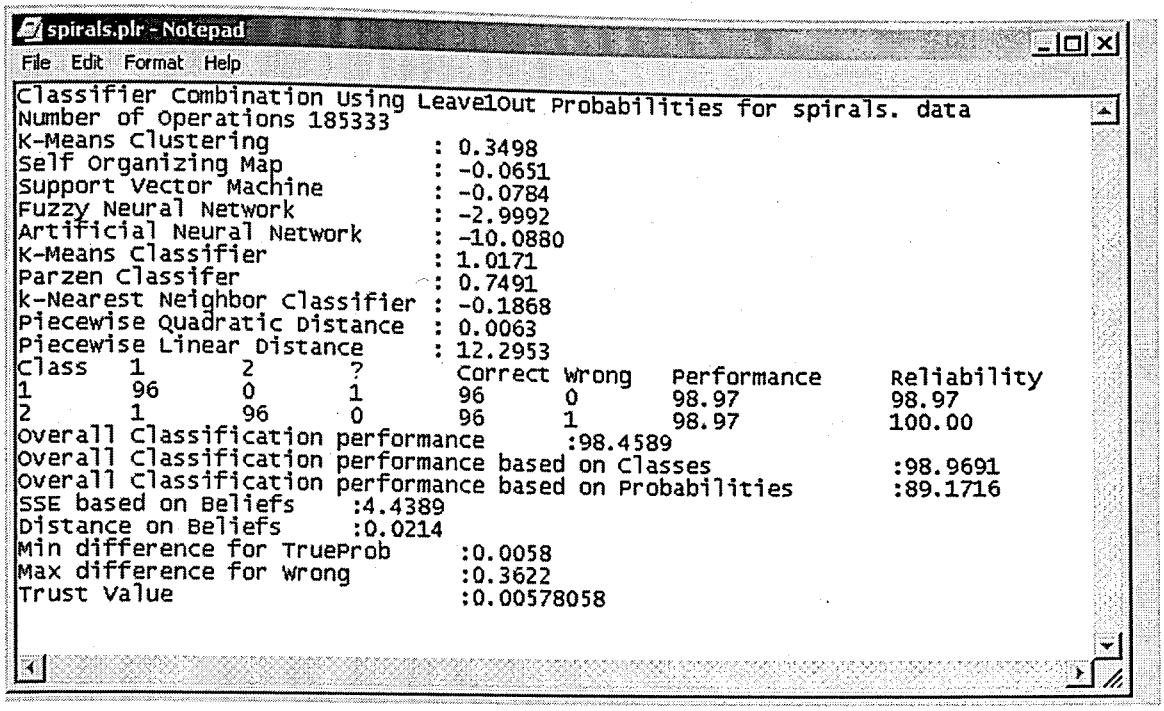


Figure 5.6. Result file of combination algorithm applied on the 2SD data set

The first line of the result report file gives information about the classifier name and the method applied on the specified data set. The second line gives the number of floating point operations to reach the classification result. This measure only includes the mathematical operations performed by classification. Next lines up to the classification matrix are the parameters used for this classification method, for the case of single classifiers, this part is used for parameters and in the case of combination, the weights of classification algorithms are given. The details of classification matrix and the definitions of the following performance measures are given in the Section 5.1.

6. NUMERICAL RESULTS

In this thesis the developed framework is used on popular data sets from the literature. The visualization techniques are applied and the analysis of the features of the data sets are done, which shows that they have different characteristics. The experiments cover the application of leave-one-out training algorithm of all classifiers on all data sets. The standard and proposed performance measures are used to evaluate the classifiers. The results cover the effects of proposed new weight assignments, integration of reliability measure in combination methods and a detailed sensitivity analysis to build a classifier set.

6.1. Classifier Set

This study focuses on the weighted combination of classification algorithms, in which the weights are optimized based on the performance in the training data set. To have a classifier combination, following classifiers with the given parameters are designed.

KMClus: K-means clustering with maximum iteration=10; maximum error=0.5.

SOM: Self organizing map clustering with iteration=1000; learning rate=1.

FANN: Fuzzy neural network classifier with fuzzification level=3; fuzzification type=0; number of hidden layer units=25; learning rate 0.001; maximum iteration=1000; minimum error=0.02.

ANN: Artificial neural network classifier with number of hidden layer units=25; learning rate 0.001; maximum iteration=1000; minimum error=0.02.

KMClas: K-means classifier.

Parzen: Parzen classifier with alfa=1.

KNN: K-nearest neighbour classifier with k=3.

PQD: Piecewise quadratic distance classifier.

PLD: Piecewise linear distance classifier.

SVM: Support vector machine using radial basis kernel with p=1

The design parameters of classifiers are chosen as typical values used in the literature or by experience. The classifiers are not specifically tuned for the data set at hand even

though they may reach a better performance with another parameter set, since the goal is to design an automated classifier combination based on any classifier in the classifier set. The aim of this study is to combine the classifiers' results in a robust way to achieve almost the best classifier's performance or better in the classifier set. In this thesis, they are modified to produce class probability estimates besides their class assignments for all classes. For that purpose their distance measures or belief values are normalized in the training set as probabilities and applied on the test set. All of the classification algorithms are applied on the data sets using the leave-one-out technique of training. For the clustering, standard k-means clustering and self-organizing map algorithms are applied on the data in a regular way. In this thesis, after the final step of clustering a one-to-one assignment algorithm is applied on the final clusters of the leave-one-out technique. The resulting mapping criteria of clusters to classes are then applied on the test sample.

6.2. Data Sets

In this thesis popular data sets from the literature are worked on. There is no special choice on the characteristics of the data sets; as can be seen later in the results, different classifiers achieve different performance on different data sets as expected. This result agrees with the fact that there is no single classifier, which performs best on different characteristics of input data. Different classification algorithms and combination methods are applied on the data sets with the following properties:

BIO: Data from cariers and non cariers of a rare genetic disorder. 5 inputs, 2 outputs (127+67) 194 case.

DIB: Pigma Indians Diabetes Database. 8 inputs, 2 outputs (500+268) 768 case.

D10: Two class data set, with Duin 10 dimensional distribution. 10 inputs, 2 outputs (100+100) 200 case.

GID: Glass Identification Databas. 9 inputs, 6 outputs (70+76+17+13+9+29) 214 case.

IMX: IEEE data file of letters I, M, O, X. 8 inputs, 4 outputs (48+48+48+48) 192 case.

SMR: Sonar data set. 60 inputs, 2 outputs (97+111) 208 case.

2SD: Two spirals two dimensional data set. 2 inputs, 2 outputs (97+97) 194 case.

WQD: Wine data set. 13 inputs, 3 outputs (59+71+48) 178 case.

80X: IEEE 80X data set. 8 inputs, 3 outputs (15+15+15) 45 case.

ZMM: Data set of 6 Zernike moments of 8 characters. 6 inputs, 8 outputs (12+12+12+12+12+12+12+12) 96 case.

BEM: Two class data set, with equal mean but different variance(20 per cent Bayes error). 2 inputs, 2 outputs (100+100) 200 case.

BEV: Two class data set, with different mean but equal variance (20 per cent Bayes error). 2 inputs, 2 outputs (100+100) 200 case.

HRD: Highleyman distributed random patterns. 2 inputs, 2 outputs (100+100) 200 case.

IFD: Classical data set of Fisher with 150 iris flowers. 4 inputs, 3 outputs (50+50+50) 150 case.

6.3. Data Handling for Multiple Classifiers

Achieving optimal performance for a pattern recognition system is not necessarily consistent with obtaining the best performance for a single classifier. Under appropriate assumptions, combining multiple classifiers may lead to improved generalization performance when compared with any one of the constituent classifiers. However certain conditions need to be satisfied to realize the performance improvement, in particular that the individual classifiers should not be highly correlated. Various methods have been devised to reduce the correlation between classifiers before combining. For unstable classifiers, such as neural networks, a small perturbation in the training set may lead to a significant change in the constructed classifier, even if identical feature representations are used.

Having limited amount of training data at hand forces to optimize the usage of the training data. There are methods like crossvalidation, bootstrapping and variations of them to attack the problem of limited training data. Crossvalidation is a classifier evaluation method that is better than residual analysis, since residual evaluations do not give an indication of how well the classifier will do when it is asked to make new classification for data it has not already seen. One way to overcome this problem is to not use the entire data

set when training a classifier. Some of the data is removed before training begins. Then when training is done, the data that was removed can be used to test the performance of the classifier on new data. This is the basic idea for a whole class of classifier evaluation methods called crossvalidation. The great advantage of crossvalidation is that all the cases in the available sample are used for testing, and almost all the cases are also used for training the classifier.

The holdout method is the simplest kind of crossvalidation. The data set is separated into two sets, called the training set and the testing set. The classifier is trained using the training set only. Then the classifier is asked to classify the data in the testing set. The errors it makes are accumulated as before to give the mean absolute test set error, which is used to evaluate the classifier. The advantage of this method is that it is usually preferable to the residual method and takes no longer to compute. However, its evaluation can have a high variance. The evaluation may depend heavily on which data points end up in the training set and which end up in the test set, and thus the evaluation may be significantly different depending on how the division is made.

K-fold crossvalidation is one way to improve over the holdout method. In k-fold cross-validation, the cases are randomly divided into k mutually exclusive test partitions of approximately equal size. The holdout method is repeated k times. Each time, one of the k subsets is used as the test set and the other k-1 subsets are put together to form a training set. Then the average error across all k trials is computed. The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once, and gets to be in a training set k-1 times. The variance of the resulting estimate is reduced as k is increased. The disadvantage of this method is that the training algorithm has to be rerun from scratch k times, which means it takes k times as much computation to make an evaluation. A variant of this method is to randomly divide the data into a test and training set k different times. The advantage of doing this is that you can independently choose how large each test set is and how many trials you average over.

Leave-one-out crossvalidation is K-fold crossvalidation taken to its logical extreme, with K equal to N, the number of data points in the set. That means that N separate times, the classifier is trained on all the data except for one sample and a classification is made for

that single test sample. As before the average error is computed and used to evaluate the classifier.

Leave-one-out is an elegant and straightforward technique for estimating classifier error rates. Because it is computationally expensive, it has often been reserved for problems where relatively small sample sizes are available. For a given method and sample size, N, a classifier is generated using (N-l) cases and tested on the single remaining case. This is repeated N times, each time designing a classifier by leave-one-out. Thus, each case in the sample is used as a test case, and each time nearly all the cases are used to design a classifier. The error rate is the number of errors on the single test cases divided by N.

Traditionally a small statistical sample has been considered to be around 30 or fewer cases. For many years, leave-one-out was the recommended technique for evaluating classifier performance on small samples, and its use was confined to them. This was mostly due to the computational costs for applying leave-one-out to larger samples. Because leave-one-out error rate estimators are virtually unbiased, the leave-one-out error rate estimator can be applied to much larger samples, yielding accurate results (Shao, 1993).

Although the leave-one-out error rate estimator is an almost unbiased estimator of the true error rate of a classifier, there are difficulties with this technique. Both the bias and variance of an error rate estimator contribute to the inaccuracy and imprecision of the error rate estimate. While leave-one-out is nearly unbiased, its variance is high for small samples. Recall that unbiased means that the estimator will, over the long run, average to the true error rate. The leave-one-out estimate also has a high variance for small samples.

The variance effect tends to dominate in small samples. Thus a low variance estimate that may even be somewhat biased has the potential of being superior to the leave-one-out approach on small samples. While at one time leave-one-out was considered computationally expensive, available computational power has increased dramatically over the years, and the accuracy of estimation can now become the overriding criterion of evaluation.

6.4. Classifier Results

The classifier set is applied on the data sets and their operation time in terms of mathematical operations are given in Table 6.1. This measure depends on the number of samples of the data, on the length of the feature vector of the data and the classification algorithm itself.

In Table 6.1 the comparison of algorithms applied on data sets is based on the floating-point operations "FLOPS" performed. This measure does not count the screen-display and file operations. To give an idea for real time equivalent of the floating point operations, the support vector machine with radial basis kernel applied on the diabetes, "DIB", data set takes 22 days of time on a pentium IV1.6 GHz processing power and 512 MB memory. The fuzzy neural network, artificial neural network and support vector machines are the most time costly algorithms, whereas k-means and k-nearest neighbour classifiers are the most effective ones.

Table 6.1. Time performance of classifiers

FLOPS	BIO	DIB	D10	GID	IMX	SMR	2SD	
KMClus	1.8E+07	5.1E+08	9.2E+06	5.8E+07	3.4E+07	2.5E+07	3.3E+06	**Medium**
SOM	3.1E+07	2.4E+08	5.7E+07	1.4E+08	7.9E+07	3.2E+08	1.7E+07	**Medium**
FANN	4.3E+10	8.6E+11	6.5E+10	1.0E+11	6.5E+10	2.9E+11	3.1E+10	**High**
ANN	4.3E+10	8.5E+11	6.5E+10	1.0E+11	6.5E+10	2.9E+11	3.1E+10	**High**
KMClass	1.8E+06	4.4E+07	3.7E+06	1.0E+07	5.1E+06	2.3E+07	7.7E+05	**Low**
Parzen	2.5E+08	2.3E+10	5.2E+08	5.8E+08	3.7E+08	3.9E+09	1.1E+08	**High**
kNN	7.6E+05	1.9E+07	1.6E+06	1.7E+06	1.2E+06	1.0E+07	3.1E+05	**Low**
PQD	8.9E+06	3.0E+08	3.2E+07	7.3E+07	3.4E+07	1.5E+09	2.4E+06	**Medium**
PLD	1.3E+07	4.6E+08	4.7E+07	1.1E+08	5.2E+07	1.8E+09	3.0E+06	**Medium**
SVM	8.3E+10	7.6E+13	6.8E+11	7.2E+12	1.7E+11	2.1E+12	1.6E+12	**High**
FLOPS	WQD	80X	ZMM	BEM	BEV	HRD	IFD	
KMClus	5.6E+07	1.2E+06	3.5E+06	2.7E+06	3.5E+06	2.7E+06	8.4E+06	**Medium**
SOM	8.8E+07	1.3E+07	5.2E+07	1.7E+07	1.7E+07	1.7E+07	2.7E+07	**Medium**
FANN	6.6E+10	3.2E+09	2.0E+10	3.3E+10	3.3E+10	3.3E+10	2.7E+10	**High**
ANN	6.6E+10	3.2E+09	2.0E+10	3.3E+10	3.3E+10	3.3E+10	2.7E+10	**High**
KMClass	5.4E+06	2.2E+05	1.9E+06	8.1E+05	8.1E+05	8.1E+05	1.2E+06	**Low**
Parzen	4.8E+08	5.2E+06	3.7E+07	1.2E+08	1.2E+08	1.2E+08	9.3E+07	**High**
kNN	1.7E+06	6.7E+04	2.4E+05	3.3E+05	3.3E+05	3.3E+05	3.7E+05	**Low**
PQD	5.8E+07	1.8E+06	1.0E+07	2.5E+06	2.5E+06	2.5E+06	5.2E+06	**Medium**
PLD	8.6E+07	2.3E+06	1.5E+07	3.2E+06	3.2E+06	3.2E+06	7.4E+06	**Medium**
SVM	9.2E+10	1.2E+08	1.6E+11	1.8E+12	1.4E+12	1.4E+12	3.1E+10	**High**

Classifiers' performances are given in Table 6.2, Table 6.3 and Table 6.4. In these tables the best performances for the data sets are marked as bold face. The number of the marked items in the last column, titled "Best" indicates the performance among classifiers, that is the number of times the classifier has outperformed the others for 14 data sets.

In Table 6.2 the results show that k-nearest neighbour classifier has the best result seven times out of 14 different data sets. All classifiers show different performance on different data set, for example the k-nearest neighbour classifier, which is the best classifier based on Table 6.2, has at least 10 per cent lower performance compared with other classifiers in some data sets like DIB, D10, WQD, BEM and HRD.

Table 6.2. Class performance of classifiers

CP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
KMClus	71.1	65.9	69.0	50.5	81.8	48.1	42.3	51.7	28.9	34.4	59.0	48.5	73.0	90.0	0
SOM	84.0	64.8	75.5	45.3	72.4	54.3	37.1	84.3	71.1	52.1	45.5	87.0	**84.0**	89.3	1
FANN	70.6	74.7	70.0	36.0	68.8	74.5	36.6	93.3	28.9	8.3	34.0	86.0	81.5	71.3	0
ANN	**87.6**	**76.8**	**78.0**	50.5	86.5	77.9	45.9	96.6	82.2	66.7	53.0	85.5	81.5	82.7	3
KMClas	75.3	46.1	76.0	33.2	88.5	65.4	47.4	62.9	**93.3**	69.8	48.5	74.5	82.5	91.3	1
Parzen	58.8	62.2	21.0	28.5	15.6	24.5	57.7	32.6	35.6	63.5	47.5	82.0	59.5	76.0	0
kNN	84.5	67.6	69.0	**73.4**	**94.3**	**82.2**	**76.3**	76.4	**93.3**	**88.5**	70.0	**92.0**	74.5	95.3	7
PQD	13.9	72.8	72.5	1.9	**94.3**	75.0	43.3	**99.4**	88.9	26.0	**79.5**	65.0	82.0	92.0	3
PLD	84.5	75.7	77.0	57.5	90.6	72.6	47.4	98.3	88.9	85.4	56.5	84.0	81.5	**97.3**	1
SVM	6.2	7.6	70.5	37.4	**94.3**	71.2	26.8	25.8	91.1	36.5	73.5	86.0	80.5	94.7	1

In Table 6.3 the results indicate that the k-nearest neighbour classifier is more successful than the case in Table 6.2; 10 times out of 14 different data sets, k-nearest neighbour classifier outperforms the others. This improvement can be explained by the fact that the artificial neural networks' probability performance is lower compared to its class performance. For example for the DIB data set, the class performance of 76.8 per cent, which is higher than the k-nearest neighbour classifier's 67.6 per cent, drops to 58.3 per cent, which is lower than the k-nearest neighbour classifier's 67.4 per cent probability performance.

Table 6.3. Probability performance of classifiers

PP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
KMClus	52.9	50.6	53.1	72.6	64.7	49.2	50.9	57.1	50.6	78.3	49.2	49.3	54.5	60.9	0
SOM	84.0	64.8	**75.5**	81.8	86.2	54.3	37.1	**89.5**	80.7	88.0	45.5	87.0	**84.0**	92.9	3
FANN	69.2	64.3	60.3	77.0	68.4	63.1	44.7	72.5	54.1	77.7	45.6	60.8	66.8	70.0	0
ANN	63.9	58.3	59.3	75.6	69.0	60.6	50.6	70.9	64.8	79.5	50.2	58.6	63.0	65.4	0
KMClas	50.3	50.1	54.2	72.6	65.1	51.4	52.2	58.4	60.1	78.5	49.9	51.0	58.6	60.9	0
Parzen	71.9	66.2	52.7	76.6	65.5	58.2	71.4	60.6	63.0	84.8	55.8	81.6	65.7	74.5	0
kNN	**84.2**	**67.4**	67.0	**90.6**	**97.3**	**81.8**	**77.9**	82.6	**96.2**	**97.2**	69.0	**91.9**	75.8	**97.0**	10
PQD	50.5	50.9	53.7	73.1	64.7	51.9	51.2	57.9	58.9	78.2	53.3	52.0	57.8	58.7	0
PLD	52.0	50.9	56.2	72.2	64.4	52.7	51.7	59.2	60.8	79.4	49.3	53.1	64.0	64.8	0
SVC	53.1	52.8	71.3	82.4	97.3	76.0	29.9	66.8	95.6	86.1	**75.9**	87.2	82.5	96.9	1

The decrease of performance can be explained by the nature of the training data set, since the training data set for supervised learning only have the class values given. The probabilities are in fact a mapping of class information to probabilities as zero or one, so the artificial neural network does not have the real probability information to train. The class and probability performance measures have differences based on the training data set supervised information. The results with overall performance measure are given in Table 6.4.

Table 6.4. Overall performance of classifiers

OP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
KMClus	51.4	44.0	48.8	29.7	69.2	23.4	18.2	31.9	26.8	21.3	35.3	24.1	54.6	82.4	0
SOM	71.3	43.7	57.1	23.6	54.6	29.6	13.8	71.8	52.7	31.7	20.7	76.7	**70.6**	81.0	1
FANN	52.1	56.0	50.2	13.8	51.3	55.9	13.5	88.1	8.9	1.1	11.7	75.2	66.8	55.7	0
ANN	**78.0**	**59.2**	**61.1**	27.2	76.4	61.6	21.3	94.5	69.6	48.7	28.2	75.1	67.8	70.9	3
KMClas	64.0	28.7	58.3	20.6	79.2	43.3	23.7	51.7	89.2	53.9	26.3	56.1	68.4	85.9	0
Parzen	52.3	42.5	15.2	14.0	15.6	24.5	57.7	16.7	35.6	59.3	30.9	75.7	44.0	74.2	0
kNN	71.9	46.8	48.4	**56.4**	90.4	**67.9**	**58.8**	59.6	**91.3**	**80.7**	49.3	**85.5**	56.7	91.6	6
PQD	13.9	54.2	53.6	1.4	**90.9**	57.1	19.8	**99.4**	82.7	23.9	**64.0**	65.0	69.0	91.3	3
PLD	74.0	58.4	59.6	36.9	83.4	54.8	22.7	97.2	84.8	75.7	32.1	71.9	67.4	**95.4**	1
SVM	6.2	6.1	50.5	26.8	90.0	53.9	7.8	24.8	89.0	26.5	58.8	74.7	67.5	91.5	0

For the sake of this study, all classifiers are like experts with different backgrounds, who try to conclude the class of the sample at hand via decision combination with an acceptable reliability. As stated before, the classifiers are not especially tuned for the training set. The results show that k-nearest neighbour classifier outperforms the others for

this classifier set with the current setting of parameters for individual classifiers. Another point is that the k-means clustering and Parzen classifiers are the worst ones. The results also validate that the data sets have different characteristics, since their performance are different for different classifiers. Based on the classifier set with current parameter settings, the data sets DIB, D10, GID, SMR and 2SD are relatively hard to classify.

6.5. Boosting a Classifier Using Combination

Different combination methods can be used on a single classifier, if there are parameters to tune for better performance. Artificial neural networks with different hidden layer structure, k-nearest neighbour with different k values and support vector machines with different kernels are examples of classifiers where this technique can be applied.

In Table 6.5, the results of applying the k-nearest neighbour classifier with leave-one-out training algorithm on the data sets with k=1,3,5 are given. The results for each data set is grouped for different performance measures: overall "OP", class "CP" and probability "PP" performance. The results confirm that the data sets have different characteristics, since there is no single k value to reach the best performance for k-nearest neigbour classifier. The column with the "Best" title gives the number of times the particular classifier on that row is best out of 14 data sets for the set of k-nearest neighbour classifiers with k=1,3,5.

Table 6.5. Performance of k-nearest neighbour classifier with k=1,3,5

OP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
K1	72.4	47.2	48.3	56.2	90.1	68.4	58.2	**59.9**	87.4	**81.0**	49.0	85.6	56.3	**92.2**	3
K3	71.9	46.8	**48.4**	**56.4**	90.4	67.9	58.8	59.6	**91.3**	80.7	49.3	85.5	56.7	91.6	3
K5	**75.4**	**49.2**	47.4	54.3	**91.9**	**68.8**	**83.2**	55.9	89.2	78.0	**52.7**	**87.0**	**65.9**	**92.2**	9
CP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
K1	85.1	68.0	**69.5**	**73.4**	**94.8**	**82.7**	76.3	**77.0**	**93.3**	**89.6**	70.0	92.5	75.0	**96.0**	8
K3	84.5	67.6	69.0	**73.4**	94.3	82.2	76.3	76.4	**93.3**	88.5	70.0	92.0	74.5	95.3	2
K5	**86.6**	**69.5**	68.5	72.0	**94.8**	81.7	**90.7**	73.6	**93.3**	87.5	**72.0**	**93.0**	**81.0**	95.3	8
PP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
K1	**85.1**	**68.0**	**69.5**	**91.1**	**97.4**	**82.7**	76.3	**84.6**	95.6	**97.4**	70.0	**92.5**	75.0	**97.3**	10
K3	84.2	67.4	67.0	90.6	97.3	81.8	**77.9**	82.6	**96.2**	97.2	69.0	91.9	75.8	97.0	2
K5	83.3	66.9	66.0	89.4	97.0	81.2	75.8	80.3	93.1	96.2	**70.2**	90.7	**75.9**	97.0	2

Table 6.6 and Table 6.7, cover the results of the boosting of the k-nearest neigbour classifier for k=1,3,5 neighbours. The performance of the combination is marked as bold face, if it is better than the best performance of the k-nearest neigbour classifier for k=1,3,5 neighbours. The results of class based combination (5.9 and 5.10) are given in Table 6.6 and probability based combination (5.11 and 5.12) results are in Table 6.7.

Results are grouped for three different performance measures. The first block is for overall performance for class based combination, "OP". Second block is for class performance, "CP", and the last block is for probability performance for class based combination, "PP". Results of different weight assignments are given in rows. Simple averaging results are given in rows with "Equal" title (5.19) and the results next to "Equal" rows, titled "xReliability", tabulate the integration of reliability for combination. For the weight assignment titled "Performance", overall performance values "OP"s are assigned as weights (5.20). The results integrated with the reliability of the classifiers are given in the rows next to "Performance" rows. The last rows are for weight assignment titled "Polynomial", where linear regression values are assigned as weights (5.21) and reliabilities are integrated in the last row.

The class based combination results' class performance does not improve well compared to probability performance. This is due to the characteristic of the k-nearest neighbour classifier. Higher k-values include the same training points of the lower k-values which affect the classification result. For example k-nearest neighbour classifier with k=5, has two more training samples added to the set of points which decide on the class of the test sample, where k=3. Thus depending on the distribution of the training samples, going up to higher k-values increases the possibility of including training samples for wrong classes. In that case since the distances of the later added training samples are higher than the previously added samples, their converted posterior probabilities are low. Although there are some improvements as seen in the "Best" column, which show the number of better results for combination compared to the best result obtained by k-nearest neighbour classifier. In fact, the result show that the 10 cases out of 14 data sets have better results based on the probability performance measure. This means that although the class performance is not improved, the true probability assignments for test samples

improve. Comparing the result rows for weighted combination and reliability integrated weighted combination, although the increase in performance is on the average less than 0.2 per cent, the integration of reliability always improves the combination performance.

Table 6.6. Class based combination of boosting a classifier

OP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	73.7	45.3	45.9	53.5	90.5	68.3	68.8	56.3	87.4	79.6	50.1	85.9	59.1	**92.2**	1
xReliability	73.0	46.9	46.6	**56.9**	90.5	67.9	68.8	59.6	87.4	79.9	49.3	85.9	57.6	**92.2**	2
Performance	73.7	46.8	**48.4**	53.5	90.5	68.3	68.8	59.6	87.4	79.6	50.6	85.9	57.1	**92.2**	2
xReliability	73.0	46.9	46.6	**56.9**	90.5	67.9	68.8	59.6	87.4	79.9	49.3	85.9	57.6	**92.2**	2
Polynomial	**75.9**	**49.2**	47.4	54.4	**91.9**	**68.8**	69.7	56.0	**91.3**	79.6	**53.2**	86.6	**65.9**	**92.8**	8
xReliability	**75.9**	**49.2**	47.4	54.1	**91.9**	**68.8**	69.7	56.0	**91.3**	79.6	**53.2**	86.6	**65.9**	**92.8**	8
CP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	84.5	62.4	64.0	68.7	**94.8**	81.2	76.3	70.8	**93.3**	87.5	68.0	91.5	72.5	**96.0**	3
xReliability	84.5	67.7	66.0	**73.8**	**94.8**	82.2	76.3	76.4	**93.3**	88.5	69.5	91.5	75.5	**96.0**	2
Performance	84.5	67.6	69.0	68.7	**94.8**	81.2	76.3	76.4	**93.3**	87.5	68.5	91.5	75.0	**96.0**	3
xReliability	84.5	67.7	66.0	**73.8**	**94.8**	82.2	76.3	76.4	**93.3**	88.5	69.5	91.5	75.5	**96.0**	4
Polynomial	**87.1**	**69.5**	68.5	72.9	**94.8**	81.7	76.3	74.2	**93.3**	87.5	**72.5**	**93.0**	**81.0**	**96.0**	8
xReliability	**87.1**	**69.5**	68.5	72.4	**94.8**	81.7	76.3	74.2	**93.3**	87.5	**72.5**	**93.0**	**81.0**	**96.0**	8
PP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	**85.7**	**68.5**	69.4	**91.1**	**97.5**	**82.8**	**81.4**	84.1	95.6	97.3	**71.3**	**92.7**	**77.2**	**97.3**	10
xReliability	**85.8**	**68.6**	69.4	**91.2**	**97.5**	**82.8**	**82.1**	84.1	95.6	97.3	**71.3**	**92.7**	**77.4**	**97.3**	10
Performance	**85.8**	**68.5**	69.4	**91.1**	**97.5**	**82.8**	**82.7**	84.2	95.6	97.3	**71.4**	**92.7**	**77.5**	**97.3**	10
xReliability	**85.8**	**68.6**	69.4	**91.2**	**97.5**	**82.8**	**83.4**	84.1	95.6	97.3	**71.4**	**92.7**	**77.6**	**97.3**	10
Polynomial	**87.1**	**69.6**	68.9	91.0	**97.7**	**83.3**	**87.8**	83.3	**96.5**	97.3	**72.9**	**92.9**	**81.2**	**97.5**	10
xReliability	**87.1**	**69.6**	68.9	90.9	**97.7**	**83.3**	**87.6**	83.3	**96.5**	97.3	**72.9**	**92.9**	**81.2**	**97.5**	10

The probability based combination results in Table 6.7 show a different characteristic compared to class based combination of Table 6.6. The probability performance could not be improved. This due to the fact that k-nearest neighbour already uses its distance measures as probabilities in the classification, the best possible result based on probabilities is reached.

Assuming the true class assignment is the crucial performance measure, the results in Table 6.6 and Table 6.7, show an improvement, if the polynomial weight assignment with or without the applied reliability is used for probability based or class based combination of the classifiers.

Table 6.7. Probability based combination of boosting a classifier

OP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	71.9	47.2	47.9	56.3	90.9	**68.9**	58.2	**59.3**	87.4	79.9	48.6	85.5	56.3	**92.2**	3
xReliability	71.9	47.2	47.9	**57.4**	90.9	**68.9**	58.2	**59.4**	89.2	79.9	48.6	85.5	56.6	**92.2**	4
Performance	71.9	47.2	47.9	56.3	90.9	**68.9**	58.2	**59.3**	87.4	79.9	48.6	85.5	56.3	**92.2**	3
xReliability	71.9	47.2	47.9	**57.4**	90.9	**68.9**	58.2	**59.4**	89.2	79.9	48.6	85.5	56.6	**92.2**	4
Polynomial	74.8	48.2	47.7	55.3	**92.5**	**72.4**	**84.0**	**58.9**	**93.5**	78.7	**56.2**	**87.9**	**66.8**	**92.2**	9
xReliability	74.8	48.2	47.5	55.6	**92.5**	**71.1**	**84.0**	60.2	**93.5**	79.6	**56.7**	**87.9**	**66.8**	**92.2**	8

CP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	84.5	67.8	69.0	72.9	94.3	82.2	76.3	76.4	**93.3**	88.5	69.5	92.0	74.5	**96.0**	2
xReliability	84.5	67.8	69.0	**74.3**	94.3	82.2	76.3	76.4	**93.3**	88.5	69.5	92.0	74.5	**96.0**	3
Performance	84.5	67.8	69.0	72.9	94.3	82.2	76.3	76.4	**93.3**	88.5	69.5	92.0	74.5	**96.0**	2
xReliability	84.5	67.8	69.0	**74.3**	94.3	82.2	76.3	76.4	**93.3**	88.5	69.5	92.0	74.5	**96.0**	3
Polynomial	85.6	68.8	67.5	72.9	**95.8**	**83.7**	**90.7**	75.8	**95.6**	87.5	**74.5**	**93.5**	**81.5**	95.3	7
xReliability	85.6	68.8	67.5	72.4	**95.8**	**84.1**	**90.7**	**77.0**	**95.6**	87.5	**75.0**	**93.5**	**81.5**	95.3	8

PP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	84.2	67.4	67.5	90.4	97.2	81.9	76.7	82.5	94.9	96.9	69.7	91.7	75.6	97.1	0
xReliability	84.2	67.9	67.5	90.5	97.2	81.9	76.6	82.3	94.9	97.0	69.6	91.6	75.6	97.1	0
Performance	84.2	67.4	67.5	90.4	97.2	81.9	76.6	82.6	95.0	96.9	69.7	91.7	75.6	97.1	0
xReliability	84.2	67.9	67.5	90.5	97.2	81.9	76.5	82.4	94.9	97.0	69.6	91.6	75.6	97.1	0
Polynomial	83.2	66.7	66.3	89.7	97.0	81.1	**81.6**	81.4	96.0	96.8	**71.2**	91.0	**76.1**	97.2	3
xReliability	83.2	66.7	66.3	89.7	96.9	81.1	**82.4**	81.4	95.4	96.8	**71.3**	91.0	**76.1**	97.2	3

The same idea of combining classifiers with different parameters can be applied to improve the learning performance of a classifier. The learning performance is classifier's ability to learn the training data set. That is the performance of classification of the whole training data set, after training the classifier with the whole training data set. The combination algorithms are applied on the support vector machine with eight different kernels. The following kernels are used in the support vector machine kernel collection.

linear	linear
poly3	third degree polynomial
rbf	radial basis with unit width
erbf	radial basis with a unit width and square root of distance calculation
sigmoid	sigmoid with scale one and no offset
fourier	fourier with zero degree
spline	spline
bspline	third degree bspline.

The individual learning performances are given in rows in the following tables. Table 6.8, Table 6.9 and Table 6.10 tabulate the results of applying these support vector machine kernels on the data sets. The results are grouped for different performance measures: class performances are given in Table 6.8, probability performances are given in Table 6.9 and overall performances are given in Table 6.10. The column with the "Best" title, is the number of cases of best learning result for the 14 data set the classifier applied.

The results indicate that the radial basis and bspline kernels have the best learning performance among these kernels. In Table 6.8, Table 6.9 and Table 6.10, the radial basis kernels outperforms the other kernels on the average 5 times out of 14 data sets and on the other hand, the bspline kernels outperforms 8 times out of 14 data sets.

Table 6.8. Class performance of SVMs' learning

CP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	14
linear	30.9	32.8	66.0	61.2	24.5	56.7	50.5	27.0	31.1	22.6	42.5	63.0	**82.5**	10.1	1
poly 3	30.2	44.5	74.0	56.7	25.0	55.8	60.8	27.0	33.3	43.8	33.5	55.5	82.0	28.4	0
rbf	**40.5**	48.6	99.0	**80.1**	99.5	77.9	72.2	**100.0**	**100.0**	53.1	74.0	**87.5**	81.0	**100.0**	6
erbf	40.2	**55.5**	89.5	78.4	91.7	**84.6**	84.5	**100.0**	**100.0**	**67.7**	61.0	85.5	**82.5**	99.3	6
sigmoid	28.8	43.2	23.7	55.7	40.9	53.4	43.1	39.9	28.7	18.8	37.0	50.0	**82.5**	0.0	1
fourier	15.7	32.9	27.3	28.0	33.1	43.4	36.1	25.1	8.9	12.2	25.0	17.0	16.5	6.0	0
spline	29.2	30.8	60.5	53.7	25.0	54.3	32.0	59.6	52.8	33.5	37.5	40.5	**82.5**	23.9	1
bspline	39.7	44.8	**100.0**	76.6	**100.0**	76.0	**100.0**	44.9	**100.0**	29.4	**75.0**	74.5	81.0	**100.0**	6

Table 6.9. Probability performance of SVMs' learning

PP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	14
linear	**49.0**	54.5	77.0	71.4	70.0	68.7	50.5	62.2	65.3	78.8	58.8	75.4	82.5	53.0	1
poly 3	40.9	45.5	86.9	74.8	72.4	77.9	60.8	62.4	66.7	79.0	65.2	73.2	82.9	61.9	0
rbf	43.0	47.1	99.5	83.1	99.8	83.8	72.2	**100.0**	**100.0**	**85.6**	77.0	84.4	83.2	**100.0**	4
erbf	41.6	**58.4**	92.2	82.4	96.8	**87.2**	84.5	**100.0**	**100.0**	82.2	70.8	**87.2**	82.4	99.5	5
sigmoid	30.5	27.2	34.8	61.3	62.5	53.4	50.0	55.6	55.6	80.8	49.6	59.4	82.6	55.6	0
fourier	20.5	26.7	17.9	38.4	31.2	47.2	36.9	50.9	50.4	78.1	45.2	38.0	17.0	40.6	0
spline	35.3	31.5	74.5	64.6	68.8	77.2	50.3	82.0	63.0	78.7	45.0	53.0	83.2	29.7	0
bspline	43.0	54.3	**100.0**	**83.9**	**100.0**	86.7	**100.0**	74.6	**100.0**	84.5	**79.8**	87.1	**83.8**	**100.0**	8

Table 6.10. Overall performance of SVMs' learning

OP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
linear	23.8	31.3	56.1	48.0	8.5	42.1	25.5	10.8	14.5	11.1	28.3	53.4	68.1	24.2	0
poly 3	26.3	42.9	74.0	50.6	10.3	55.8	37.0	10.9	16.7	31.2	30.8	47.9	67.9	30.8	0
rbf	36.1	48.8	99.0	68.1	99.5	**76.9**	52.1	**100.0**	**100.0**	42.3	59.5	77.2	68.7	**100.0**	4
erbf	**37.7**	**50.5**	85.2	67.8	91.2	75.3	71.5	**100.0**	**100.0**	**58.6**	46.2	**79.1**	70.5	98.7	6
sigmoid	21.6	15.0	22.0	36.8	32.3	28.5	22.7	19.9	13.2	10.4	19.5	31.1	68.1	69.9	0
fourier	6.6	2.6	13.5	11.9	10.4	28.5	13.0	10.8	21.8	8.0	11.4	5.0	2.8	1.4	0
spline	23.5	21.0	51.1	40.5	8.3	54.3	16.2	59.6	40.0	12.2	16.5	22.2	69.1	13.8	0
bspline	37.2	45.1	**100.0**	**72.7**	**100.0**	76.0	**100.0**	36.9	**100.0**	27.7	**70.1**	74.0	**70.6**	**100.0**	8

The results of different combinations can be summarized as in Table 6.11 and in Table 6.12. The performance of the combination is marked as bold face, if it is better than the best learning performance of kernels applied. Results are grouped for three different performance measures. The first block is for overall performance for class based combination, "OP". The second block is for class performance, "CP", and the last block is for probability performance for class based combination, "PP". Results of different weight assignments are given in rows. Simple averaging results are given in rows with "Equal" title (5.19) and the results next to "Equal" rows, titled "xReliability", tabulate the integration of reliability for combination. For the weight assignment titled "Performance", overall performance values "OP"s are assigned as weights (5.20). The results integrated with the reliability of the classifiers are given in the rows next to "Performance" rows. The last rows are for weight assignment titled "Polynomial", where linear regression values are assigned as weights (5.21) and reliabilities are integrated in the last row.

In the learning case, the results show that, especially the polynomial weight assignment always improves the performance. The "Best" column indicates this result, which is the number of times the combination algorithm is better than the best kernel based on the same performance measure.

Table 6.11. Class based combination for learning

OP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	33.3	42.7	81.1	54.1	64.6	59.0	55.0	68.2	93.3	42.2	60.6	66.4	69.0	72.9	0
xReliability	37.1	49.6	90.5	66.0	**100.0**	72.2	73.9	**100.0**	**100.0**	54.6	64.6	**81.9**	67.9	96.4	4
Performance	**38.3**	**51.2**	87.7	**73.0**	**100.0**	73.7	74.9	**100.0**	**100.0**	54.5	62.5	**79.7**	67.9	88.9	7
xReliability	34.0	48.6	95.8	68.7	**100.0**	**76.9**	83.9	**100.0**	**100.0**	55.5	65.3	**80.2**	67.9	91.3	5
Polynomial	37.1	**53.8**	92.2	71.6	**100.0**	**92.8**	**100.0**	**100.0**	**100.0**	56.5	**83.7**	**93.6**	**74.3**	**100.0**	10
xReliability	**45.2**	**54.0**	96.2	**75.9**	**100.0**	**92.3**	**100.0**	**100.0**	**100.0**	56.9	**83.2**	**91.6**	**75.1**	**100.0**	12
CP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	34.6	47.8	87.2	62.2	64.6	72.6	63.4	72.5	93.3	52.1	69.5	74.0	82.0	84.4	0
xReliability	**43.6**	54.9	90.3	75.9	**100.0**	84.1	85.1	**100.0**	**100.0**	66.7	**78.5**	**89.5**	82.0	96.8	6
Performance	40.3	**62.1**	**104.4**	**80.7**	**100.0**	**85.1**	86.1	**100.0**	**100.0**	**67.7**	**77.5**	**88.5**	82.0	88.9	10
xReliability	**46.4**	**59.8**	97.5	74.0	**100.0**	**87.0**	89.7	**100.0**	**100.0**	**67.7**	**79.5**	**89.0**	82.0	94.0	9
Polynomial	**46.7**	**61.5**	98.3	**82.8**	**100.0**	**92.8**	**100.0**	**100.0**	**100.0**	**67.7**	**91.0**	**95.5**	**85.5**	**100.0**	13
xReliability	**43.2**	**56.3**	**111.1**	**84.4**	**100.0**	**92.3**	**100.0**	**100.0**	**100.0**	**67.7**	**90.5**	**93.5**	**86.0**	**100.0**	14
PP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	41.7	47.7	87.2	80.9	86.4	73.4	65.6	83.1	86.8	**89.0**	66.4	75.4	75.0	83.8	1
xReliability	**52.3**	**62.0**	90.7	79.8	93.6	78.2	72.2	91.3	93.3	**91.3**	71.3	80.9	81.4	86.0	3
Performance	**50.1**	**63.3**	90.4	77.8	98.0	79.8	78.6	96.7	97.7	**91.5**	73.5	84.7	82.9	88.9	3
xReliability	**53.7**	**64.5**	91.8	**87.0**	99.2	83.2	83.3	98.2	98.9	**91.8**	76.3	86.6	83.2	96.4	4
Polynomial	48.9	**63.9**	**100.0**	**94.9**	**100.0**	**96.3**	**100.0**	95.6	97.6	**92.2**	**87.0**	**94.3**	82.7	**100.0**	10
xReliability	**53.9**	**59.8**	**100.0**	**85.5**	**100.0**	**95.8**	**100.0**	97.6	98.9	**93.1**	**87.6**	**94.3**	**83.8**	**100.0**	12

Table 6.12. Probability based combination for learning

OP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	32.9	42.1	81.4	56.1	63.0	59.6	55.0	73.6	93.3	41.5	63.7	70.3	69.3	70.0	0
xReliability	**40.2**	46.6	94.6	68.3	**100.0**	71.4	73.9	**100.0**	**100.0**	51.4	69.7	**81.2**	68.3	84.0	5
Performance	**40.9**	48.8	93.3	71.5	**100.0**	71.4	74.9	**100.0**	**100.0**	55.9	64.5	**83.2**	68.3	83.8	5
xReliability	**42.0**	**52.4**	88.1	63.3	**100.0**	**78.5**	83.9	**100.0**	**100.0**	52.6	67.0	**84.1**	68.6	96.7	7
Polynomial	**38.5**	47.0	**100.0**	69.7	**100.0**	**97.1**	**100.0**	**100.0**	**100.0**	**63.1**	**85.2**	**95.1**	**76.4**	93.3	11
xReliability	**42.4**	**51.3**	**100.0**	**76.9**	**100.0**	**93.8**	**100.0**	**100.0**	**100.0**	**66.2**	**84.7**	**94.1**	**76.9**	**96.1**	13
CP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	34.7	42.4	80.0	69.4	69.3	75.0	63.4	78.7	93.3	58.3	74.0	80.0	82.0	80.8	0
xReliability	37.6	**56.0**	98.3	**84.8**	**100.0**	83.7	85.1	**100.0**	**100.0**	**68.8**	**83.0**	**89.5**	82.0	92.2	8
Performance	**41.1**	**60.1**	**100.0**	**82.4**	**100.0**	83.7	86.1	**100.0**	**100.0**	**68.8**	**78.0**	**90.5**	82.0	99.0	10
xReliability	**40.9**	**57.4**	**100.0**	**82.8**	**100.0**	**88.0**	89.7	**100.0**	**100.0**	**68.8**	**80.5**	**91.0**	82.0	94.4	11
Polynomial	**48.6**	**62.6**	**100.0**	**84.6**	**100.0**	**97.6**	**100.0**	**100.0**	**100.0**	**72.1**	**92.0**	**97.0**	**86.5**	98.7	13
xReliability	**41.6**	**64.1**	**100.0**	**82.8**	**100.0**	**95.7**	**100.0**	**100.0**	**100.0**	**74.8**	**91.5**	**96.5**	**87.5**	96.5	13
PP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	43.0	55.1	78.2	65.8	78.9	73.5	63.0	73.4	75.1	82.7	62.0	70.2	75.1	79.7	0
xReliability	**50.8**	**60.0**	84.5	71.5	90.0	76.8	71.4	83.1	89.8	**89.0**	66.8	75.5	81.6	91.5	3
Performance	48.5	55.5	90.8	75.6	96.5	77.8	78.1	91.0	96.6	**89.7**	69.7	79.3	83.0	99.5	1
xReliability	**54.8**	**61.3**	**100.0**	78.8	98.0	80.0	82.9	92.4	98.3	**90.4**	73.1	81.2	83.3	94.3	4
Polynomial	48.8	**60.7**	**100.0**	81.4	**100.0**	**92.0**	**100.0**	95.6	95.6	**89.7**	**83.1**	**88.0**	78.3	**100.0**	9
xReliability	**49.8**	**64.5**	94.5	**91.5**	**100.0**	**94.1**	**100.0**	95.9	96.6	**88.1**	77.4	**91.5**	**84.2**	**100.0**	10

Implementing leave-one-out for support vector machines will obviously decrease performances. This is intuitive, since removing any training sample that constructs a support vector for generalization on the hyperplanes found by the algorithm, will degrade

the classification performance. If the removed training sample is not a support vector, the performance will not be affected. If the data sets were linearly separable, there will be only C(C-1)/2 support vectors for C classes, which is the minimum number of support vectors possible. Considering the nature of the real data sets, they are not linearly separable, thus the number of support vectors found by the algorithm increases and in the worst case every sample becomes part of the support vector and the performance will decrease. This can be validated by cross checking Table 6.2, where leave-one-out result of the support vector machine with radial basis kernel for class performance measure are given in the last row, titled "SVM", with Table 6.8, where the learning class performance of the same kernel is given in the "rbf" row. For example, the learning performance 77.9 per cent for the SMR data set drops to 71.2 per cent. Similarly cross checks of Table 6.3 with Table 6.9, and of Table 6.4 with Table 6.10 indicate the same conclusion.

6.6. Classifier Combination Results

As the first combination algorithm, the defined different weight assignment algorithms are used on class based classifier combination, where the classifiers have produced only the class assignment results. The results for this combination can be summarized as in Table 6.13. The performance of the combination is marked as bold face, if it is better than the average of the top three best performance of the classifier set in Table 6.2. Results are grouped for three different performance measures. The first block is for overall performance for class based combination, "OP". Second block is for class performance, "CP", and the last block is for probability performance for class based combination, "PP". Results of different weight assignments are given in rows. Simple averaging results are given in rows with "Equal" title (5.19) and the results next to "Equal" rows, titled "xReliability", tabulate the integration of reliability for combination. For the weight assignment titled "Performance" overall performance values "OP"s are assigned as weights (5.20). The results integrated with the reliability of the classifiers are given in the rows next to "Performance" rows. The last rows are for weight assignment titled "Polynomial", where linear regression values are assigned as weights (5.21) and reliabilities are integrated in the last row.

The results in Table 6.13 of the simple averaging are not so promising as expected, since the classifiers in the set are not specifically chosen to be independent and unbiased. The "Best" column indicates the number of times the combination algorithm performs better than the average of the top three classifier. The results show that even these performance can be improved using the classifier reliability for the class assignments. For the class based combination algorithm, the integration of the reliability of the classifiers always improves the performance of the combination. The results show that the performance assignment as weights is better than the other methods of combination weight assignments.

Table 6.13. Class based classifier combination

OP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	**77.0**	52.7	54.3	33.2	85.1	61.6	21.0	93.3	87.0	60.8	39.0	77.9	69.0	91.5	1
xReliability	**77.4**	54.9	58.1	**41.1**	86.7	**66.3**	24.6	96.1	**93.5**	69.2	48.7	**80.3**	68.4	91.5	5
Performance	**78.9**	56.7	58.8	**47.1**	86.7	**66.6**	39.3	96.7	**93.5**	**75.5**	54.6	78.7	68.8	90.9	5
xReliability	**79.6**	55.1	59.1	**51.6**	87.6	**67.9**	**59.6**	**97.2**	**93.5**	**75.5**	**58.0**	**79.3**	68.8	91.5	9
Polynomial	60.8	56.1	**60.8**	34.7	**90.9**	**65.9**	46.7	93.4	66.0	65.9	**58.8**	54.9	55.8	88.0	4
xReliability	**77.7**	57.0	**61.6**	35.7	90.1	62.1	43.8	90.7	73.3	70.1	47.6	74.3	67.9	87.9	2
CP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	85.6	68.0	69.5	53.7	90.6	72.1	45.4	94.9	91.1	71.9	56.0	86.5	80.5	94.0	0
xReliability	**87.6**	74.0	76.0	**62.6**	92.7	**80.8**	49.5	97.8	**95.6**	**82.3**	68.0	**89.0**	82.5	94.7	6
Performance	**87.6**	75.1	76.5	**67.3**	92.7	**81.3**	**62.4**	97.8	**95.6**	**85.4**	72.5	88.0	82.5	94.7	6
xReliability	**87.1**	74.1	76.5	**71.0**	92.7	**81.3**	**76.8**	**98.3**	**95.6**	**85.4**	**75.5**	88.0	82.5	94.7	8
Polynomial	76.3	74.3	**77.0**	57.0	**94.8**	**80.8**	**61.9**	96.1	80.0	79.2	**75.5**	66.0	73.5	93.3	5
xReliability	**86.1**	75.1	**77.5**	57.5	**94.8**	75.5	47.4	94.9	84.4	**82.3**	67.5	86.0	80.5	93.3	4
PP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	**82.6**	**70.4**	**73.6**	84.2	93.2	70.4	49.3	**88.3**	85.1	**90.8**	59.4	83.2	80.7	94.2	5
xReliability	**83.3**	**70.9**	**73.9**	**86.1**	**93.9**	72.4	55.4	**91.9**	**92.6**	**93.9**	62.9	85.1	**80.9**	94.9	9
Performance	**83.2**	**71.1**	**74.0**	**85.6**	**94.0**	73.2	59.6	**92.8**	**91.9**	**94.0**	65.3	86.0	**81.1**	94.8	9
xReliability	**83.8**	**71.6**	**74.2**	**87.4**	**94.6**	**74.5**	65.7	**94.6**	**94.6**	**95.1**	**68.1**	86.9	**81.3**	95.4	11
Polynomial	76.1	**74.8**	**77.8**	**86.0**	**96.8**	**81.6**	**67.7**	**96.1**	78.9	**94.3**	**76.8**	76.0	73.8	94.4	9
xReliability	**87.0**	**74.6**	**78.2**	84.5	**97.0**	**79.5**	**71.6**	**90.8**	84.5	**95.4**	**68.6**	83.2	78.9	94.9	9

In Table 6.14 the classifiers' class probability estimates are used to combine using all the weight assignment algorithms. The integration of the reliability improves the performance. The results compared to the class based combination ones show us a different behaviour, the "Polynomial" weight assignment is better than the others. This can be explained by the discrete type of function values of the class based combination compared to continuous type of function values for polynomial weight assignment.

Table 6.14 Probability based classifier combination

OP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	**80.0**	54.5	55.3	**42.1**	89.0	**64.4**	39.1	77.1	**93.5**	65.4	48.5	**80.3**	**71.3**	91.6	6
xReliability	**78.0**	51.4	55.6	**52.8**	89.4	**67.6**	**56.2**	84.9	**93.5**	71.9	**58.5**	**79.7**	**71.3**	91.6	8
Performance	**79.5**	54.3	55.8	**53.2**	89.9	**68.9**	**62.5**	82.6	**91.3**	**76.2**	**61.3**	**80.3**	**70.9**	91.6	9
xReliability	**78.5**	51.0	55.8	**54.3**	89.9	**71.0**	**66.5**	84.5	**91.3**	**78.1**	**61.0**	78.7	**70.9**	91.6	8
Polynomial	**81.8**	**59.5**	**61.2**	**56.7**	**90.8**	**72.0**	**98.5**	90.8	**93.5**	**81.7**	**64.7**	**87.4**	**70.1**	92.2	12
xReliability	**79.1**	**58.1**	**61.4**	**56.0**	**90.8**	**73.2**	**67.8**	87.6	**97.8**	**81.6**	**60.8**	**87.9**	**70.4**	92.2	12

CP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	**89.2**	73.6	74.0	**63.1**	93.8	**79.3**	**62.4**	87.1	**95.6**	79.2	69.0	**89.0**	**84.0**	95.3	7
xReliability	**87.1**	71.5	74.0	**71.5**	93.8	**81.7**	**74.7**	91.0	**95.6**	**83.3**	**76.0**	88.0	**84.0**	95.3	8
Performance	**88.7**	73.4	74.5	**71.0**	93.8	**82.2**	**78.9**	90.4	**93.3**	**86.5**	**77.5**	**89.0**	**84.0**	95.3	9
xReliability	**88.1**	71.4	74.5	**72.9**	93.8	**83.7**	**80.9**	91.6	**93.3**	**87.5**	**77.5**	88.0	**84.0**	95.3	8
Polynomial	**90.2**	**77.1**	**78.0**	**74.3**	**94.3**	**84.6**	**99.0**	94.9	**95.6**	**89.6**	**79.0**	**93.0**	**83.5**	95.3	12
xReliability	**88.7**	**76.2**	**78.0**	**72.9**	**94.3**	**85.1**	**70.6**	93.3	**97.8**	**89.6**	**76.5**	**93.5**	**83.5**	95.3	12

PP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	63.2	57.6	60.4	77.5	74.3	59.9	51.8	67.6	68.5	82.8	54.4	67.2	67.3	74.2	0
xReliability	63.6	58.9	60.3	79.7	75.2	60.9	56.5	68.3	72.0	84.1	56.1	68.3	67.3	74.3	0
Performance	65.8	58.2	60.8	79.3	75.5	61.8	60.2	68.4	72.2	84.4	57.4	69.8	67.5	74.6	0
xReliability	65.5	59.4	60.8	82.0	76.0	62.6	64.7	68.5	73.6	85.1	58.9	70.3	67.6	74.7	0
Polynomial	**81.0**	**67.5**	68.3	**86.6**	**96.1**	**77.7**	**92.5**	**86.4**	**95.0**	**94.6**	**67.9**	88.3	75.0	94.4	10
xReliability	76.6	**70.7**	69.7	84.9	**96.5**	**85.1**	**83.8**	**86.4**	89.8	**97.3**	**76.2**	**92.4**	74.4	94.7	8

Based on the class performance, the proposed polynomial weight assignment for weighted classifier combination outperforms the other combination methods. Comparing Table 6.14 and Table 6.2, except for the data sets WQD, BEM and IFD, the combination performance is even higher than the best classifier's performance. For example for the DIB data set, the best possible class performance is 76.8 per cent, which is increased to 77.1 per cent.

Finally both the class and probability based combination results are combined in Table 6.15. This results shows the same characteristic; that is the integration of reliability of classifiers increase the performance and the "Polynomial" weight assignment is better than the other weight assignments.

Table 6.15. Performance of combined classifier combination

OP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	**78.1**	55.8	54.3	37.6	86.2	**66.5**	24.4	91.8	**91.3**	64.8	43.3	**80.3**	**69.5**	90.9	5
xReliability	**78.1**	53.5	54.8	**43.7**	87.2	**67.6**	45.8	93.9	**93.5**	**72.4**	48.9	**80.3**	**69.5**	91.5	7
Performance	**81.0**	54.9	56.5	**49.6**	86.8	**69.4**	**62.0**	95.0	**93.5**	**77.4**	**57.8**	**80.3**	**69.5**	90.9	9
xReliability	**80.1**	55.2	56.4	**54.7**	90.4	**69.9**	**62.0**	96.7	**93.5**	**76.5**	**60.1**	**80.2**	69.1	92.7	8
Polynomial	66.9	57.8	**60.5**	**48.3**	90.4	**70.7**	**81.0**	93.4	83.0	**80.6**	**64.0**	56.9	56.9	90.1	6
xReliability	**80.4**	**59.1**	**61.5**	**43.2**	90.4	**64.3**	44.8	91.8	84.9	**75.3**	51.0	78.7	69.3	**93.5**	7

CP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	**87.6**	74.6	73.5	59.8	91.7	**80.8**	49.0	95.5	**93.3**	78.1	65.5	**89.0**	82.5	94.7	4
xReliability	**87.6**	73.0	73.5	**64.5**	92.7	**81.7**	**67.5**	96.1	**95.6**	**82.3**	69.5	**89.0**	82.5	94.7	7
Performance	**89.7**	74.0	75.0	**69.2**	92.7	**82.7**	**78.4**	97.2	**95.6**	**86.5**	**75.5**	**89.0**	82.5	94.7	8
xReliability	**89.2**	74.2	74.5	**72.4**	**94.3**	**83.2**	**78.4**	97.8	**95.6**	**86.5**	**77.0**	**88.5**	82.5	94.7	9
Polynomial	80.4	**75.8**	**77.0**	**67.8**	**94.3**	**83.7**	**82.5**	96.1	88.9	**88.5**	**78.5**	66.0	74.5	92.7	8
xReliability	**89.2**	**76.7**	**78.0**	**64.0**	**94.3**	75.0	44.8	95.5	91.1	**82.3**	64.0	86.0	**83.0**	**96.0**	8

PP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	72.9	64.0	67.0	80.8	83.7	65.2	50.6	77.9	76.8	86.8	56.9	75.2	74.0	84.2	0
xReliability	73.4	64.9	67.1	82.9	84.5	66.6	56.0	80.1	82.3	89.0	59.5	76.7	74.1	84.6	0
Performance	74.5	64.7	67.4	82.4	84.8	67.5	59.9	80.6	82.0	89.2	61.4	77.9	74.3	84.7	0
xReliability	74.6	65.5	67.5	84.7	85.3	68.6	65.2	**81.6**	84.1	90.1	63.5	78.6	74.4	85.0	1
Polynomial	78.4	**71.2**	**73.0**	**86.3**	**96.4**	**79.7**	**80.1**	**91.2**	87.0	**94.5**	**72.3**	79.1	74.4	94.4	9
xReliability	**81.8**	**72.7**	**74.0**	84.7	**96.7**	**80.7**	**72.4**	**88.6**	87.1	**96.3**	**72.7**	87.8	76.7	94.8	9

6.7. Sensitivity of Combination for Classifier Set

The classifier set is detailed in Section 6.1. In this thesis, it is aimed to include all kind of classifiers in the set. The basic idea behind selection of a classifier for the classifier set is that, the individual classifier which will be added in the set should not be strongly correlated in the misclassification of the current classifiers in the classifier set. This criteria is not easily satisfied, since even different classes of the same data set may have different characteristics. That is, the classifier's performance may vary for the different classes of the same data set. A basic sensitivity analysis is done by removing the worst classifiers and best classifier of the classifier set.

The worst classifiers based on results of Table 6.2, Table 6.3 and Table 6.4 are k-means clustering, self-organizing map clustering based classification algorithms and the k-means classifier. The analysis of removing the clustering algorithms are given in Table 6.16 through Table 6.21. The performance of the combination is marked as bold face, if it is better than the average of the top three best performance of the classifier set in Table 6.2. Results are grouped for three different performance measures. The first block is for overall

performance for class based combination, "OP". Second block is for class performance, "CP", and the last block is for probability performance for class based combination, "PP". Results of different weight assignments are given in rows. Simple averaging results are given in rows with "Equal" title (5.19) and the results next to "Equal" rows, titled "xReliability", tabulate the integration of reliability for combination. For the weight assignment, titled "Performance", overall performance values "OP"s are assigned as weights (5.20). The results integrated with the reliability of the classifiers are given in the rows next to "Performance" rows. The last rows are for weight assignment titled "Polynomial", where linear regression values are assigned as weights (5.21) and reliabilities are integrated in the last row. The "Best" column indicates the number of times the combination algorithm performs better than the average of the top three classifiers. The difference tables tabulate the performance differences between two tables as stated in their titles. A positive value indicates a performance improvement comparing the same weight assignment for the sama data set. Their "Best" column indicates the number of times the new classifier set perform better than the original set.

In Table 6.16, class based classifier combination results, without the k-means clustering and self-organizing map classification algorithms are tabulated. In Table 6.17, the difference of the performance values between the new classifier set (Table 6.16) and the whole classifier set (Table 6.13) is taken. Based on these tables, the improvement especially on the probability performance is obvious. The difference table's "Best" column show an improvement for almost all data sets.

In Table 6.18, probability based classifier combination results, without the k-means clustering and self-organizing map classification algorithms are tabulated in the same manner. In Table 6.19, the difference of the performance values between the new classifier subset (Table 6.18) and the original classifier set with all the classifiers (Table 6.14) are given. As in the case of class based classifier combination the performance improves.

Table 6.16. Class based classifier combination (no clustering)

OP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	**76.8**	53.7	55.6	33.9	86.3	**64.5**	20.8	92.7	88.9	67.4	34.9	78.2	67.8	92.7	2
xReliability	**78.3**	56.2	55.9	40.8	88.0	**69.1**	25.1	**97.2**	**91.3**	**76.5**	50.7	78.2	67.8	92.7	5
Performance	**77.8**	56.3	56.9	**47.2**	88.0	**68.6**	**48.6**	**97.8**	89.0	**76.4**	51.5	77.9	68.6	92.7	6
xReliability	**78.1**	56.4	57.9	**52.2**	88.9	**69.1**	**58.7**	**98.3**	**91.3**	**76.5**	53.2	**80.6**	68.6	92.2	8
Polynomial	**78.4**	**59.2**	**61.1**	36.7	**90.9**	56.6	**55.1**	91.8	72.5	69.9	**65.0**	68.1	68.4	**93.5**	7
xReliability	**75.2**	**60.0**	**60.7**	**56.0**	90.1	46.4	45.9	89.6	74.3	**72.9**	55.0	**85.3**	67.0	**93.5**	7
CP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	85.1	68.4	71.5	53.3	90.6	74.0	44.8	93.8	88.9	76.0	51.0	86.5	81.5	94.0	0
xReliability	**88.1**	74.6	74.5	**62.1**	92.7	**82.7**	50.0	**98.3**	**93.3**	**86.5**	71.0	87.5	81.5	94.0	6
Performance	**87.6**	74.9	75.0	**66.4**	92.7	**82.2**	**62.9**	**98.3**	91.1	**85.4**	71.5	87.5	82.0	94.0	6
xReliability	**87.6**	74.9	75.5	**71.0**	93.2	**82.7**	**71.6**	**98.9**	**93.3**	**86.5**	72.5	88.0	82.0	95.3	7
Polynomial	**87.6**	**76.7**	**78.0**	59.8	**94.8**	75.0	**69.1**	94.9	82.2	**82.3**	**80.0**	73.5	82.5	**96.0**	8
xReliability	**85.6**	**77.2**	76.5	**72.9**	**94.8**	67.8	46.9	93.8	82.2	81.2	73.0	**90.5**	81.0	**96.7**	6
PP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	**84.5**	**71.9**	**73.9**	84.5	**94.5**	**75.7**	51.8	**91.3**	89.9	**92.2**	61.3	87.0	**81.1**	94.5	8
xReliability	**84.9**	**72.2**	**74.1**	**86.6**	**95.1**	**76.2**	58.3	**93.7**	**93.7**	**94.8**	65.0	87.4	**81.2**	95.3	10
Performance	**84.8**	**72.6**	**74.2**	**86.3**	**95.1**	**76.1**	62.4	**94.8**	**94.9**	**95.2**	**67.6**	87.4	**81.5**	95.2	11
xReliability	**85.2**	**72.9**	**74.4**	**88.0**	**95.5**	**76.5**	**67.8**	**96.0**	**95.6**	**95.7**	**70.1**	87.8	**81.6**	**95.9**	13
Polynomial	**87.6**	**76.0**	**77.9**	**87.2**	**96.8**	**74.7**	**68.6**	**95.5**	90.8	**95.4**	**80.5**	83.8	**81.8**	95.0	11
xReliability	**86.5**	**76.5**	**77.9**	**87.3**	**97.1**	68.5	**72.9**	**95.3**	**91.3**	**95.7**	**72.9**	**92.4**	**81.9**	**96.5**	13

Table 6.17. Differences of class based classifier combination (Table 6.16.-6.13)

OP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	-0.1	1.0	1.3	0.7	1.2	2.8	-0.3	-0.6	1.9	6.5	-4.1	0.4	-1.2	1.3	9
xReliability	0.9	1.3	-2.2	-0.3	1.3	2.8	0.5	1.1	-2.2	7.3	2.0	-2.1	-0.6	1.2	9
Performance	-1.2	-0.4	-2.0	0.1	1.3	1.9	9.3	1.1	-4.4	1.0	-3.1	-0.8	-0.1	1.8	7
xReliability	-1.5	1.3	-1.2	0.5	1.3	1.1	-0.9	1.1	-2.2	1.0	-4.8	1.3	-0.1	0.7	8
Polynomial	17.6	3.1	0.3	2.0	0.0	-9.3	8.4	-1.6	6.5	4.0	6.2	13.2	12.6	5.5	12
xReliability	-2.5	3.0	-0.9	20.3	0.0	-15.7	2.1	-1.1	1.0	2.8	7.4	11.0	-0.9	5.6	9
CP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	
Equal	-0.5	0.4	2.0	-0.5	0.0	1.9	-0.5	-1.1	-2.2	4.2	-5.0	0.0	1.0	0.0	8
xReliability	0.5	0.7	-1.5	-0.5	0.0	1.9	0.5	0.6	-2.2	4.2	3.0	-1.5	-1.0	-0.7	8
Performance	0.0	-0.3	-1.5	-0.9	0.0	1.0	0.5	0.6	-4.4	0.0	-1.0	-0.5	-0.5	-0.7	6
xReliability	0.5	0.8	-1.0	0.0	0.5	1.4	-5.2	0.6	-2.2	1.0	-3.0	0.0	-0.5	0.7	9
Polynomial	11.3	2.4	1.0	2.8	0.0	-5.8	7.2	-1.2	2.2	3.1	4.5	7.5	9.0	2.7	12
xReliability	-0.5	2.1	-1.0	15.4	0.0	-7.7	-0.5	-1.1	-2.2	-1.1	5.5	4.5	0.5	3.4	7
PP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	
Equal	1.9	1.5	0.2	0.3	1.3	5.3	2.4	2.9	4.8	1.4	1.9	3.8	0.4	0.3	14
xReliability	1.7	1.3	0.2	0.4	1.2	3.8	2.9	1.8	1.1	0.9	2.1	2.4	0.3	0.4	14
Performance	1.6	1.5	0.2	0.7	1.1	2.9	2.8	2.1	3.0	1.2	2.2	1.4	0.4	0.4	14
xReliability	1.5	1.3	0.2	0.6	0.9	2.0	2.2	1.4	1.0	0.6	1.9	0.9	0.2	0.5	14
Polynomial	11.5	1.2	0.1	1.2	0.0	-6.9	0.9	-0.6	11.9	1.1	3.7	7.8	8.0	0.6	12
xReliability	-0.5	1.9	-0.3	2.8	0.1	-11.0	1.3	4.5	6.8	0.3	4.3	9.2	3.0	1.6	11

Table 6.18 Probability based classifier combination (no clustering)

OP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	**74.9**	55.0	51.4	**51.3**	89.5	**72.8**	41.7	75.4	87.0	**73.3**	52.6	**80.8**	68.8	92.2	5
xReliability	**75.2**	50.1	51.4	**52.5**	89.5	**72.7**	**64.0**	80.4	89.2	**76.1**	55.6	**83.3**	69.2	91.6	6
Performance	**75.8**	54.6	51.4	**53.0**	89.5	**73.1**	**67.0**	83.5	89.2	**77.6**	**62.8**	**82.6**	**70.1**	92.2	8
xReliability	**75.8**	52.9	51.7	**52.6**	89.5	**71.7**	**62.8**	86.4	89.2	**78.7**	**62.0**	**83.3**	68.8	91.6	7
Polynomial	**81.3**	**58.2**	57.1	**57.1**	**90.8**	**68.0**	**97.4**	93.9	**97.8**	**81.7**	**64.5**	**86.9**	**70.1**	92.2	11
xReliability	**77.4**	**58.0**	59.0	**57.7**	**90.8**	**70.5**	**67.8**	93.9	**97.8**	**81.6**	**62.8**	**86.9**	**70.1**	92.2	11
CP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	**86.1**	74.1	71.5	**70.1**	**94.3**	**84.6**	**64.4**	85.4	91.1	**84.4**	72.0	**89.5**	82.5	95.3	7
xReliability	**86.1**	70.7	71.5	**71.5**	**94.3**	**84.6**	**79.4**	89.3	**93.3**	**86.5**	74.0	**90.5**	**83.0**	95.3	9
Performance	**86.6**	73.8	71.5	**71.5**	**94.3**	**85.1**	**81.4**	90.4	**93.3**	**86.5**	**78.5**	**90.5**	**83.5**	95.3	10
xReliability	85.6	72.5	71.0	**71.5**	**94.3**	**84.1**	**78.9**	92.7	**93.3**	**87.5**	**77.0**	**90.5**	82.5	95.3	8
Polynomial	**89.7**	**76.2**	75.0	**74.8**	**94.3**	**82.2**	**98.5**	96.6	**97.8**	**89.6**	**79.0**	**93.0**	**83.5**	95.3	11
xReliability	**86.6**	**76.0**	76.5	**74.8**	**94.3**	**83.7**	**70.6**	96.6	**97.8**	**89.6**	**78.0**	**92.5**	**83.5**	95.3	11
PP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	61.9	57.6	59.4	77.5	74.0	61.9	53.7	66.1	69.2	82.7	56.1	67.0	66.8	73.5	0
xReliability	62.5	58.8	59.3	79.8	75.2	62.3	58.8	66.2	71.6	84.0	57.9	67.3	66.7	73.6	0
Performance	64.3	58.3	59.7	80.0	75.8	63.0	62.3	66.2	72.4	84.4	59.2	68.3	66.6	74.0	0
xReliability	64.0	59.4	59.6	82.6	76.3	63.5	66.4	66.1	73.2	85.1	60.5	68.5	66.5	74.1	0
Polynomial	**80.3**	**67.5**	66.2	**86.6**	**96.1**	**76.2**	**92.4**	**85.9**	**95.0**	**94.7**	**67.4**	88.5	74.5	94.4	10
xReliability	77.9	**69.8**	67.1	83.7	**96.4**	**80.7**	**83.8**	**86.9**	**95.5**	**97.3**	**74.3**	**90.3**	74.3	94.8	9

Table 6.19. Differences of probability based classifier combination (Table 6.18-6.14)

OP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	-5.1	0.5	-3.9	9.1	0.5	8.5	2.6	-1.6	-6.5	7.9	4.1	0.5	-2.5	0.6	9
xReliability	-2.8	-1.3	-4.2	-0.3	0.1	5.1	7.9	-4.4	-4.3	4.3	-2.9	3.6	-2.0	0.0	6
Performance	-3.6	0.3	-4.4	-0.2	-0.4	4.2	4.5	1.0	-2.1	1.4	1.5	2.2	-0.8	0.6	8
xReliability	-2.7	1.9	-4.1	-1.7	-0.4	0.7	-3.7	1.9	-2.1	0.6	1.0	4.6	-2.2	0.0	7
Polynomial	-0.5	-1.3	-4.1	0.4	0.0	-4.0	-1.1	3.1	4.3	0.0	-0.2	-0.5	0.0	0.0	7
xReliability	-1.7	-0.1	-2.4	1.7	0.0	-2.7	0.0	6.3	0.0	0.0	2.0	-1.0	-0.3	0.0	8
CP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	-3.1	0.5	-2.5	7.0	0.5	5.3	2.1	-1.7	-4.4	5.2	3.0	0.5	-1.5	0.0	9
xReliability	-1.0	-0.8	-2.5	0.0	0.5	2.9	4.6	-1.7	-2.2	3.1	-2.0	2.5	-1.0	0.0	7
Performance	-2.1	0.4	-3.0	0.5	0.5	2.9	2.6	0.0	0.0	0.0	1.0	1.5	-0.5	0.0	11
xReliability	-2.6	1.2	-3.5	-1.4	0.5	0.5	-2.1	1.1	0.0	0.0	-0.5	2.5	-1.5	0.0	8
Polynomial	-0.5	-0.9	-3.0	0.5	0.0	-2.4	-0.5	1.7	2.2	0.0	0.0	0.0	0.0	0.0	9
xReliability	-2.1	-0.2	-1.5	1.9	0.0	-1.4	0.0	3.3	0.0	0.0	1.5	-1.0	0.0	0.0	9
PP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	-1.3	0.0	-1.0	0.1	-0.3	2.0	1.9	-1.4	0.7	-0.1	1.8	-0.2	-0.5	-0.7	5
xReliability	-1.1	-0.2	-1.1	0.0	0.0	1.4	2.2	-2.1	-0.4	-0.1	1.8	-1.0	-0.6	-0.7	5
Performance	-1.5	0.1	-1.2	0.7	0.3	1.2	2.2	-2.2	0.2	0.0	1.8	-1.4	-0.9	-0.5	8
xReliability	-1.5	-0.1	-1.2	0.6	0.4	0.8	1.6	-2.4	-0.4	-0.1	1.6	-1.7	-1.0	-0.6	5
Polynomial	-0.7	0.0	-2.1	0.0	0.0	-1.5	-0.1	-0.5	0.0	0.1	-0.5	0.2	-0.5	0.0	7
xReliability	1.3	-0.9	-2.6	-1.2	-0.1	-4.4	0.0	0.5	5.7	0.0	-1.9	-2.1	-0.1	0.1	6

Table 6.20. Performance of combined classifier combination (no clustering)

OP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	**77.2**	56.4	56.1	38.2	88.5	**70.2**	35.9	90.3	89.2	**72.7**	43.2	**79.4**	69.2	91.6	4
xReliability	**76.9**	56.4	55.8	**46.1**	90.4	**70.7**	43.8	96.7	**91.3**	**74.5**	49.6	**79.7**	69.2	91.6	6
Performance	**77.4**	56.8	56.1	**53.5**	89.4	**71.0**	**64.2**	**97.2**	**91.3**	**77.4**	52.8	**79.7**	69.2	92.2	8
xReliability	**77.4**	57.4	54.6	**54.8**	**90.9**	**71.8**	**67.5**	**97.8**	**93.5**	**76.5**	54.6	**81.1**	69.2	92.2	9
Polynomial	**79.3**	**59.2**	**60.7**	**52.1**	90.4	56.8	**80.9**	92.3	87.0	**79.9**	**66.5**	67.6	68.6	92.8	7
xReliability	**76.7**	**60.0**	**60.7**	**56.2**	90.1	51.1	45.4	91.2	84.9	**74.7**	55.4	**86.0**	69.0	92.8	6
CP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	**87.1**	75.0	74.5	59.8	93.2	**83.2**	59.8	94.4	**93.3**	**83.3**	65.5	**88.5**	**83.0**	95.3	6
xReliability	**87.1**	75.0	74.5	**66.4**	**94.3**	**83.7**	**66.0**	97.8	**93.3**	**85.4**	70.0	**88.5**	**83.0**	95.3	9
Performance	**87.6**	75.3	74.5	**71.5**	93.2	**83.7**	**79.9**	**98.3**	**93.3**	**86.5**	72.0	**88.5**	**83.0**	95.3	9
xReliability	**87.6**	75.7	73.5	**72.4**	**94.8**	**84.1**	**82.0**	**98.3**	**95.6**	**86.5**	73.5	**88.5**	**83.0**	95.3	10
Polynomial	**88.1**	**76.8**	**77.5**	**71.0**	**94.3**	75.0	**82.0**	95.5	91.1	**88.5**	**81.0**	73.0	82.0	**96.0**	9
xReliability	**87.1**	**77.3**	**77.5**	**72.9**	**94.8**	71.2	45.4	94.4	91.1	**82.3**	73.0	**92.5**	82.0	**96.0**	8
PP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	73.2	64.8	66.6	81.0	84.2	68.8	52.7	78.7	79.5	87.4	58.7	77.0	73.9	84.0	0
xReliability	73.7	65.5	66.7	83.2	85.1	69.2	58.5	80.0	82.7	89.4	61.5	77.3	74.0	84.4	0
Performance	74.6	65.5	67.0	83.2	85.5	69.6	62.3	80.5	83.7	89.8	63.4	77.9	74.0	84.6	0
xReliability	74.6	66.1	67.0	**85.3**	85.9	70.0	67.1	81.0	84.4	90.4	65.3	78.2	74.0	85.0	1
Polynomial	**83.8**	**71.8**	**72.1**	**86.9**	**96.4**	**75.5**	**80.5**	**90.6**	**92.9**	**95.0**	**74.0**	83.0	78.1	94.7	11
xReliability	**82.1**	**73.1**	**72.5**	**85.5**	**96.7**	**74.6**	**72.7**	**91.1**	**93.4**	**95.7**	**73.8**	**91.3**	78.1	**95.7**	13

Table 6.21. Differences of combined classifier combination (Table 6.20-6.15)

OP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	-0.9	0.6	1.8	0.6	2.3	3.7	11.6	-1.5	-2.1	7.9	-0.1	-1.0	-0.2	0.6	8
xReliability	-1.3	2.9	1.0	2.4	3.2	3.1	-2.1	2.8	-2.2	2.1	0.7	-0.6	-0.2	0.1	9
Performance	-3.6	1.9	-0.5	3.9	2.5	1.5	2.1	2.2	-2.2	0.0	-4.9	-0.6	-0.2	1.2	8
xReliability	-2.7	2.2	-1.8	0.1	0.5	1.9	5.5	1.1	0.0	0.0	-5.5	0.9	0.1	-0.6	10
Polynomial	12.4	1.4	0.2	3.8	0.0	-13.9	-0.1	-1.1	4.0	-0.7	2.5	10.7	11.7	2.7	10
xReliability	-3.7	0.9	-0.8	13.0	-0.3	-13.2	0.6	-0.6	0.0	-0.6	4.4	7.3	-0.3	-0.7	6
CP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	-0.5	0.4	1.0	0.0	1.6	2.4	10.8	-1.1	0.0	5.2	0.0	-0.5	0.5	0.7	11
xReliability	-0.5	2.0	1.0	1.9	1.6	1.9	-1.5	1.7	-2.2	3.1	0.5	-0.5	0.5	0.7	10
Performance	-2.1	1.3	-0.5	2.3	0.5	1.0	1.5	1.1	-2.2	0.0	-3.5	-0.5	0.5	0.7	9
xReliability	-1.5	1.4	-1.0	0.0	0.5	1.0	3.6	0.6	0.0	0.0	-3.5	0.0	0.5	0.7	11
Polynomial	7.7	1.0	0.5	3.2	0.0	-8.7	-0.5	-0.6	2.2	0.0	2.5	7.0	7.5	3.3	11
xReliability	-2.1	0.6	-0.5	8.9	0.5	-3.8	0.6	-1.1	0.0	0.0	9.0	6.5	-1.0	0.0	9
PP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	0.3	0.7	-0.4	0.2	0.5	3.7	2.2	0.7	2.7	0.6	1.8	1.8	0.0	-0.2	11
xReliability	0.3	0.6	-0.4	0.2	0.6	2.6	2.6	-0.2	0.4	0.4	1.9	0.7	-0.2	-0.2	10
Performance	0.0	0.8	-0.5	0.7	0.7	2.1	2.5	0.0	1.6	0.6	2.0	0.0	-0.3	-0.1	9
xReliability	0.0	0.6	-0.5	0.6	0.7	1.4	1.9	-0.5	0.3	0.2	1.8	-0.4	-0.4	0.0	9
Polynomial	5.4	0.6	-0.9	0.6	0.0	-4.2	0.4	-0.6	5.9	0.5	1.7	3.9	3.7	0.3	11
xReliability	0.3	0.4	-1.5	0.8	0.0	-6.1	0.3	2.5	6.3	-0.6	1.1	3.5	1.4	0.9	11

In Table 6.20, class based and probability based classifier combination results are combined without the k-means clustering and self-organizing map classification algorithms. In Table 6.21, the difference of the performance values between the new classifier set (Table 6.20) and the original classifier set with all the classifiers (Table 6.15) are given. The performances are improved as expected, since both the class based and probability based combination algorithms indicate improvements.

Another sensitivity analysis is based on the removal of the best classifier. The best classifier based on results of Table 6.2, Table 6.3 and Table 6.4 is k-nearest neighbour classifier. The analysis of removing the best classifier are given in tables Table 6.22 through Table 6.27 in the same manner as the removal of worst classifiers.

In Table 6.22, class based classifier combination results, without the k-nearest neighbour classifier, are tabulated. In Table 6.23, the difference of the performance values between the new classifier set (Table 6.22) and the original classifier set with all the classifiers (Table 6.13) are given. Based on Table 6.23, it is noted that even the removal of the best classifier may improve the performance of the combination. The "Best" column indicates on the average five improvements for 14 data sets. A closer look on the Table 6.23 indicate that the improvements are less than one per cent, where the decrease in performance are more than five per cent for most decreases. Comparing the "Best" columns of Table 6.13 and Table 6.22, there is a decrease in performance. For example for the class based combination of the original classifier set with all the classifiers using polynomial weight assignment, outperforms the single classifiers five times (Table 6.13) for 14 data sets. The same performance for the subset without the k-nearest neighbour classifier decreases to three (Table 6.22). Even this subset of classifiers without the k-nearest neighbour classifier indicates some improvements as given in Table 6.23. Continuing the same example of class based combination with polynomial weight assignment, Table 6.23 indicates that there are six cases out of 14 data set, where there are increase in performance.

Table 6.22. Class based classifier combination (no k-NN)

OP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	**77.4**	54.6	59.0	29.3	84.6	58.5	19.8	93.9	86.8	57.9	35.0	77.8	68.8	92.1	1
xReliability	**76.8**	54.4	**59.9**	35.6	85.8	61.1	21.6	96.1	89.2	64.1	45.2	78.3	68.8	**92.8**	3
Performance	**78.0**	56.0	**60.7**	34.3	85.8	59.2	22.2	**97.2**	89.2	67.2	53.9	76.3	68.8	92.8	3
xReliability	**78.0**	56.3	**61.2**	38.0	85.8	60.6	44.8	**97.8**	89.2	70.4	**58.0**	76.8	68.8	92.8	4
Polynomial	52.0	56.2	**60.5**	26.9	88.6	**64.6**	40.1	91.2	69.6	58.3	**59.5**	54.7	55.8	88.0	3
xReliability	**78.9**	57.0	**61.3**	24.5	87.7	53.3	43.8	87.6	70.3	61.1	47.4	44.9	67.5	87.9	2

CP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	85.6	71.9	73.5	50.0	90.6	72.6	43.8	96.1	88.9	68.8	49.5	86.5	82.5	94.0	0
xReliability	**86.6**	73.7	**77.0**	57.0	92.2	77.9	46.4	97.2	**93.3**	78.1	67.0	87.0	82.5	**96.0**	4
Performance	**87.1**	74.6	**77.5**	57.0	92.2	76.4	46.9	**98.3**	**93.3**	80.2	72.0	86.5	82.5	95.3	4
xReliability	**87.1**	74.6	**78.0**	59.8	92.2	77.4	**63.9**	**98.3**	**93.3**	81.3	**75.0**	87.0	82.5	95.3	6
Polynomial	70.6	74.5	**77.0**	50.9	93.8	**79.8**	60.3	94.9	82.2	74.0	**76.0**	65.5	73.5	93.3	3
xReliability	**88.1**	75.1	**77.5**	47.2	93.2	64.4	47.4	93.3	82.2	75.0	67.5	53.0	81.5	92.7	2

PP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	**82.1**	**70.8**	**74.1**	83.1	92.7	68.9	46.2	**88.8**	83.5	89.9	58.0	82.1	**81.2**	93.8	5
xReliability	**82.8**	**71.2**	**74.3**	84.8	93.4	70.9	51.6	**92.8**	**91.7**	**93.1**	61.8	84.0	**81.5**	94.6	7
Performance	**82.8**	**71.6**	**74.4**	83.7	93.5	71.5	54.6	**93.7**	90.7	**93.0**	64.4	85.0	**81.6**	94.5	6
xReliability	**83.4**	**72.0**	**74.6**	**85.2**	**94.1**	72.8	62.0	**95.6**	**93.8**	**94.3**	**67.7**	85.9	**81.8**	95.1	10
Polynomial	75.3	**74.9**	**77.9**	82.9	**96.3**	**80.6**	66.7	**95.4**	90.4	**93.9**	**77.1**	75.9	77.0	94.4	7
xReliability	**88.1**	**74.6**	**78.3**	81.1	**96.1**	**75.4**	**71.6**	**90.5**	79.0	**92.9**	**68.6**	71.5	79.9	94.6	9

Table 6.23. Differences of class based classifier combination (Table 6.22-6.13)

OP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	0.4	1.9	4.8	-3.9	-0.5	-3.1	-1.2	0.6	-0.1	-2.9	-4.0	0.0	-0.2	0.7	5
xReliability	-0.5	-0.4	1.8	-5.5	-0.9	-5.2	-3.0	0.0	-4.3	-5.0	-3.5	-2.0	0.3	1.3	3
Performance	-0.9	-0.7	1.9	-12.8	-0.9	-7.5	-17.1	0.6	-4.3	-8.3	-0.7	-2.3	0.0	1.9	4
xReliability	-1.6	1.2	2.0	-13.7	-1.8	-7.3	-14.8	0.6	-4.3	-5.1	0.0	-2.5	0.0	1.3	6
Polynomial	-8.8	0.1	-0.3	-7.8	-2.3	-1.3	-6.6	-2.2	3.6	-7.6	0.7	-0.2	0.0	0.0	5
xReliability	1.2	0.0	-0.3	-11.2	-2.4	-8.8	0.0	-3.1	-3.0	-9.0	-0.2	-29.4	-0.4	0.0	4

CP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	0.0	3.9	4.0	-3.7	0.0	0.5	-1.5	1.1	-2.2	-3.1	-6.5	0.0	2.0	0.0	9
xReliability	-1.0	-0.3	1.0	-5.6	-0.5	-2.9	-3.1	-0.6	-2.2	-4.2	-1.0	-2.0	0.0	1.3	3
Performance	-0.5	-0.5	1.0	-10.3	-0.5	-4.8	-15.5	0.6	-2.2	-5.2	-0.5	-1.5	0.0	0.7	4
xReliability	0.0	0.5	1.5	-11.2	-0.5	-3.8	-12.9	0.0	-2.2	-4.2	-0.5	-1.0	0.0	0.7	6
Polynomial	-5.7	0.2	0.0	-6.1	-1.0	-1.0	-1.6	-1.2	2.2	-5.2	0.5	-0.5	0.0	0.0	6
xReliability	2.0	0.0	0.0	-10.3	-1.6	-11.1	0.0	-1.6	-2.2	-7.3	0.0	-33.0	1.0	-0.6	6

PP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	-0.5	0.4	0.4	-1.0	-0.5	-1.5	-3.2	0.5	-1.6	-0.9	-1.4	-1.1	0.5	-0.4	4
xReliability	-0.4	0.4	0.4	-1.3	-0.5	-1.5	-3.8	0.9	-1.0	-0.8	-1.1	-1.1	0.5	-0.3	4
Performance	-0.5	0.4	0.4	-1.9	-0.5	-1.7	-5.0	0.9	-1.2	-1.0	-1.0	-1.1	0.5	-0.3	4
xReliability	-0.4	0.4	0.4	-2.2	-0.5	-1.7	-3.6	0.9	-0.8	-0.8	-0.5	-1.0	0.5	-0.3	4
Polynomial	-0.8	0.1	0.1	-3.1	-0.5	-1.0	-1.0	-0.7	11.5	-0.4	0.3	-0.1	3.2	0.0	6
xReliability	1.1	0.0	0.1	-3.4	-0.9	-4.1	0.0	-0.3	-5.5	-2.5	0.0	-11.7	1.0	-0.3	6

In Table 6.24, probability based classifier combination results, without the k-nearest neighbour classifier, are tabulated. In Table 6.25, the difference of the performance values between the new classifier set (Table 6.24) and the original classifier set with all the classifiers (Table 6.14) are given. Table 6.25 shows almost the same behaviour as the Table 6.23, except that the probability performance could not be improved. Another point is that the improvements are approximately two per cent, where the decrease in performance are more than 10 per cent for most decreases.

In Table 6.26, class based and probability based classifier combination results are combined without the k-nearest neighbour classifier. In Table 6.27, the difference of the performance values between the new classifier set (Table 6.26) and the original classifier set with all the classifiers (Table 6.15) are given. These results show the same behaviour as in the case of probability based combination. Although there is no improvement for the probability performance of the classifier subset, class performance could be improved. For this combination, the level of change in the performance is less than one per cent for improvements, but there are more than 10 per cent for decreases in performance.

Table 6.24. Probability based classifier combination (no k-NN)

OP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	**81.8**	51.2	56.5	30.8	83.6	57.9	22.6	76.0	65.5	38.2	44.3	**79.8**	**69.9**	**92.9**	4
xReliability	**80.0**	48.5	56.5	36.0	87.6	58.5	40.4	80.6	**93.5**	41.1	52.9	**80.3**	**69.9**	91.6	4
Performance	**80.9**	52.1	58.4	28.2	88.6	57.9	**50.9**	74.5	**93.5**	50.0	**57.9**	**79.8**	**69.9**	92.1	6
xReliability	**77.2**	50.3	58.7	36.1	89.4	58.5	**64.8**	80.7	89.0	51.8	**61.1**	**80.3**	**69.6**	91.6	5
Polynomial	**81.7**	**59.3**	**60.0**	37.5	90.0	**72.2**	**98.5**	89.1	**93.5**	**78.2**	**64.5**	**86.1**	**70.1**	92.2	10
xReliability	**81.3**	**57.9**	**61.5**	37.5	90.0	**68.7**	**71.9**	85.6	**93.5**	51.0	**61.3**	**82.5**	**70.1**	92.2	9
CP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	**89.7**	71.4	75.0	53.3	90.6	75.5	47.4	86.5	73.3	58.3	66.0	**88.5**	**83.0**	**96.0**	4
xReliability	**89.2**	69.5	75.0	58.4	92.7	75.5	**63.4**	89.3	**95.6**	60.4	71.5	**89.0**	**83.0**	95.3	5
Performance	**89.7**	71.9	76.0	50.5	93.8	75.5	**71.1**	85.4	**95.6**	68.8	**74.5**	**88.5**	**83.0**	94.7	6
xReliability	**87.1**	70.8	76.0	58.9	93.8	76.0	**79.9**	88.2	91.1	69.8	**76.5**	**89.0**	**83.0**	95.3	5
Polynomial	**89.7**	**77.0**	**77.0**	58.9	**94.3**	**84.1**	**99.0**	93.8	**95.6**	**86.5**	**78.5**	**92.0**	**83.5**	95.3	11
xReliability	**89.7**	**75.9**	**78.0**	58.4	**94.3**	**82.2**	**74.7**	92.1	**95.6**	54.2	**77.0**	87.0	**83.5**	95.3	9
PP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	60.9	56.5	59.6	76.0	71.7	57.5	48.9	65.9	65.4	81.2	52.7	64.5	66.3	71.7	0
xReliability	61.0	57.9	59.6	77.5	72.2	58.2	52.8	66.8	68.4	81.8	54.4	65.4	66.5	71.7	0
Performance	63.0	57.1	60.2	76.0	72.3	58.4	54.9	66.9	68.1	81.4	55.6	66.6	66.7	71.7	0
xReliability	62.4	58.4	60.2	77.7	72.5	59.0	59.8	67.2	69.2	81.6	57.2	66.9	66.8	71.8	0
Polynomial	**80.5**	**67.5**	68.3	81.3	**94.5**	73.5	**91.6**	**86.0**	**93.2**	89.0	**67.9**	85.8	74.6	94.2	7
xReliability	76.3	**70.8**	69.7	79.6	**95.0**	**82.6**	**85.8**	**85.0**	83.0	89.8	**76.3**	**91.0**	73.9	94.4	7

Table 6.25. Differences of probability based classifier combination (Table 6.24-6.14)

OP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	1.7	-3.3	1.2	-11.3	-5.4	-6.5	-16.5	-1.0	-28.0	-27.3	-4.1	-0.5	-1.3	1.3	3
xReliability	2.0	-2.9	0.9	-16.8	-1.9	-9.0	-15.7	-4.3	0.0	-30.8	-5.6	0.6	-1.3	0.0	4
Performance	1.4	-2.2	2.6	-25.0	-1.3	-11.1	-11.7	-8.1	2.2	-26.2	-3.4	-0.5	-1.0	0.6	4
xReliability	-1.3	-0.7	2.9	-18.2	-0.5	-12.5	-1.7	-3.8	-2.2	-26.3	0.1	1.6	-1.3	0.0	4
Polynomial	-0.1	-0.2	-1.2	-19.2	-0.8	0.2	0.0	-1.7	0.0	-3.5	-0.2	-1.3	0.0	0.0	5
xReliability	2.2	-0.2	0.1	-18.5	-0.8	-4.5	4.1	-2.0	-4.3	-30.6	0.5	-5.4	-0.3	0.0	5
CP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	0.5	-2.2	1.0	-9.8	-3.1	-3.8	-14.9	-0.6	-22.2	-20.8	-3.0	-0.5	-1.0	0.7	3
xReliability	2.1	-2.0	1.0	-13.1	-1.0	-6.3	-11.3	-1.7	0.0	-22.9	-4.5	1.0	-1.0	0.0	5
Performance	1.0	-1.6	1.5	-20.6	0.0	-6.7	-7.7	-5.1	2.2	-17.7	-3.0	-0.5	-1.0	-0.7	4
xReliability	-1.0	-0.5	1.5	-14.0	0.0	-7.7	-1.0	-3.4	-2.2	-17.7	-1.0	1.0	-1.0	0.0	4
Polynomial	-0.5	-0.1	-1.0	-15.4	0.0	-0.5	0.0	-1.1	0.0	-3.1	-0.5	-1.0	0.0	0.0	5
xReliability	1.0	-0.3	0.0	-14.5	0.0	-2.9	4.1	-1.2	-2.2	-35.4	0.5	-6.5	0.0	0.0	7
PP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	-2.3	-1.1	-0.7	-1.5	-2.6	-2.4	-2.9	-1.7	-3.1	-1.6	-1.6	-2.7	-0.9	-2.5	0
xReliability	-2.6	-1.0	-0.7	-2.2	-2.9	-2.7	-3.8	-1.5	-3.6	-2.2	-1.7	-2.9	-0.9	-2.6	0
Performance	-2.9	-1.1	-0.7	-3.3	-3.2	-3.4	-5.3	-1.5	-4.1	-3.0	-1.8	-3.2	-0.8	-2.8	0
xReliability	-3.0	-1.0	-0.6	-4.3	-3.5	-3.6	-5.0	-1.3	-4.4	-3.6	-1.7	-3.3	-0.8	-2.9	0
Polynomial	-0.5	0.0	0.0	-5.3	-1.6	-4.2	-0.9	-0.4	-1.8	-5.6	0.0	-2.5	-0.4	-0.2	3
xReliability	-0.3	0.1	0.0	-5.3	-1.5	-2.5	2.0	-1.4	-6.8	-7.5	0.1	-1.4	-0.5	-0.3	4

Table 6.26. Performance of combined classifier combination (no k-NN)

OP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	**77.9**	54.5	**59.9**	32.4	85.8	59.2	20.3	90.7	83.0	57.1	42.7	78.8	68.8	91.7	2
xReliability	**78.2**	54.8	**60.4**	38.5	85.8	60.9	26.7	93.9	**91.3**	64.9	51.1	**80.3**	69.2	91.6	4
Performance	**79.5**	56.1	**60.7**	35.2	85.8	60.7	28.1	95.0	89.2	62.3	54.7	77.8	69.2	91.6	2
xReliability	**79.1**	55.4	**60.4**	38.1	87.1	**62.3**	**64.5**	**97.2**	**91.3**	69.5	**57.9**	**80.3**	69.2	92.1	8
Polynomial	66.3	57.0	**60.5**	26.4	89.1	**66.2**	**68.4**	91.8	84.8	68.1	**64.1**	59.2	**70.0**	90.1	5
xReliability	**78.4**	**59.0**	**61.5**	24.5	88.6	53.6	**46.9**	87.6	85.0	47.3	51.0	46.2	67.6	**94.1**	5
CP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	**87.6**	73.6	**77.0**	55.1	92.2	76.4	44.3	94.9	88.9	72.9	65.0	87.5	82.5	95.3	2
xReliability	**87.6**	74.0	**77.5**	**60.7**	92.2	77.4	51.5	96.6	**93.3**	78.1	70.5	**89.0**	**83.0**	95.3	6
Performance	**88.7**	74.7	**77.5**	57.9	92.2	76.9	52.6	96.1	**93.3**	77.1	73.0	87.5	**83.0**	95.3	4
xReliability	**88.7**	74.3	**77.5**	**60.7**	92.2	**78.4**	**79.9**	**98.3**	**93.3**	80.2	**74.5**	**89.0**	**83.0**	94.7	10
Polynomial	79.4	75.0	**77.0**	50.5	**94.3**	**79.8**	**82.5**	95.5	88.9	78.1	**79.0**	64.5	**83.0**	92.7	6
xReliability	**87.6**	**76.6**	**78.0**	46.7	93.8	61.5	46.9	93.3	88.9	49.0	64.0	48.5	82.0	**96.7**	4
PP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	71.5	63.7	66.8	79.6	82.2	63.2	47.5	77.3	74.5	85.5	55.4	73.3	73.8	82.8	0
xReliability	71.9	64.6	66.9	81.2	82.8	64.6	52.2	79.8	80.0	87.5	58.1	74.7	74.0	83.1	0
Performance	72.9	64.3	67.3	79.9	82.9	65.0	54.7	80.3	79.4	87.2	60.0	75.8	74.1	83.1	0
xReliability	72.9	65.2	67.4	81.5	83.3	65.9	60.9	81.4	81.5	87.9	62.5	76.4	74.3	83.4	0
Polynomial	77.7	**71.2**	**73.1**	82.1	**95.4**	**77.0**	**79.1**	**90.7**	**91.8**	**91.4**	**72.5**	77.9	75.8	94.3	9
xReliability	**82.2**	**72.7**	**74.0**	80.3	**95.6**	**75.7**	**73.5**	**87.8**	81.0	88.8	**72.7**	72.9	76.9	94.5	8

Table 6.27. Differences of combined classifier combination (Table 6.26-6.15)

OP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	-0.3	-1.4	5.6	-5.2	-0.4	-7.3	-4.1	-1.1	-8.3	-7.8	-0.6	-1.5	-0.7	0.7	2
xReliability	0.1	1.3	5.5	-5.2	-1.4	-6.6	-19.1	0.0	-2.2	-7.4	2.1	0.0	-0.2	0.1	6
Performance	-1.5	1.2	4.1	-14.5	-1.1	-8.8	-33.9	-0.1	-4.3	-15.1	-3.1	-2.5	-0.2	0.7	3
xReliability	-1.0	0.2	4.0	-16.6	-3.3	-7.6	2.5	0.6	-2.2	-7.0	-2.2	0.1	0.1	-0.6	6
Polynomial	-0.6	-0.8	0.0	-21.9	-1.3	-4.5	-12.6	-1.6	1.8	-12.5	0.1	2.3	13.1	0.0	6
xReliability	-2.0	-0.1	0.0	-18.7	-1.8	-10.7	2.1	-4.2	0.1	-28.0	0.0	-32.5	-1.7	0.6	5
CP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	0.0	-1.0	3.5	-4.7	0.5	-4.3	-4.6	-0.6	-4.4	-5.2	-0.5	-1.5	0.0	0.7	5
xReliability	0.0	0.9	4.0	-3.7	-0.5	-4.3	-16.0	0.6	-2.2	-4.2	1.0	0.0	0.5	0.7	8
Performance	-1.0	0.8	2.5	-11.2	-0.5	-5.8	-25.8	-1.1	-2.2	-9.4	-2.5	-1.5	0.5	0.7	4
xReliability	-0.5	0.1	3.0	-11.7	-2.1	-4.8	1.5	0.6	-2.2	-6.3	-2.5	0.5	0.5	0.0	7
Polynomial	-1.0	-0.8	0.0	-17.3	0.0	-3.9	0.0	-0.6	0.0	-10.4	0.5	-1.5	8.5	0.0	7
xReliability	-1.6	-0.1	0.0	-17.3	-0.5	-13.5	2.1	-2.2	-2.2	-33.3	0.0	-37.5	-1.0	0.7	4
PP	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	-1.4	-0.4	-0.2	-1.2	-1.5	-2.0	-3.0	-0.6	-2.3	-1.3	-1.5	-1.9	-0.2	-1.4	0
xReliability	-1.5	-0.3	-0.1	-1.8	-1.7	-2.1	-3.8	-0.3	-2.3	-1.5	-1.4	-2.0	-0.2	-1.5	0
Performance	-1.7	-0.3	-0.1	-2.6	-1.9	-2.5	-5.1	-0.3	-2.6	-2.0	-1.4	-2.1	-0.2	-1.6	0
xReliability	-1.7	-0.3	-0.1	-3.2	-2.0	-2.7	-4.3	-0.2	-2.6	-2.2	-1.1	-2.2	-0.1	-1.6	0
Polynomial	-0.7	0.0	0.1	-4.2	-1.0	-2.7	-1.0	-0.5	4.8	-3.1	0.2	-1.2	1.4	-0.1	5
xReliability	0.4	0.0	0.0	-4.4	-1.1	-5.0	1.1	-0.8	-6.1	-7.5	0.0	-14.9	0.2	-0.3	6

The analysis of removing the worst and best classifiers improves the classification performance for some data sets. For the worst classifier removal, improvements are better than the case of removing the best classifier. A closer look at the tables show that the behaviour is different depending on the characteristics of the data sets. In fact the removal of worst classifier improves the performance as expected, but the result for the best classifier was an unexpected result. A closer look at the k-nearest neighbour classifier based on the sum of squared errors of posterior probabilities is given in Table 6.28. The bold face values indicate the best sum of squared errors among the classifiers on the same data set on that column. The last column titled "Best" indicates the number of times the classifier has the best sum of squared error of posterior probability among the classifier collection. The results show that the k-nearest neighbour classifier is one of the best ones, but not in all cases. A cross-check with Table 6.23, Table 6.25, Table 6.27 and Table 6.28 shows that there is no improvement for the data sets, where the k-nearest neighbour performs best. The GID, IMX, 2SD, 80X and ZMM are examples for this case. For example consider the GID data set, where k-nearest neighbour performs well, the class performance is 73.4 per cent in Table 6.2, probability performance is 90.6 per cent in Table 6.3 and overall performance is 56.4 per cent in Table 6.4. All these performance values

outperform other classifiers for the GID data set by at least 10 per cent. However, we also note from Table 6.23, Table 6.25 and Table 6.27 there is some decrease in performance of by least 10 per cent.

Table 6.28. Sum of squared errors on probabilities of classifier set

SSE	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
KMClus	74.3	89.7	56.5	370.5	174.6	70.4	71.6	141.6	135.3	520.1	85.1	91.4	65.2	109.1	0
SOM	32.0	70.3	49.0	109.3	55.2	91.3	125.8	31.5	57.8	95.8	109.0	26.0	32.0	**21.3**	1
FANN	32.6	**33.8**	**38.0**	116.2	56.1	**34.0**	69.2	**27.3**	85.8	224.0	62.4	34.3	**29.0**	35.1	5
ANN	**31.3**	41.1	39.9	80.8	69.9	38.4	58.5	29.7	48.2	131.4	54.4	40.6	34.2	48.3	1
KMClas	95.0	95.4	53.1	380.7	165.2	65.3	71.4	143.0	92.4	528.5	83.1	85.2	52.4	109.1	0
Parzen	72.9	70.2	96.8	86.2	95.2	93.5	79.2	87.0	92.7	82.4	86.4	59.3	76.7	84.4	0
kNN	41.6	54.7	55.5	**51.9**	**33.7**	43.1	**37.0**	49.1	**33.0**	**36.0**	52.9	32.9	45.7	33.1	5
PQD	95.8	80.5	53.6	386.0	195.0	74.7	72.3	147.3	129.3	639.0	75.3	87.5	53.9	140.4	0
PLD	53.0	51.9	42.1	183.6	121.0	56.4	56.0	76.0	69.2	190.3	51.9	56.4	30.6	51.5	0
SVC	53.1	54.7	77.9	69.4	73.7	46.0	128.6	75.5	85.9	68.4	**47.6**	**21.2**	34.5	68.9	2

To have an automated classifier, we may have a collection of fuzzy neural networks, artificial neural networks, k-nearest neigbour and support vector machine based classifiers, and combine them. The design of this subset of classifiers is based on their sum of squared error on posterior probabilities in Table 6.28. The four classifiers chosen, have the lowest sum of squared errors. The results of the combination performed using this subset are given in tables Table 6.29 and Table 6.31. Table 6.30 and Table 6.32 give the results of the differences with the complete classifier set, relatively. The results of this set was almost as good as the original classifier set with all the classifiers. The probability based combination with overall performance values assigned as weights performs better than the whole classifier set for probability performance.

Table 6.29. Class based classifier combination (best 4)

OP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	62.0	46.6	51.2	29.1	85.6	61.5	14.5	73.6	80.0	50.5	33.0	76.9	67.6	79.5	0
xReliability	**75.1**	57.6	53.0	39.9	86.4	**68.3**	25.4	93.4	89.1	69.2	48.6	77.4	68.6	79.5	2
Performance	**75.1**	57.8	55.4	**54.2**	85.7	**66.6**	34.9	92.8	86.8	**79.0**	53.4	77.4	68.6	79.5	4
xReliability	**77.4**	57.8	53.0	**53.9**	86.4	**67.5**	**46.9**	93.4	**91.3**	**80.0**	**58.8**	77.4	68.6	91.6	7
Polynomial	**77.4**	56.0	**61.6**	**49.2**	89.5	**67.5**	24.1	92.8	**93.5**	69.7	48.5	**82.4**	67.6	84.1	6
xReliability	62.5	55.7	58.4	**49.2**	89.5	**69.2**	**51.6**	92.8	**91.4**	**81.0**	48.0	**82.4**	67.6	92.8	6

CP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	68.0	57.2	65.5	45.8	90.6	72.6	32.0	74.7	80.0	60.4	47.0	84.5	81.0	82.0	0
xReliability	**86.1**	**75.8**	70.0	**61.7**	91.1	**80.3**	47.9	96.1	91.1	79.2	68.0	85.0	82.0	82.7	4
Performance	**86.1**	**75.9**	74.0	**71.5**	91.1	**78.8**	45.9	95.5	88.9	**87.5**	71.5	85.0	82.0	82.7	5
xReliability	**86.1**	**75.9**	70.0	**70.6**	91.1	**79.8**	**60.8**	96.1	**93.3**	**88.5**	72.0	85.0	82.0	95.3	7
Polynomial	**86.1**	74.7	**78.5**	**63.6**	**94.3**	76.4	37.6	95.5	**95.6**	76.0	66.0	88.0	82.0	86.7	5
xReliability	73.2	73.2	74.5	**63.6**	**94.3**	**82.7**	**65.5**	95.5	**95.6**	**89.6**	65.5	88.0	82.0	**96.0**	7

PP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	**82.6**	**73.3**	**72.7**	84.9	93.4	**77.4**	47.9	**93.1**	84.6	89.4	58.8	88.2	**81.0**	91.7	6
xReliability	**83.2**	**73.6**	**72.9**	**87.0**	**94.3**	**77.8**	54.8	**93.8**	**91.2**	**93.7**	63.6	88.3	**81.1**	92.8	10
Performance	**83.7**	**73.6**	**73.0**	**87.2**	**94.4**	**77.8**	61.7	**94.2**	**93.8**	**94.7**	**67.2**	88.4	**81.3**	92.8	11
xReliability	**84.2**	**73.9**	**73.2**	**88.8**	**95.1**	**78.1**	**67.2**	**94.8**	**95.0**	**95.4**	**69.7**	88.5	**81.3**	93.8	12
Polynomial	**87.5**	**74.3**	**76.9**	**88.4**	**97.1**	**79.9**	62.7	**94.4**	**96.6**	**95.3**	**67.8**	**91.0**	**81.4**	94.7	12
xReliability	79.7	**74.5**	**75.7**	**87.1**	**97.0**	**79.5**	**72.0**	**93.8**	**97.1**	**97.0**	**67.9**	**90.9**	**81.3**	95.3	12

Table 6.30. Differences of class based classifier combination (Table 6.29-6.13)

OP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	-15.0	-6.1	-3.1	-4.1	0.5	-0.1	-6.5	-19.7	-7.0	-10.3	-6.0	-1.0	-1.4	-12.0	1
xReliability	-2.3	2.7	-5.1	-1.2	-0.3	2.0	0.8	-2.7	-4.4	0.0	-0.1	-2.9	0.2	-12.0	5
Performance	-3.8	1.1	-3.4	7.1	-1.0	0.0	-4.4	-3.9	-6.7	3.5	-1.2	-1.3	-0.2	-11.4	3
xReliability	-2.2	2.7	-6.1	2.3	-1.2	-0.4	-12.7	-3.8	-2.2	4.5	0.8	-1.9	-0.2	0.1	5
Polynomial	16.6	-0.1	0.8	14.5	-1.4	1.6	-22.6	-0.6	27.5	3.8	-10.3	27.5	11.8	-3.9	8
xReliability	-15.2	-1.3	-3.2	13.5	-0.6	7.1	7.8	2.1	18.1	10.9	0.4	8.1	-0.3	4.9	9

CP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	-17.6	-10.8	-4.0	-7.9	0.0	0.5	-13.4	-20.2	-11.1	-11.5	-9.0	-2.0	0.5	-12.0	2
xReliability	-1.5	1.8	-6.0	-0.9	-1.6	-0.5	-1.6	-1.7	-4.5	-3.1	0.0	-4.0	-0.5	-12.0	2
Performance	-1.5	0.8	-2.5	4.2	-1.6	-2.5	-16.5	-2.3	-6.7	2.1	-1.0	-3.0	-0.5	-12.0	3
xReliability	-1.0	1.8	-6.5	-0.4	-1.6	-1.5	-16.0	-2.2	-2.3	3.1	-3.5	-3.0	-0.5	0.6	3
Polynomial	9.8	0.4	1.5	6.6	-0.5	-4.4	-24.3	-0.6	15.6	-3.2	-9.5	22.0	8.5	-6.6	7
xReliability	-12.9	-1.9	-3.0	6.1	-0.5	7.2	18.1	0.6	11.2	7.3	-2.0	2.0	1.5	2.7	9

PP Class	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	0.0	2.9	-0.9	0.7	0.2	7.0	-1.4	4.8	-0.5	-1.4	-0.6	5.0	0.3	-2.5	7
xReliability	-0.1	2.7	-1.0	0.9	0.4	5.4	-0.6	1.9	-1.4	-0.2	0.7	3.2	0.2	-2.1	8
Performance	0.5	2.5	-1.0	1.6	0.4	4.6	2.1	1.4	1.9	0.7	1.9	2.4	0.2	-2.0	12
xReliability	0.4	2.3	-1.0	1.4	0.5	3.6	1.5	0.2	0.4	0.3	1.6	1.6	0.0	-1.6	11
Polynomial	11.4	-0.5	-0.9	2.4	0.3	-1.7	-5.0	-1.7	17.7	1.0	-9.0	15.0	7.6	0.3	8
xReliability	-7.3	-0.1	-2.5	2.6	0.0	0.0	0.4	3.0	12.6	1.6	-0.7	7.7	2.4	0.4	10

Table 6.31 Probability based classifier combination (best 4)

OP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	**75.5**	54.9	51.4	**52.3**	89.5	**72.5**	27.2	72.2	89.2	**72.8**	**58.7**	79.1	67.9	92.2	5
xReliability	**74.8**	56.0	50.9	**51.8**	89.5	**71.8**	**59.3**	77.3	89.2	**78.0**	**62.9**	78.1	68.4	91.6	6
Performance	74.3	54.8	51.4	**53.0**	89.5	**70.7**	**62.2**	79.4	89.2	**76.2**	**63.6**	78.6	68.8	92.2	5
xReliability	73.6	57.0	51.2	**53.0**	89.5	**68.9**	**64.0**	82.2	89.2	**78.7**	**63.8**	78.1	68.4	91.6	5
Polynomial	72.7	56.0	52.8	**56.9**	**90.8**	**68.7**	**71.8**	82.5	89.2	**80.7**	**61.8**	**86.4**	**70.1**	92.2	8
xReliability	**75.1**	56.3	52.8	**57.8**	**90.8**	**68.0**	**79.1**	78.9	89.2	**80.7**	**61.0**	**86.4**	**70.1**	92.2	9

CP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	**86.6**	73.7	71.5	**71.0**	**94.3**	**84.6**	52.1	82.6	**93.3**	**84.4**	**76.0**	**88.5**	82.0	95.3	8
xReliability	**86.1**	74.7	71.0	**71.0**	**94.3**	**83.7**	**76.8**	87.1	**93.3**	**87.5**	**78.5**	88.0	82.5	95.3	8
Performance	85.1	73.7	71.5	**71.5**	**94.3**	**83.7**	**75.8**	88.8	**93.3**	**86.5**	**78.5**	88.0	82.5	95.3	7
xReliability	**85.6**	75.4	71.0	**72.0**	**94.3**	**82.7**	**75.8**	89.9	**93.3**	**87.5**	**78.5**	88.0	82.5	95.3	8
Polynomial	85.1	74.7	72.5	**74.3**	**94.3**	**82.7**	**84.5**	90.4	**93.3**	**88.5**	**76.5**	**92.5**	**83.5**	95.3	9
xReliability	**86.1**	74.9	72.5	**74.8**	**94.3**	**82.2**	**88.7**	88.2	**93.3**	**88.5**	**76.5**	**92.5**	**83.5**	95.3	10

PP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	67.6	60.7	64.5	81.4	83.0	70.4	50.8	73.2	77.7	85.1	60.2	74.6	72.0	82.3	0
xReliability	71.2	61.3	64.4	**85.2**	83.7	70.7	57.5	72.8	84.8	88.6	63.8	74.6	72.0	83.0	1
Performance	71.9	62.6	64.2	84.0	85.5	70.7	64.1	73.6	86.1	89.7	66.3	75.2	71.9	84.9	0
xReliability	72.0	63.0	64.1	**87.2**	86.1	71.0	**69.2**	73.3	87.9	**90.9**	**68.5**	75.2	71.8	85.6	4
Polynomial	76.1	64.9	61.4	**85.3**	**96.0**	73.5	**75.3**	75.0	**94.7**	**94.8**	**68.3**	88.5	73.9	94.4	6
xReliability	79.2	64.6	61.7	84.1	**96.1**	**73.7**	**87.6**	75.4	**96.2**	**97.3**	**69.3**	88.5	73.9	94.7	6

Table 6.32 Differences of probability based classifier combination (Table 6.31-6.14)

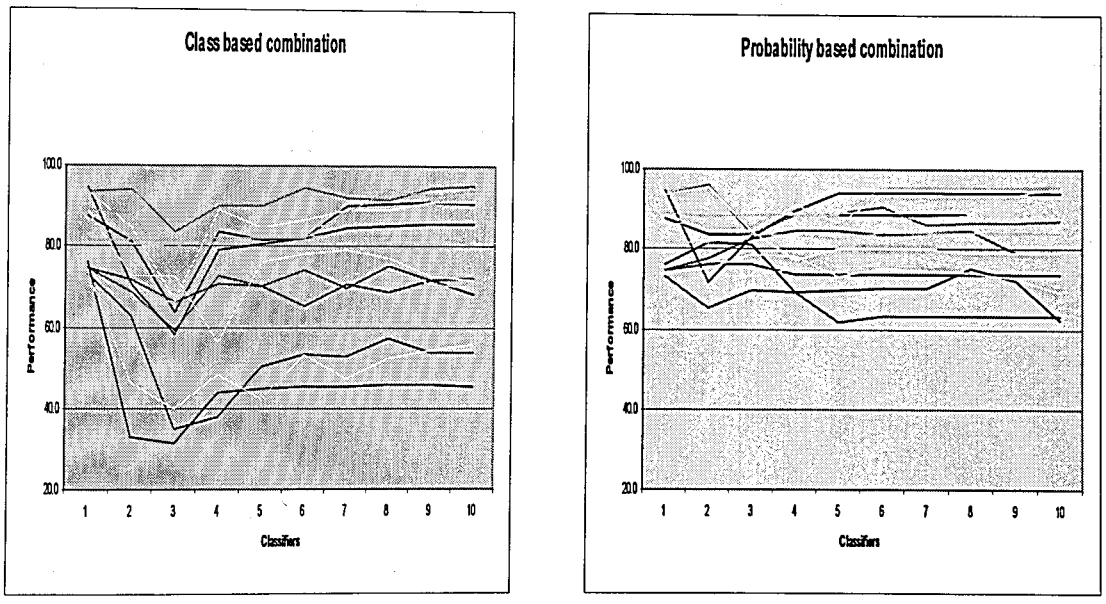
OP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	-4.5	0.4	-3.9	10.2	0.5	8.1	-11.9	-4.9	-4.3	7.4	10.2	-1.2	-3.4	0.6	7
xReliability	-3.2	4.6	-4.7	-1.0	0.1	4.2	3.1	-7.6	-4.3	6.1	4.4	-1.6	-2.9	0.0	7
Performance	-5.2	0.5	-4.4	-0.2	-0.4	1.8	-0.3	-3.2	-2.1	0.0	2.3	-1.7	-2.1	0.6	4
xReliability	-4.9	6.0	-4.6	-1.3	-0.4	-2.1	-2.5	-2.3	-2.1	0.6	2.8	-0.6	-2.5	0.0	4
Polynomial	-9.1	-3.5	-8.4	0.2	0.0	-3.3	-26.7	-8.3	-4.3	-1.0	-2.9	-1.0	0.0	0.0	4
xReliability	-4.0	-1.8	-8.6	1.8	0.0	-5.2	11.3	-8.7	-8.6	-0.9	0.2	-1.5	-0.3	0.0	5

CP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	-2.6	0.1	-2.5	7.9	0.5	5.3	-10.3	-4.5	-2.3	5.2	7.0	-0.5	-2.0	0.0	6
xReliability	-1.0	3.2	-3.0	-0.5	0.5	2.0	2.1	-3.9	-2.3	4.2	2.5	0.0	-1.5	0.0	7
Performance	-3.6	0.3	-3.0	0.5	0.5	1.5	-3.1	-1.6	0.0	0.0	1.0	-1.0	-1.5	0.0	6
xReliability	-2.5	4.0	-3.5	-0.9	0.5	-1.0	-5.1	-1.7	0.0	0.0	1.0	0.0	-1.5	0.0	5
Polynomial	-5.1	-2.4	-5.5	0.0	0.0	-1.9	-14.5	-4.5	-2.3	-1.1	-2.5	-0.5	0.0	0.0	4
xReliability	-2.6	-1.3	-5.5	1.9	0.0	-2.9	18.1	-5.1	-4.5	-1.1	0.0	-1.0	0.0	0.0	6

PP Prob	BIO	DIB	D10	GID	IMX	SMR	2SD	WQD	80X	ZMM	BEM	BEV	HRD	IFD	Best
Equal	4.4	3.1	4.1	3.9	8.7	10.5	-1.0	5.6	9.2	2.3	5.8	7.4	4.7	8.1	13
xReliability	7.6	2.4	4.1	5.5	8.5	9.8	1.0	4.5	12.8	4.5	7.7	6.3	4.7	8.7	14
Performance	6.1	4.4	3.4	4.7	10.0	8.9	3.9	5.2	13.9	5.3	8.9	5.4	4.4	10.3	14
xReliability	6.5	3.6	3.3	5.2	10.1	8.4	4.5	4.8	14.3	5.8	9.6	4.9	4.2	10.9	14
Polynomial	-4.9	-2.6	-6.9	-1.3	-0.1	-4.2	-17.2	-11.4	-0.3	0.2	0.4	0.2	-1.1	0.0	4
xReliability	2.6	-6.1	-8.0	-0.8	-0.4	-11.4	3.8	-11.0	6.4	0.0	-6.9	-3.9	-0.5	0.0	5

A more detailed sensitivity analysis is done on data sets separately. For each data set the classifiers are sorted based on the sum of squared errors, respectively. Beginning with the best classifier, all classifiers are incrementally added to the classifier set. Their performance changes are graphically presented in Figure 6.1 and Figure 6.2.

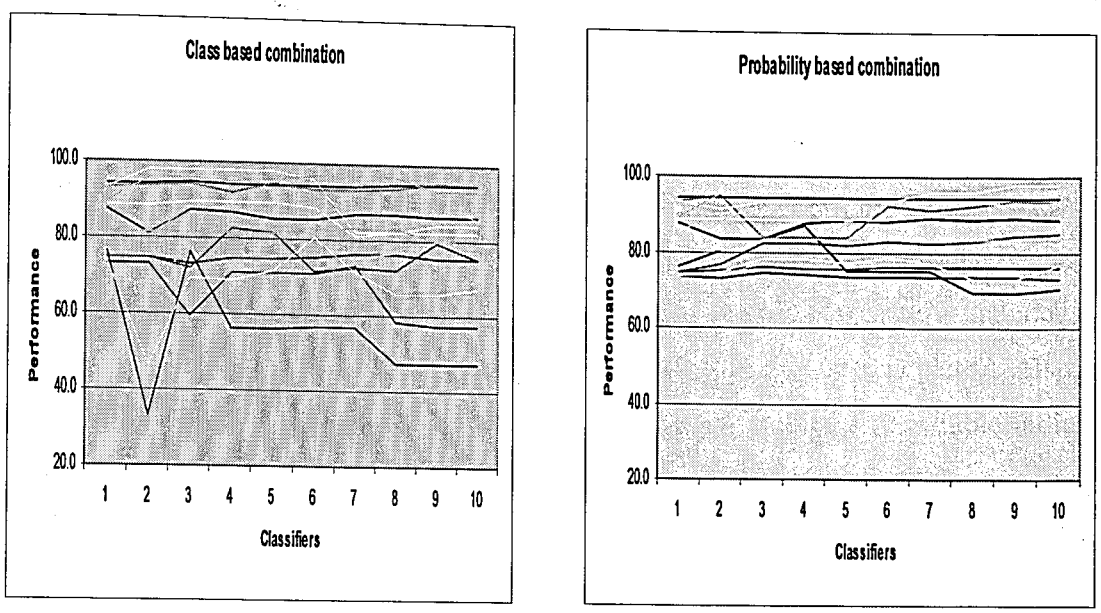
As can be seen in Figure 6.1, the class performance for the equal weight assigned classifier set, after the best classifier is in the subset, the performance first drops by adding the second or third best classifier and then it begins to improve sometimes to its initial level and sometimes above it. This is due to the fact that the best classifiers in the classifier set are not independent with each other, that is they are correlated in the misclassification of the same test samples. A complete search for the best classifier subset may also be carried out by considering all possible subsets of the original classifier set. A further research should be done on the class characteristics of the data sets, since the performance of classifiers are not just data set dependent, their performance also depend on the different classes of the data set. For some classifiers, certain classes of some data sets may be classified more reliable than the other classes of the same data set. In fact in this thesis, this kind of analysis is added to the combination by the proposed reliability factors. Another research may be the possibility to enlarge the classifier set, even such a set was not used in literature before, different kinds of neural networks could be added to the set.



(a) (b)

Figure 6.1. Class performance of equal weight assignment

Comparing the sensitivity of adding classifiers to the set in the figures Figure 6.1 and Figure 6.2 show that polynomial weight assignment is more robust for addition of new classifiers. Another point is that probability based classifier combination is more robust to class based combination algorithms, as can be noticed by the less variability in the performance, which ranges between 70 per cent and 100 per cent (Figure 6.2(b)), compared to the large variation of the equal case of 60-100 per cent range (Figure 6.1(b)). Based on this analysis we can outline a guide for designing a classifier set for combination.



(a) (b)

Figure 6.2. Class performance of polynomial weight assignment with reliability

7. SUMMARY AND CONCLUSIONS

Pattern recognition is the science that is concerned with the description or classification/recognition of measurements. There is little doubt that pattern recognition is an important, useful, and rapidly developing technology with cross-disciplinary interest and participation. The aim of pattern recognition is the design of classifier mechanisms that takes some features of sample test patterns as input and output a classification of the pattern to indicate to which class it belongs to.

It is very difficult to make sense of the multitude of empirical comparisons for classifier performances that have been made. There are no agreed objective criteria by which to judge algorithms. The situation is made more difficult because rapid advances are being made in all fields of pattern recognition. Any comparative study that does not include the majority of the algorithms is clearly not aiming to be complete. Also, any comparative study that looks at limited number of data sets cannot give reliable indicators of performance. In this thesis, an extensive comparative study is realized among a wealth of classifiers applied on different data sets. Their performances are evaluated using a variety of performance measures. This comparative study builds the base of the study on combining classifiers to improve the classification performance.

No single technology is the optimum solution for all pattern recognition problems, hence we have to incorporate all available and relevant information in a structured fashion to formulate a solution. This includes the development or modification of models that incorporates structural and a priori information. For different applications we may have different feature sets, different training sets, different classification methods or different training sessions, all resulting in a set of classifiers whose outputs may be combined, with the hope of improving the overall classification accuracy. If this set of classifiers is fixed, the problem focuses on the combination function. It is also possible to use a fixed combiner and optimize the set of input classifiers. In this thesis the set of classifiers are fixed, i.e. they are not optimized especially for the application at hand. Different combination schemes are developed hoping to reach reasonable classification accuracy,

which is independent of the characteristic of the application. In this thesis, we aim to build a robust classifier combination system given a classifier set.

For the purpose of classifier combination, the current trends in classifier combination are studied and various classifier combination schemes have been devised. To test their performances a wide range of classifiers are collected in a classifier set and classifier combination schemes are applied on this set. The K-means clustering and self organizing map clustering algorithms are modified to use them as classifiers, by solving an optimization problem to assign clusters to class. For some classifiers such as fuzzy neural network classifier, artificial neural network classifier and support vector machine, their belief values are converted to posterior probabilities using a normalized mapping. For K-means classifier, Parzen, K-nearest neighbour, piecewise quadratic and piecewise linear distance classifiers their implicit distance measures are explicitly calculated and converted to posterior probabilities for each class. One of the main contributions of this thesis is that all the classifiers and clustering based classifier algorithms in the classifier set are modified to produce posterior probabilities for their class assignments for all classes.

In this thesis the modified classifiers are integrated in a framework including a complete software package for pattern recognition. The developed software enables to test the proposed combination schemes providing a useful pattern recognition toolbox. Different classifier combination schemes are proposed and realized in this framework. All classifiers and combination schemes are evaluated using a variety of performance measures.

When combining different classifiers, the weighted combination methods are applied as the combination schemes. The two basic questions concerning the weighting methods: what to weight and how to weight, direct us to new ideas of alternatives for combination methods and for weight assignments. Class based and probability based and combined combination methods are applied on the data set and experimentally demonstrated that the proposed probability based weighted combination method is a robust way of combining classifiers. The weights of classifiers are basically based on their performances in the training phase, assuming they will achieve almost the same performance for the test samples. Overall performance values, originated by the class performance and reliability

values proportional to the class population for a specific class after the leave-one-out training phase, is proposed for weight assignment. A better way of assigning weights is proposed by using the least square fit parameters of true and estimated posterior probabilities in the leave-one-out training, and it is called polynomial weight assignment. The classical equal weight assignment is also implemented and tested in the framework to compare the effectiveness of proposed methods.

To have a complete comparative study, all the proposed combination schemes and weight assignments are applied on the data sets and their performances are evaluated using the proposed performance measures. Some of the performance measures proposed in this thesis include the reliabilities of the classifiers for their decisions based on their training performances. The reliability values integrate their trust for their decisions, which is used with the assigned weights, to improve the performance of correct classification and reduce the overall misclassification error. So the integration of the reliability measure in the combination improves the classification performance, even the simplest combination scheme of equal weight assignment for classifiers is improved. The main contribution of this thesis is that a typical one to three per cent consistent improvement compared to a single best classifier is seen when combining classifiers using the polynomial weight assignment applied with probability based combination.

Sensitivity analysis of selecting a classifier subset to achieve best performance possible with the current classifier set is performed. The basic idea behind selection of a classifier for the classifier set is that, the individual classifier which will be added in the set should not be strongly correlated in the misclassification of the current classifiers in the classifier set. This criterion is not easily satisfied, since even different classes of the same data set may have different characteristics. That is, the classifier's performance may vary for the different classes of the same data set. The results of different sensitivity analysis show that the probability based combination with polynomial weights is a robust way to combine classifiers. Probability based combination with polynomial weights achieves the best possible performance with the current set of classifiers at hand.

Different classifiers, training methods and combination algorithms are covered throughout this study. The classifiers are not specifically tuned for the data set at hand

even though they may reach a better performance with a different parameter set, since the goal is to design an automated classifier combination based on any classifier in the classifier set. The classifiers are modified to produce class probability estimates besides their class assignments and they are integrated in a framework. The time consuming leave-one-out training method is used and the results are combined using weighted combination algorithm. The weights of the classifiers are determined based on the performance of the classifiers on the training data set. Results of probability based combination with polynomial weights are promising and the integration of the reliability of the classifier for a class assignment improves the classification performance.

A further research topic would be to study the dependency structure of the classifiers. If the classifier dependence could be evaluated and if it is independent of the characteristics of the classifier set, this information could be used to try to improve the performances further. Another further research topic is the evaluation of local performances for classifiers, which may be used for development of a hierarchical architecture for classifier combination.

Assuming that at the design time of the classifier set, the time is not a critical issue, hence all available classifiers should be trained. If possible, they may be tuned for better performance. At training phase the time costly leave-one-out algorithm should be used. The reliabilities after this training should be recorded for testing phase. Than a sensitivity analysis should be carried out: either beginning with best classifier based on the sum of squared probability errors, all classifiers are added to the classifier set and the new performance is traced after each step till all classifiers are added, or all possible subsets of classifiers for classifier set should be considered. Either the best set or if the performance difference is not so high, all classifiers should be selected for classifier set. Finally using results of leave-one-out, the polynomial weights and the class reliabilities of classifiers should be calculated and used for the testing phase. The probability based combination algorithm is more robust as the combination algorithm.

The main contribution of this thesis is that a typical one to three per cent consistent improvement compared to a single best classifier is seen when combining classifiers using the polynomial weight assignment applied with probability based combination.

8. REFERENCES

Alexandre, L., A. Campilho and M. Kamel, 2000, "Combining Unbiased and Independent Classifiers Using Weighted Average", *11th Portuguese Conference on Pattern Recognition*, pp. 495-498, Porto, Portugal.

Ali, K. M. and M. J. Pazzani, 1996, "Error Reduction Through Learning Multiple Descriptions", Machine Learning, Vol. 24, No. 3, pp. 173-202.

Al-Sultan, K. S. and M. M. Khan, 1996, "Computational Experience on Four Algorithms for the Hard Clustering Problem", *Pattern Recognition Letters*, Vol. 17, No. 3, pp. 295-308.

Anderberg, M. R., 1973, *Cluster Analysis for Applications*, Academic Press, New York.

Anderson, J. A. and E. Rosenfeld, 1988, *Neurocomputing: Foundations of Research*, MIT Press, Cambridge, Massachusetts.

Backer, E., 1995, *Computer-Assisted Reasoning in Cluster Analysis*, Prentice Hall International Ltd., Hertfordshire, UK.

Bauer, E. and R. Kohavi, 1999, "An Empirical Comparison of Voting Classification Algorithms: Bagging", *Boosting and Variants, Machine Learning*, Vol. 36.

Bishop, C. M., 1995, *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford.

Boser, B. E., I. M. Guyon and V. Vapnik, 1992, "A Training Algorithm for Optimal Margin Classifiers", *In Fifth Annual Workshop on Computational Learning Theory*, ACM, Pittsburgh.

Brailovsky, V. L., 1991, "A Probabilistic Approach to Clustering", *Pattern Recognition Letters*, Vol. 12, No. 4, pp. 193-198.

Breiman, L, 1996a, "Bagging Predictors", *Machine Learning*, Vol. 4, pp. 123-140.

Breiman, L, 1996b, "Stacked Regressions", *Machine Learning*, Vol. 4, pp. 49-64.

Burges, C. J. C. and B. Schölkopf, 1997, "Improving the Accuracy and Speed of Support Vector Learning Machines", *Advances in Neural Information Processing Systems*, pp. 375-331, MIT Press, Cambridge, Massachusetts.

Carpenter, G. and S. Grossberg, 1990, "ART3: Hierarchical Search using Chemical Transmitters in Self Organizing Pattern Recognition Architectures", *Neural Networks*, No. 3, pp. 129-152.

Christianini, N. and J. S. Taylor, 2000, *Support Vector Machines and other kernel-based learning methods*, Cambridge University Press, UK.

Clemen, R. T., 1989, "Combining Forecasts: A Review and Annotated Bibliography", *International Journal of Forecasting*, No. 5, pp. 559-583.

Cortes, C. and V. Vapnik, 1995, "Support Vector Networks", *Machine Learning*, Vol. 20, pp. 273-297.

Diday, E. and J. C. Simon, 1976, "Clustering Analysis", *Digital Pattern Recognition*, Ed. Springer-Verlag, Secaucus, New Jersey, pp. 47-94.

Dietterich, T. G., 1997, *Machine Learning Research: Four Current Directions*.

Dubes, R. C. and A. K. Jain, 1980, "Clustering Methodology in Exploratory Data Analysis", *Advances in Computers*, Ed. Academic Press, New York, pp. 113-125.

Duda, R. O. and P. E. Hart, 1973, *Pattern Classification and Scene Analysis*, John Wiley&Sons, New York.

Duran, B. S. and P. L. Odell, 1974, *Cluster Analysis: A Survey*, Springer Verlag, New York.

Efron, B., 1983, "Estimating the Error-rate of a Prediction Rule: Improvement on Cross-validation", *Journal of the American Statistical Association*, No. 78, pp. 316-331.

Efron, B. and R. J. Tibshirani, 1993, *An Introduction to the Bootstrap*, Chapman &Hall.

Everitt, B. S., 1993, *Cluster Analysis*, Edward Arnold, Ltd., London, UK.

Fisher, R. A., 1936, "The Use of Multiple Measurements in Taxonomic Problems", *Contributions to Mathematical Statistics*, John Wiley and Sons, New York.

Fletcher, R., 1987, *Practical Methods of Optimization*, John Wiley and Sons, New York.

Freund, Y. and R. E. Schapire, 1997, "A Decision-theoretic Generalization of On-line Learning and an Application to Boosting," *Journal of Computer and System Sciences*, Vol. 5, No. 1, pp. 119-139.

Friedman, J. H. and W. Stuetzle, 1981, "Projection Pursuit Regression", *Journal of the American Statistical Association*, No. 76, pp. 817-823.

Fu, K. S., 1982, *Syntactic Pattern Recognition and Applications*, Prentice Hall, Englewood Cliffs, New Jersey.

Fukunaga, K., 1972, *Statistical Pattern Recognition*, Academic Press, New York.

Gorman, R. P. and T. J. Sejnowski, 1988, "Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets", *Neural Networks*, Vol. 1, pp. 75-89.

Gowda, K. C. and G. Krishna, 1977, "Agglomerative Clustering using the Concept of Mutual Nearest Neighbourhood", *Pattern Recognition*, No. 10, pp. 105-112.

Haerdle, W., 1990, *Applied Nonparametric Regression*, Cambridge University Press, Cambridge UK.

Hand, D. J., 1981, *Discrimination and Classification*, John Wiley &Sons, New York.

Hartigan, J. A. , 1975, *Clustering Algorithms*, John Wiley and Sons, New York.

Hashem, S., 1993, "Optimal Linear Combinations of Neural Networks", Ph. D. Thesis, Purdue University, School of Industrial Engineering, Indiana.

Hertz, J., A. Krogh and R. G. Palmer, 1991, "Introduction to the Theory of Neural Computation", Santa Fe Institute Studies in the Sciences of Complexity lecture notes, AddisonWesley Longman Publication, Reading, Massachusetts.

Hopfield, J. J., 1982, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *National Academy of Sciences*, USA, Vol. 79, pp. 2554-2558.

Ichino, M. and H. Yaguchi, 1994, "Generalized Minkowski Metrics for Mixed Feature Type Data Analysis", *IEEE Trans. on Systems*, Man, Cybernetics, No. 24, pp. 698-708.

Jain, A. K. and R. C. Dubes, 1988, *Algorithms for Clustering Data*, Prentice-Hall Advanced Reference Series, Prentice-Hall, Upper Saddle River, New Jersey.

Jain, A. K. and P. J. Flynn, 1996, "Image Segmentation Using Clustering", *Advances in Image Understanding*, pp. 65-83, IEEE Press, Piscataway, New Jersey.

Jain, N. C., A. Indrayan and L. R. Goel, 1986, "Monte Carlo Comparison of Six Hierarchical Clustering Methods on Random Data", *Pattern Recognition*, Vol. 19, No. 1, pp. 95-99.

Jain, A. K. and J. Mao, 1996, "Artificial Neural Networks: A Tutorial", *IEEE Computer*, No. 29, pp. 31-44.

Jain, A. K., M. N. Murty and P. J. Flynn, 1999, "Data Clustering: A Review", *ACM Computing Surveys*, Vol. 31, No. 3, pp. 264-323.

Jarvis, R. A. and E. A. Patrick, 1973, "Clustering Using a Similarity Method Based on Shared Near Neighbours", *IEEE Trans. on Computation*, No. 22, pp. 1025-1034.

Jordan, M. I. and R. A. Jacobs, 1994, "Hierarchical Mixtures of Experts and the EM Algorithm", *Neural Computation*, Vol. 6, No. 2, pp. 181-214.

Kaufman, L., 1998, "Solving the QP Problem for Support Vector Training", *Preceedings of the NIPS Workshop on Support Vector Machines.*

King, B., 1967, "Stepwise Clustering Procedures", *Journal of Statistical Association*, No. 69, pp. 86-101.

Kittler, J., 1998, "Combining Classifiers: A Theoretical Framework", *Pattern Analysis and Applications*, Vol. 1, No. 1, pp. 18-28.

Kittler, J., M. Hatef, R. P. W. Duin and J. Matas, 1998, "On Combining Classifiers", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 3, pp. 226-240.

Kohonen, T., 1984, *Self-Organization and Associative Memory*, Springer-Verlag, Berlin.

Lee, R. C. T. , 1981, "Cluster Analysis and its Applications", *Advances in Information Systems Science*, J. T. Tou, Ed. Plenum Press, New York.

Lu, S. Y. and K. S. Fu, 1978, "A Sentence to Sentence Clustering Procedure for Pattern Analysis", *IEEE Trans. on Systems, Man, Cybernetics*, No. 8, pp. 381-389.

Mao, J. and A. K. Jain, 1992, "Texture Classification and Segmentation using Multiresolution Simultaneous Autoregressive Models", *Pattern Recognition*, Vol. 25, No. 2, pp. 173-188.

Mao, J. and A. K. Jain, 1995, "Artificial Neural Networks for Feature Extraction and Multivariate Data Projection", *IEEE Trans. on Neural Networks*, No. 6, pp. 296-317.

Mao, J. and A. K. Jain, 1996, "A Self Organizing Network for Hyperellipsoidal Clustering (HEC)", *IEEE Trans. on Neural Networks*, No. 7, pp. 16-29.

McCulloch, W. S. and W. Pitts, 1943, "A Logical Calculus of Ideas Immanent in Nervous Activity", *Bulletin Mathematical Bio-physics*, Vol. 5, pp. 115-133.

MCQueen, J., 1967, "Some Methods for Classification and Analysis of Multivariate Observations", *Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281-297.

Michalski, R., R. E. Stepp and E. Diday, 1983, "Automated Construction of Classifications: Conceptual Clustering versus Numerical Taxonomy", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-5, pp. 396-409.

Minsky, M. and S. Papert, 1969, *Perceptrons: An Introduction to Computational Geometry*, MIT Press, Cambridge, Massachusetts.

Mishra, S. K. and V. V. Raghavan, 1994, "An Empirical Study of the Performance of Heuristic Methods for Clustering", *Pattern Recognition in Practice*, E. S. Gelsema and L. N. Kanal, Eds., pp. 425-436.

Murtagh, F., 1984, "A Survey of Recent Advances in Hierarchical Clustering Algorithms which use Cluster Centers", *Computer Journal*, No. 26, pp. 354-359.

Oehler, K. L. and R. M. Gray, 1995, "Combining Image Compression and Classification Using Vector Quantization", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 17, pp. 461-473.

Osuna, E., R. Freund and F. Girosi, 1997, An Improved Training Algorithm for Support Vector Machines, *IEEE Workshop on Neural Networks for Signal Processing*, pp. 276 -285, Amelia Island, Florida.

Pavlidis, T., 1977, *Structural Pattern Recognition*, Springer-Verlag, New York.

Perrone, M. P. and L. N. Cooper, 1993, "When Networks Disagree: Ensemble Methods for Hybrid Neural Networks", *Neural Networks for Speech and Image Processing*, Chapman and Hall.

Rasmuusen, E., 1992, "Clustering Algorithms", *Information Retrieval: Data Structures and Algorithms*, Eds. Prentice-Hall, Upper Saddle River, NJ, pp. 419-442.

Ripley, B. D., 1989, *Statistical Inference for Spatial Processes*, Cambridge University Press, New York.

Rosenblatt, R., 1962, *Principles of Neurodynamics*, Spartan Books, New York.

Rumelhartand, D. E. and J. L. McClelland, 1986, *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, MIT Press, Cambridge, Massachusetts.

Salton, G., 1991, *Developments in Automatic Text Retrieval*, Science 253, pp. 974-980.

Shalkoff, R. J., 1990a, *Artificial Intelligence: An Engineering Approach*, McGraw-Hill, NewYork.

Shalkoff, R. J., 1992b, *Pattern Recognition: Statistical, Structural and Neural Approaches*, John Wiley&Sons, NewYork.

Schölkopf, B., C. Borges and V. Vapnik, 1996, "Incorporating Invariances in Support Vector Learning Machines", *Artificial Neural Networks ICANN'96*, Springer Lecture Notes in Computer Science, Vol. 1112. pp. 47 -52, Berlin.

Schölkopf, B., P. Simard, A. Smola and V. Vapnik, 1998, "Prior Knowledge in Support Vector Kernels", *Advances in Neural Information Processing Systems 10*, MIT Press, Cambridge, Massachusetts.

Schölkopf, B., A. Smola, K. R. Müller, C. J. C. Burges and V. Vapnik, 1998, "Support Vector Methods in Learning and Feature Extraction", *Ninth Australian Congress on Neural Networks.*

Schölkopf, B., K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio and V. Vapnik, 1997, "Comparing Support Vector Machines with Gaussian Kernels to Radial Basis Function Classifiers", *IEEE Trans. on Signal Processing*, Vol. 45, pp. 2758-2765.

Shao, J., 1993, "Linear Model Selection by Crossvalidation", *Journal of the American Statistical Association*, No. 88, pp. 486-494.

Singh, A., J. F. Haddon and M. Markou, 1999, "Nearest Neighbour Strategies for Image Understanding", *Workshop on Advanced Concepts for Intelligent Vision Systems (ACIVS'99)*, Baden-Baden, pp. 74-79.

Smith, J. W., J. E. Everhart, W. C. Dickson, W. C. Knowler and R. S. Johannes, 1998, "Using the ADAP Learning Algorithm to Forecast the onset of Diabetes Mellitus", *Symposium on Computer Applications and Medical Care*, IEEE Computer Society Press, pp. 261-265.

Smola, A. and B. Schölkopf, 1998, "On a Kernel-based Method for Pattern Recognition, Regression, Approximation and Operator Inversion", *Algorithmica.*

Sneath, P. H. A. and R. R. Sokal, 1973, *Numerical Taxonomy*, Freeman, London, UK.

Spath, H., 1980, *Cluster Analysis Algorithms for Data Reduction and Classification*, Ellis Horwood, Upper Saddle River, NJ.

Stitson, M. O., A. Gammerman, V. Vapnik, V. Vovk, C. Watkins and J. Weston, 1997, "Support Vector Anova Decomposition", Technical Report, Royal Holloway College, Report number CSD-TR-97-22.

Xu, L., A. Krzyzak and C. Y. Suen, 1992, "Methods for Combining Multiple Classifiers and Their Application in Handwritten Character Recognition", *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 22, pp. 418-435.

Ward, J. H. , 1963, "Hierarchical Grouping to Optimize an Objective Function", *Journal of American Statistical Association*, No. 58, pp. 236-244.

Warren, S. S. , 1994, "Neural Networks and Statistical Models", *Ninth Annual SAS Users Group International Conference*, NewYork.

Werbos, P., 1974, "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences", PhD Thesis, Dept. of Applied Mathematics, Harvard University, Cambridge, Massachusetts.

Weston, J., A. Gammerman, M. O. Stitson, V. Vapnik, V. Vovk and C. Watkins, 1997, "Density Estimation Using Support Vector Machines", Technical Report, Royal Holloway College, Report number CSD-TR-97-23.

White, H., 1992, *Artificial Neural Networks: Approximation and Learning Theory*, Oxford, Blackwell, UK.

Widrow, B. and M. A. Lehr, 1990, "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 78, No. 9, pp. 1415-1442.

Wilson, D. R. and T. R. Martinez, 1997, "Improved Heterogeneous Distance Functions", *Journal of Artificial Intelligence*, No. 6, pp. 1-34.

Wolpert, D., 1992, "Stacked Generalization", *Neural Networks*, Vol. 5, No. 2, pp. 241-260.

Zadeh, L. A., 1965, "Fuzzy sets", *Information and Control*, No. 8, pp. 338-353.

Zahn, C. T., 1971, "Graph Theoretical Methods for Detecting and Describing Gestalt Clusters", *IEEE Trans. on Computation*, No. 20, pp. 68-86.