An Investigation into Feature Selection Techniques for Reducing Input Dimensionality in Pattern Recognition Applications

prepared by

Willem Andries Jacobus Nel

Submitted to the Department of Electrical Engineering
in partial fulfilment of the requirements for the degree of

Master of Science in Engineering

at the

UNIVERSITY OF CAPE TOWN

October 1998

Declaration

I declare that this dissertation is my own work. It is being submitted in partial fulfilment of the requirements for the degree of Master of Science in Engineering at the University of Cape Town. It has not been submitted before for any degree or examination at this or any other university.

W.A.J. Nel

Acknowledgements

The work presented herein would not have been possible without the help of many people and the financial support of two institutions. Therefore I would like to express my gratitude towards:

- My supervisor, Professor Gerhard de Jager, for his support, guidance and enthusiasm during the project period.
- Fred Nicolls, for the many discussions on problems pertaining to pattern recognition, for proof reading the work presented here, for enlightening me in the ways of UNIX and for directing me to several relevant sources of information.
- Dr. Brendt Wohlberg for proof reading this document.
- My fellow students of the Digital Image Processing Laboratory for the stimulating, interesting and enjoyable working environment. In particular I would like to thank Praven Reddy, Yon Rosenthal, and Marc Servais for their support and friendship.
- My family, for their love and support during the past two years.
- The Foundation for Research and Development, for their financial assistance.
- DebTech, for their financial assistance and the provision of seriously needed processing power.

Abstract

Pattern recognition systems can suffer from problems of high-dimensionality when inferring decisions from a finite number of data samples. Methods are needed to reduce the input dimensionality of such systems. This thesis investigates the use of feature subset selection techniques to address problems pertaining to high-dimensionality in classification systems.

An investigation into the origins of the "curse-of-dimensionality" is given and a detailed survey of the literature on feature selection techniques are shown. Three methods aimed at addressing the problem of finding the most important features for use by a classifier are identified and investigated.

Mutual information techniques allow testing of the relevance of features on the output variable as well as other features. This allows a ranking of features in order of their importance to the classifier.

The Gamma Test method allows the evaluation of the utility of different subsets. The output of this method is shown to correlate well with expected classifier error rates. This allows the method to be useful for feature selection as well as other areas of application.

A fast neural network technique is employed as third feature selection method. This technique addresses several problems regarding feature selection as applied to neural networks.

The three methods are evaluated and compared on standard data sets from the Machine Learning literature. It is shown that all three methods have different strengths and weaknesses. The techniques prove to be able to reduce the dimensionality of the data sets significantly. It is shown that the use of these techniques in pattern recognition problems can possibly even give rise to higher classification rates.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The question of whether we can make computers think and react to problems in the same way humans do is a long standing one. Much effort has been put into understanding the way humans approach problems of perception and analysis. As human beings we find it very easy, for example, to look at a picture of an outdoor scene and tell the difference between the trees, a house and, in fact, most any other object in the scene. Without thinking about any of this, we use the measurements of our five senses with cunning accuracy. We easily recognise

> a tune heard before on the radio,
> the faces of hundreds of people we know,
> different makes of cars,
> the type of food being cooked in the kitchen.

The list of things seems infinite, yet we do all of this by using only a few inputs and a vast network of interconnected brain cells.

A multitude of these tasks are concerned with classifying into different categories those objects which we perceive with our senses. The field of study that tries to imitate or model this task is called pattern recognition/classification. As stated by Ripley [46] the aim is, in essence, to solve the following problem:

> "Given some examples of complex signals and the correct decisions for them,
> make decisions automatically for a stream of future examples"

In order to perform such a pattern recognition task, the first step then is to find from the problem those inputs or signals that would allow the machine to make these decisions. A major problem, however, is that there exists no simple way of knowing which inputs would allow

the pattern recognition system to yield the desired results. The designers of such systems, therefore, have to use some of their own knowledge and experience to extract features from the process which, according to them, would give the classifier a decent starting point for doing the pattern recognition.

Pattern classification systems can be grouped into two main streams of designs. The one stream is that of *supervised learning*. Here, a human being first classifies all the observations made from the problem, and assigns the correct "class labels" to each of these observations. The classifier is then allowed to "learn" from these examples and aims to find a general solution which, when given *new* samples from the problem, is able to tell what the correct class labels for these new samples must be.

The second stream, that of *unsupervised learning*, desires that the classifier learns, by itself, which of the input observations are alike, and assign similar class labels to these inputs. The learning system must then be able to classify future inputs into the classes which the system itself created from the original data. In most cases these classes or groupings must be assigned in such a way as to make sense to human understanding of the problem.

Be this as it may, the one thing these methods have in common is that it is up to the designer to select signals as inputs for these classification systems. The approach used by most classifier designers are to extract, using their knowledge of the problem, as many descriptive features as possible.

The philosophy behind this approach is that every feature added to the input of the classifier adds some new information not yet contained in some of the other inputs. This should allow the classification system to make better estimates of the exact output class of the inputs.

This method does not however paint the full picture of classifier design. A major problem with this approach is that it does not take into account the manner in which classification systems build estimates of the class labels. Every feature added to the system increases the dimensionality of the space in which these estimates are made and, lacking ample input examples, the space might become too vast for a robust classification system to be built.

It has been observed that classification systems can sometimes suffer from what is known as the *curse of dimensionality*. The classification results become better as more features are added, but beyond a certain point classification results worsen, giving rise to a counter intuitive effect: more information is given to the system, but the system performance decreases[1].

Thus there exists a need to find ways of decreasing the input dimensionality in classification systems by finding the most informative features. With this background stated, a typical classification system can then be represented by the schematic shown in Figure 1.1.

[1] A more detailed discussion of this effect is given in Chapter 2.

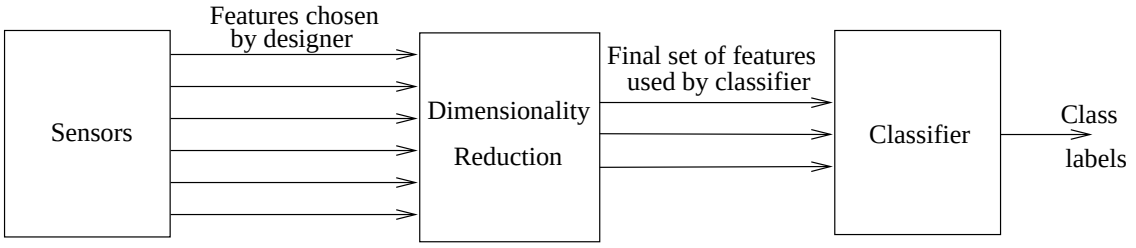


Figure 1.1: The stages of a classification system

In conjunction with the aim of building more robust classifiers, there exist other reasons, not yet mentioned, for reducing input dimensionality. Some of these considerations result from engineering constraints such as limited bandwidths of the channels carrying information from the process being measured to the classification system. The complexity and speed of the designed classifier might also have a direct relationship with input dimensionality, and it is therefore important to use only the most discriminatory features when designing and building classification systems.

The work presented herein belongs to this area of dimensionality reduction techniques. The reduction in the input space are normally achieved in one of two ways. *Feature Selection*, which is the focus of the work presented in this thesis, aims to decrease dimensionality by finding those subsets of the input feature set which contain the most relevant information to the output.

The second related method is known as *feature extraction*. This technique aims to find analytic transformations of the input space that give rise to lower dimensional sub-spaces which still contain most of the information from the original space.

Contrary to first impressions, these two methods do not always stand in opposition to each other, but have different roles to play in the whole process of dimensionality reduction.

1.1 Aim of the work presented here

The first intention of this work was to address issues concerning the curse of dimensionality in pattern recognition systems. It was necessary to seek a deeper understanding of how the dimensionality problem affects pattern recognition systems, and in what ways these effects manifest themselves in different types of classifiers. Of importance here was to find out, either by experimentation or from previous literature, whether all types of classifiers are affected by problems of high dimensional input spaces.

A secondary aim of this investigation was to perform a background study of the area of

feature selection as a means of addressing the problems of dimensionality. Several methods from literature had to be investigated and evaluated in order to obtain an idea of which types of approaches to the problem had shown success in the past. Some problem areas in the previous literature were identified, and solutions to some of these problem areas had to be investigated.

The final aim of the work was to investigate and compare some techniques for doing feature subset selection more thoroughly, and point out their strengths and weaknesses. As will be explained later, the thesis focuses primarily on feature selection as a means of reducing input dimensionality for neural network type approaches, but this does not restrict these methods to this purpose alone. To highlight this fact, the effects of the methods are also compared on the simple k-Nearest Neighbour classifier[2].

In the work presented it was also decided to focus more deeply on the different criterion functions for feature selection rather than concentrating on the search strategies involved in all selection systems. The correspondence between criterion function and goodness of feature subsets were taken to be of primary importance. If the criterion function does not correspond well to the goodness of a specific subset, then no search algorithm would be able to perform well. However, the search function was not totally ignored, and the background study investigates some previous search algorithms from the literature. Portions of the practical experimentation involved using some of these search functions.

In order to restrict the space of possible methods[3] for consideration, and also as a means of addressing problems specifically relevant to the pattern recognition field, it was decided to investigate only feature selection methods that are able to handle continuous range, noisy features. These are the types of features mostly found in machine vision and standard signal processing applications. This restriction was not seen as a disadvantage since most of the discreet non-noisy feature selection methods form a subset of the continuous noisy case. This does imply however, that some methods used for the discreet non-noisy feature case would not be directly applicable to the work presented here.

Lastly, the problems on which the methods were tested and explained focussed only on the two-class problem in classification. This constraint is however not as harsh as it sounds, since the methods discussed are extendable to multi-class problems.

[2] See Chapter 2 for details on different classifiers.
[3] It will be seen from the literature survey that a multitude of feature selection methods exist.

1.2 Layout

In *Chapter 2* the reader is presented with the background on pattern recognition and classification theory. This leads directly to a discussion of the problem of dimensionality and how it manifests itself in different types of classifiers. The last section of the chapter is concerned with explaining different types of search strategies. These strategies address a primary problem in feature subset selection techniques when the number of features becomes too high for exhaustive searches of all subsets.

Chapter 3 goes on to discuss some background on the major types of techniques used for feature space reduction. Firstly, two of the main extraction techniques are described. Some of the advantages and disadvantages of the two techniques are discussed, and it is shown that a need for feature subset selection does indeed exist. The next section of Chapter 3 describes some of the basics of feature selection, and details a literature survey of work done in this field. From this survey two promising methods are identified, and a need for work in the field of feature selection for neural network classifiers is shown.

Chapter 4 details how the information theoretic concept of mutual information can be applied to the feature selection problem. Some implementation details are also discussed, and a simple example problem is shown to illustrate the concepts.

In *Chapter 5* the reader is introduced to the Gamma Test. This technique has some interesting theoretical origins, and has been used previously on feature selection in other scenarios of neural network applications. The method is discussed and applied to some explanatory examples, allowing a deeper understanding of the concepts involved.

The last method to be evaluated is also the only wrapper[4] technique applied to the problem. The Random Artificial Neural Network is introduced in *Chapter 6* as a neural network method that is fast enough to handle many evaluations of the feature selection criterion function, and is thus well suited to address the needs identified in Chapter 3.

Chapter 7 discusses the experiments and comparative tests that were performed using the three different methods. Each experiment is explained and a discussion follows the tabulated results of every experiment.

Instead of giving a code listing as appendix to the thesis, it was decided to make the code that was used available on a CDROM. Therefore, *Appendix A* details the directory structure of the CDROM, and highlights some of the more important files. The CDROM also contains, in PostScript format, many of the papers referenced in the thesis.

[4]See Chapter 3 for details on wrapper techniques.

Chapter 2

Background

This chapter highlights some of the background theory and ideas required for understanding the concepts used in later chapters. An overview of some classification theory is given In Section 2.1. A few different types of classifiers are also described. The next section discusses the problem of the curse of dimensionality in more detail. The section considers some of the ways in which this problem manifests itself in different classifiers and also references some literature concerned with this subject.

The last section of the chapter reviews some standard search algorithms. These algorithms are used in many feature selection techniques when the number of features become exceedingly high. The search techniques employed in some of the experimental work of Chapter 7 are pointed out.

2.1 Classification Theory

Since the early beginnings of the fields of pattern recognition and decision theory, many different techniques have been developed to automate the process of classification. The different methods all have their own angle of attack to the problem, but have the same goal in mind: to classify *new* examples not yet seen by the classifier with the highest possible degree of accuracy. Unfortunately, there does not exist a perfect measure for the accuracy of classification systems, especially since the degree of accuracy is supposed to be measured on unseen examples. However, it is widely accepted that there does exist a method which, if implementable, can build the optimal classifier for any problem. This technique, which also forms a solid basis for understanding the key concepts in classification theory, is discussed first. The drawback of the method however, is that it is almost impossible to implement perfectly and therefore can serve only as a guideline for the obtainable accuracies of classification systems.

2.1.1 The Bayesian Basics

The field of probability theory brought to the problem of classification an approach called Bayesian inferencing. This approach is based on the assumption that the problem can be posed in purely probabilistic terms, *and* that the probabilities of all the different events are known.

The Bayesian approach will be demonstrated by an example with notation similar to that of Duda and Hart [18]. Suppose we want to predict whether tomorrow will be a sunny or rainy day. Because of the random nature of this prediction, we let the random variable ω denote this state of nature. Next, define ω_1 to represent the class of *sunny* days and ω_2 that of *rainy* days.

In the Bayesian approach we assume that we have some *a priori* knowledge about the probability of occurrence of these two classes. Let the *a priori* probability of class ω_1 be $P(\omega_1)$, and that of class ω_2, $P(\omega_2)$. The *a priori* probabilities could have been gathered by recording the occurrences of sunny and rainy days during the last year.

Suppose now that we want to make a prediction about tomorrow's weather. The most logical decision rule (knowing nothing but the *a priori* probabilities we collected) would be to say that tomorrow would be sunny if $P(\omega_1) > P(\omega_2)$ and that it would be rainy if $P(\omega_2) > P(\omega_1)$. At this point, we would now be choosing the class of highest probability for *all consecutive days*, even though we know that both sunny and rainy days occur.

To improve upon this situation, we could try to make some measurements about the current climate conditions in order to make our forecast. Suppose we gather daily the three measurements of highest temperature, average humidity and average wind direction. Considering this measurement vector (feature vector) to to be a random vector $\mathbf{x}$ we can now use this information to find $p(\mathbf{x}|\omega_i)_{i=1,2}$ for both classes. This is known as the *conditional* probability density functions of the classifier. In the example, these densities represent the probability that a certain measurement vector occurs, given the day is sunny or rainy.

Knowing this information allows us to use *Bayes' Rule*:

$$P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)\,P(\omega_i)}{p(\mathbf{x})} \tag{2.1}$$

where

$$p(\mathbf{x}) = \sum_{i=1}^{2} p(\mathbf{x}|\omega_i)\,P(\omega_i)\,. \tag{2.2}$$

$P(\omega_i|\mathbf{x})$ is called the *a posteriori* probability.

It can be shown [18] that the minimum average error decision rule can now be created by

choosing ω_1 when $P(\omega_1|\mathbf{x}) > P(\omega_2|\mathbf{x})$ and vice versa.

This explains the basic principals of Bayesian inferencing. It is generally agreed that the Bayesian classifier is the optimal classifier that can be built for any given problem with known priors and conditional densities. If other risk factors have to be built into the classifier, it can be done by scaling the *a priori* probabilities in the correct manner. The resulting classifier is then called the minimum risk classifier.

The practicality of the Bayesian approach is however very disputable. This dispute is caused by the fact that *a*lmost no real-life problems have well-defined or well-known conditional densities and a priori probabilities. In most cases it is very difficult, if not impossible, to obtain this information perfectly from the data gathered for the problem.

In an attempt to overcome this problem, many different techniques have been developed. These techniques either base their decisions on estimates of the *conditional* or *a posteriori* density functions, or try to avoid making these estimates at all. The methods developed to overcome the unknown density problems have, in the literature, been categorised into two main categories, namely parametric and non-parametric techniques. The following sections will discuss some of these different techniques.

2.1.2 Parametric Techniques

The fact that the optimal classifier cannot be built in most cases has the implication that the designer of a classification system has to estimate, in some way, the unknown density functions. In parametric classification techniques, this is done by first choosing an analytic form for the distributions, and then estimating the parameters describing these distributions from the data being used. The choice of which exact form of distribution to use is made based on samples drawn from the distributions and on the experience of designer of the system. Another choice that has to be made by the designer is whether to try and model the *a posteriori* density functions or the *conditional* density functions. A discussion on this can be found in [46].

In most cases the choices of the forms of the density functions are kept as simple as possible, and standard distributions like the multivariate normal distribution are assumed. Returning to the previous example, the conditional probabilities for the measurements of average daily humidity, maximum daily temperature and average wind-direction might be considered to be normally distributed, each having some mean value, and some variance. If we denote the mean values as μ_{humid}, μ_{temp} and μ_{dir}, and the variances as $\sigma^2_{humid}, \sigma^2_{temp}$ and σ^2_{dir}, we would know have twelve parameters describing the conditional densities (six per class). These parameters would then be estimated using techniques like the maximum likelihood estimate [46, 22].

After the gathered data is used for estimating these values, we would then plug the estimated *conditional* distribution into equations 2.1 and 2.2, and again use the decision boundaries from that procedure to do the classification.

By only estimating the variances of each feature and not finding covariances between the different features it has in this example implicitly been assumed that the features were independent. These correlations might adversely affect the accuracy of our models, and corrupt our final decisions. There could for example be a high degree of correlation between our measurements of temperature and humidity, and this would not be reflected in the model of the system.

Model complexity is one of the big issues in the design of all classifier systems. In many real cases models are kept very simple, even though the designer of the system knows this assumption to be far from the truth. The rationale behind this apparent disregard for the complexity of the problem comes from a trade-off in estimation techniques. This trade-off involves the number of estimated parameters and the accuracy with which they can be estimated, and is known in the parametric estimation literature as the bias/variance trade-off. It will be discussed in a bit more detail as part of Section 2.2.

From the above discussion it can be seen that classification methods using parametric techniques are in many cases very dependent on the type of data being looked at, and require a lot of expert knowledge and interaction in designing and setting up these classifiers. Other objections that have been lodged against these methods are that the simplistic models used in these methods sometimes oversimplify the problem, giving bad results in real applications. Further difficulties arise when the techniques discussed here have to be applied to very high-dimensional problems, because humans are not very well adapted to visualising data in such high-dimensional spaces. This brings about problems in deciding the exact forms of the different distributions.

The factors mentioned above do not imply that there are not a vast community of statisticians that use them daily to design classification systems. However, these factors have come into consideration in deciding what type of classifiers to use in the work presented here. As will be seen from the literature review, most feature selection techniques require that computer-human interaction be kept as low as possible during the search for feature subsets. The fact that some of the other techniques, like neural networks, allow much easier adaptation to problem situations, make them more suitable for use in feature selection systems.

2.1.3 k-Nearest Neighbour Classifiers

In non-parametric techniques, the classifier does not explicitly make an attempt to determine the unknown probability density functions, but rather to estimate good decision boundaries

directly from the data. This is done by using the collected samples in some manner in order to decide where the decision boundaries should be placed.

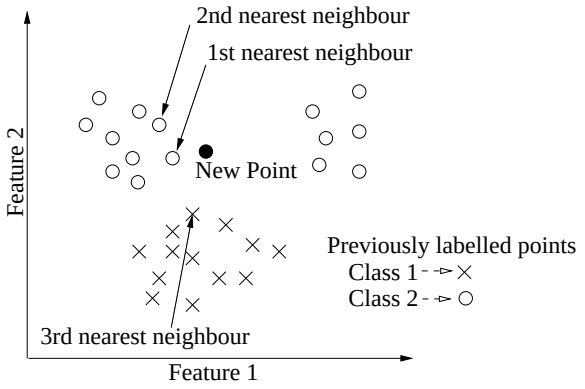


Figure 2.1: Illustration showing simple k-nearest neighbour classification

One such type of classifier, illustrated in Figure 2.1, uses only the nearest neighbours of the new measurement to decide to which class this measurement should be assigned. This technique is known as k-Nearest Neighbour classification. In this technique the class label ω assigned to a feature vector is based on the class occurring most frequently in the k nearest neighbours of this vector. The nearest neighbours are found from points that were already labelled by the designer of the system. In the figure it is seen that, if using a 3-nearest neighbour technique, the new point would be classified as belonging to class 2, since two of the three nearest neighbours to this point were labelled as class 2.

The basic idea on which the success of this classification technique is based, is that measurements lying close in input space in most cases have the same class. It therefore makes sense to use some kind of distance measure in the input space, finding the closest neighbours to the point to be classified, and basing the class decision on the labels of these neighbours. In many cases the distance measure is based on the Mahalanobis distance (see [18]) instead of normal Euclidean distance, in order to remove the dependence of the distance measure on the variance differences in the different features.

One of the problems in k-nearest neighbour techniques is to estimate a good value for k. Basing the decision boundaries on too few neighbours cause too much local variation in the decision boundaries. On the other hand, having too many neighbours prevent the decisions from being adaptive to local changes in the data.

Although the k-nearest neighbour technique does not explicitly estimate density functions, the method does in fact build a piecewise constant model of the posterior distributions. Ripley [46] illustrates this fact through some examples, and it is also shown in Duda and Hart [18].

It has been proven [18] that nearest neighbour approaches are close to optimal when the

number of labelled training data points become very high. In fact, it is proven to be no worse than twice the minimum Bayes error as the number of such points tend to infinity [18]. This has made nearest neighbour approaches one of the standard techniques with which other classifiers are compared.

There exist some arguments that nearest neighbour classifiers are highly affected by input dimensionality problems. This was seen in some of the experiments conducted in this work, and will also be discussed in Section 2.2.

2.1.4 Artificial Neural Networks

A second classifier approach historically falling into the mould of non-parametric classification techniques is that of Artificial Neural Networks (ANNs). In recent years these types of classifiers have gained mixed support from scientists working in the field of classification theory. Newcomers to the field often use these classifiers as quick-fix answers to the problem at hand and expect them to perform miracles. Like all classifiers, ANNs do however suffer from some drawbacks. They can be highly susceptible to over-fitting and are not transparent to the user, thus making their performance for unknown data somewhat unpredictable.

The ANN was inspired by the chemical functioning of neurons in the brain, but has since grown into a field that is not highly related to its biological origins. The first ideas on ANNs were developed in the early 1960's by Widrow & Hoff and Rosenblatt (for references see Ripley [46]). In the mid 1980's the field was revived and suddenly started receiving attention from, and finding application to, many areas of research.

The main ANN in use today is known as the feed-forward ANN. According to Ripley [46, p.143] it consists of a network of units each of which has one-way connections to other units in the network. Each unit takes as input some value, performs a function on this input, and outputs the calculated value. The units are arranged in layers so that each unit is connected only to units in a later layer. A sample network is illustrated graphically in Figure 2.2

Every unit in the network takes the sum of all its inputs to form a total input x. The unit function f is applied to x which forms the output y. This is propagated through the output connections by multiplying the output y with the weight w_k of each connection on the output of the neuron. In many networks each node is provided with a constant (bias) input.

The network shown in Figure 2.2 can thus be represented by[1]

$$y_k = f_k\left(\alpha_k + \sum w_{jk} f_j\left(\alpha_j + \sum w_{ij} x_i.\right)\right) \tag{2.3}$$

[1]Here i, j, k denotes entities in the first, second, and third layer respectively

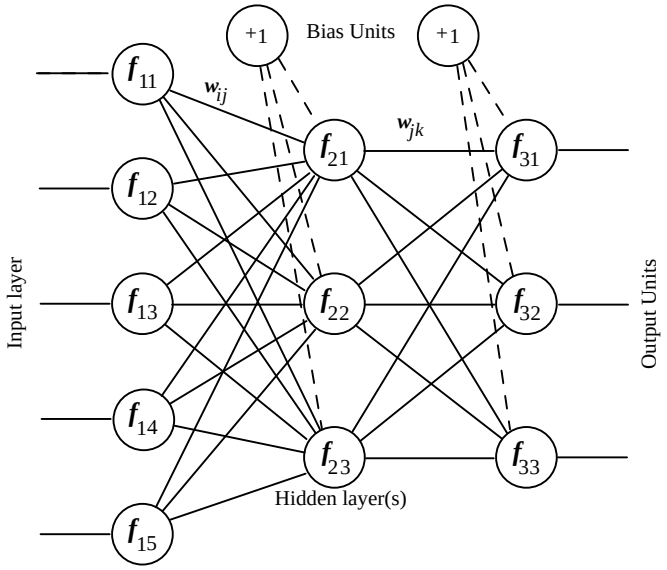


Figure 2.2: The feed forward artificial neural network structure.

The input units to the network are normally just used to distribute the input data and therefore have linear transfer functions. The second layer of the network are normally called the *hidden* layer and mostly have either linear, sigmoid or hyperbolic tangent transfer functions. The output layer can sometimes contain a step function (especially when used in the context of classification theory) but this is usually not shown explicitly in the model.

The general model can be extended in several ways not discussed here. These include different transfer functions, feedback connections, inhibitory connections, and more.

Artificial neural networks have the ability (when given enough hidden neurons, even with only linear transfer functions) to approximate any function arbitrarily well (see Ripley [46, p.147]). The purpose of an ANN in classification theory is therefore to find some function that forms a well-suited decision boundary between the different classes of the problem. Using non-linear transfer functions, these networks can sometimes give surprisingly good results on some data sets. This has led many people to blindly use neural networks for every problem at hand.

In order to find the decision function that suits a particular problem well, the idea is to adjust the weights w_{ij} and w_{jk} in order to minimise some criterion function (typically Minimum Mean Square Error). Many algorithms have been developed for doing these adjustments of the weights. The first rather successful approach was that of back-propagation. After this many other algorithms and modifications ware made, and at present a multitude of these algorithms can be found in most neural network software packages. A discussion on some algorithms can be found in Ripley [46] and Haykin [24], and some implementations of these

algorithms are found in [15].

It must be stated that many of these algorithms take a very long time to find a solution to the problem, and even then, these solutions do not imply optimality (a local optimum is mostly found). Furthermore, ANNs suffer from what is known as over-fitting: The decision boundary created by the neural network might be very good at separating the training data classes, but when new data is presented to the classifier it struggles to generalise. This is a problem very similar to the over-fitting problem in function estimation by polynomials: If the degree of freedom of the polynomial is too high, the fitted function represents the data well, but does not interpolate smoothly between points.

It is therefore necessary to search not only for the optimal weights, but also for the optimal network structure when designing ANN classification systems. The number of inputs and number of hidden neurons jointly determine the number of parameters that has to be estimated in building the classifier[2]. This implies searching for both the optimal number of hidden neurons and the best inputs to use with the network. Thus, it is seen that neural networks also call for some method of feature subset selection to decrease the number of inputs, and thereby (perhaps[3]) the number of parameters to be estimated.

A major problem in searching for different network topologies is that it is very difficult to train neural networks very quickly. This will be discussed further in the next chapter as well as in Chapter 6.

2.1.5 Other Techniques

The field of pattern recognition has, of late, become a vast field with many new ideas being added in recent years. The techniques for classification mentioned thus far are not at all a full representation of all the available methods. Many of the feature selection algorithms discussed in the literature survey of the next chapter use other classification techniques.

Of these, the two outstanding classification techniques worth mentioning here are that of decision tree classification and fuzzy logic inferencing. Decision tree classifiers use a rule based approach for finding different classes from the data. The computer sets up these rules automatically by building, from the training data, a decision tree structure that, if followed for new examples, will hopefully give the correct class for the data. These algorithms are

[2]It can be seen that the terms parametric and non-parametric classifier are not totally apt for distinguishing between the different classification techniques. In a sense, neural networks are also parametric in that they have many parameters to estimate. The different classifiers should rather be categorised to show whether they estimate density explicitly or not.

[3]Decreasing the number of inputs *might*, in some cases, lead to a more difficult problem which needs more hidden nodes to solve. This will, however, only be apparent after investigating the effect feature subset selection has on the network.

best suited to cases where the input variables are discreet, non-noisy inputs. Some of the techniques have however been extended to continuous cases. This field of recognition is also known as syntactic pattern recognition. A interesting discussion on these types of classifiers can be found in Ripley [46].

The use for these types of classifiers is most common in areas where human understanding of the classification approach is essential. The path followed to every decision through the decision tree can be reconstructed, and thus erroneous classification can be ascribed to certain nodes in the decision tree.

The reason so many feature selection systems in modern literature use these tree based classifiers seems to be the fact that different features can be tested by pruning of the decision trees, and the effects of including certain features can easily be investigated. The work presented here, however, does not include the use of these classifiers, due to the fact that many of these methods cannot handle noisy continuous features[4] (features that can have many values for the same class, and also show class overlap in the input space) and that recent comparisons amongst these methods already exist in the machine learning literature.

Another field of classification techniques is that of Fuzzy Logic Inferencing [26, 35]. The idea behind these systems is to use fuzzy reasoning in order to find appropriate class labels. Classes might be labelled by the degree to which they belong to that class instead of using hard labelling. Inputs are also made to indicate the degree of membership to a certain rule. A summary of fuzzy logic pattern recognition falls beyond the scope of this introduction.

Implementing all of the classification schemes for every problem would be an impossible task. However, many people are of the opinion that applying a different technique normally does not imply a sudden leap in the classification accuracy. For this reason it was decided to use only the neural network and k-Nearest Neighbour techniques in the experiments presented here. At the end of Chapter 3 and in Chapter 6, other reasons for investigating the neural network approach are pointed out. The k-Nearest Neighbour approach is used as a reference for comparing the results obtained using the neural network technique.

2.2 The Curse of Dimensionality

Although there exist such a multitude of different classification techniques, one thing that stays common to all is the fact that they must infer knowledge from the same inputs if used in the same situation. This inherently leads to an underlying fact that all of these methods

[4]This does not mean that there do not exist *extensions* to some of these algorithms which are able to handle noisy features. It implies simply, that most of the feature selection methods reported in the machine learning literature do not handle noisy features.

are affected in some way by the dimensionality in which the knowledge inferencing is done.

In many practical situations it is found that a large number of input measurements are made from the process that needs to be categorised or classified. As was stated in the introduction, the reason for this is the human instinct to think that more information leads to more accurate decisions.

Looking at the Bayesian approach to classification, there are some theoretical results that support this intuition. One such result stated in [18] comes from the two-class multivariate normal problem. Assume that the input features are statistically independent, and the conditional density function $p(\mathbf{x}|\omega_i)$ is normally distributed with mean μ_i and covariance matrix Σ. If the a priori probabilities are equal, it can be shown [18] that the average probability of error is

$$P(e) = \frac{1}{\sqrt{2\pi}} \int_{r/2}^{\infty}, e^{-\frac{1}{2}u^2} du \tag{2.4}$$

where r is the Mahalanobis distance[5]

$$r^2 = (\mu_1 - \mu_2)^t \Sigma^{-1} (\mu_1 - \mu_2) . \tag{2.5}$$

From this it can be seen that $P(e)$ decreases as the distance r increases, approaching zero as r approaches infinity. For the case of statistical independence between features, Σ is the diagonal matrix with the variances $\sigma_1^2, \sigma_2^2, ..., \sigma_n^2$ along the diagonal.

This then changes the Mahalanobis distance to

$$r^2 = \sum_{f=1}^{d} \left(\frac{\mu_{f1} - \mu_{f2}}{\sigma_f^2} \right)^2 \tag{2.6}$$

From Equation 2.6 it can be seen how each feature contributes to the probability of error. The most useful features have large differences in mean relative to the standard deviations. It can also be seen that, for this multivariate normal case, adding any independent feature which has a difference in mean for the two classes will reduce the probability of error slightly. Thus repeating this procedure indefinitely will allow the designer to make the probability of error as small as necessary.

By this and many other observations, it has been argued that adding any feature to the optimal Bayes classifier can only improve the performance of the classifier. Features adding no new information will just be ignored, and those that add information will help to decrease the error rate of the classifier.

[5]The Mahalanobis distance measure is sometimes used instead of normal Euclidean distance. Mahalanobis distance is used because it normalises distances according to the variances along the different dimensions, thus taking out effects of variance differences amongst different features. The distance measure also has the advantage of taking correlation amongs variables into account.

In practice however, a different picture has been found. It has been observed in many problems that beyond a certain point, the inclusion of more features causes a performance decrease rather than increase. This problem has been observed in many fields of study. Kittler [28] refers to it as the *peaking phenomenon*. The field of density estimation calls it *the curse of dimensionality* [49] and Duda and Hart refer to it as "Problems Of Dimensionality" [18].

Several attempts have already been made to try an explain how and why this counter intuitive effect occurs. In statistical estimation theory, it is sometimes related to the so-called bias/variance trade-off [21, 22, 46]. It has been shown that for fixed data size, there exists a trade-off between the accuracy with which parameters can be estimated, and the number of parameters estimated. In pattern recognition, this has an effect on the size of the input space that can be handled, since every feature that is added to this input space brings with it more parameters to estimate. In the parametric methods the covariance matrices that have to be estimated grow rapidly in size. In neural networks, more weights have to be calculated from the available data. In Bayesian techniques, the estimation of the probability density function has to be carried out in higher-dimensional spaces, also adding more parameters to be estimated.

The k-Nearest Neighbour approaches are another set of classifiers prone to suffer from dimensionality problems. This can be seen by looking at the distance between any two points in feature space. If we consider adding another feature, then under most distance metrics the distance between the two points can only increase (or at best stay constant) after adding this new feature. If this feature contains a high degree of noise, it will affect the distance measure between all points in the space. Many such features will eventually sow havoc in the classifier.

It is believed that all modern day classifiers suffer in one way or another from this *curse of dimensionality*. The book by Duda and Hart [18] has an interesting discussion of the problem: They include a result from earlier authors describing an analysis pertaining to the class of all classification problems, and show that if the number of samples for any given problem is fixed, there exists some point after which increasing the number of features has a negative effect on the error rate of a classifier.

According to Jain and Chandrasekaran [25], for the two-class multivariate normal problem there exists a condition that can indicate whether the peaking phenomenon will occur. They show that, to avoid peaking, the minimum Mahalanobis distance between two classes should increase more than a certain threshold every time another feature is added. This threshold is proportional to the number of samples in the data set, the number of features already present, and the within-class scatter added by the feature. In a sense this implies that the new feature should, on average, add more distance between the classes than it adds to the within-class scatter. This result has not been extended to non-normal cases, and it has to be pointed out that estimation of the normal densities is also erroneous if the number of available samples

is too small, thus making the result somewhat impractical. Jain and Chandrasekaran also discuss some of the effects of the peaking phenomenon on k-Nearest Neighbour classifiers.

The effect of dimensionality has also been studied in some detail in the field of density estimation. Scott [49] contains a whole chapter illustrating some of these effects on probability density estimates. It is argued that kernel density estimation techniques cannot be used in high-dimensional spaces with a high rate of success. This is mostly due to the sparseness of data in high dimensions. The effects of this problem on some simple examples from the field of density estimation is also investigated. Another interesting example is found in Fukunaga [22], where it is shown that the sampled multivariate Gaussian density function shows some peculiar behaviour in high-dimensional spaces.

One of the biggest hurdles to understanding this problem is that it mostly does not occur in simple two dimensional cases, and it becomes very difficult to visualise in higher-dimensional spaces. However, the existence of this problem is one of the main thrusts for research such as the work presented here, that aims to reduce the effects of the problem of high-dimensional spaces on pattern recognition tasks by reducing the input dimensionality.

2.3 Search Algorithms

Most feature selection algorithms have to employ some method of searching the space of input features. When the number of features becomes exceedingly high, exhaustively testing all combinations of inputs becomes computationally expensive, causing the required run-time of tests on all subsets to become prohibitive. For this reason many selection search routines have been developed, some specifically for feature selection, and some from general formulations of search algorithms for optimisation.

2.3.1 Greedy Search Methods

Many "greedy" search methods have been developed in the field of feature selection. These techniques find good, but non-optimal sets of features. The methods are sometimes called "greedy" methods because they prevent access to some points in the search space and mostly find only sub-optimal solutions.

Forward Selection and Backward Elimination

Forward Selection starts out by assuming the empty set as the starting set of features. Next, the best feature according to some criterion is added to this set. This process is continued

until a preselected number of features has been reached, or some other stopping criterion has been satisfied.

In *Backward Elimination* the above process is reversed. Starting with the full set of features, the worst feature in the set, according to some criterion, is removed. The process is continued until the stopping criterion is reached, which can again either be a pre-specified number of features, or some other evaluation function.

Plus l- Take Away r Search ($+l-r$)

These methods are based on that of Forward Selection and Backward Elimination, but were designed to improve some of their bad characteristics. The problem with strictly sequential methods are that some of the nodes in the search space becomes hidden after a selection has been made. Representing the chosen features with a 1 and non-chosen ones with a zero, it can be seen for example that after selecting 0100 as the first node with Forward Selection, any nodes containing a zero for the second feature cannot be visited. This has the adverse effect that complex interactions between features cannot be investigated, which is particularly bad in the case of non-independent features.

In the $+l-r$ method, higher orders of interaction are considered by allowing features to be added to, and removed from, the currently selected set. For example, if the process is started with the empty set, the first step is to add the l best features according to some criterion, and then to remove r of these features by investigating some interactions in the newly selected ones. This process is then repeated until the needed number of features is reached, or some other stopping criterion is met[6].

2.3.2 Branch-and-Bound Technique

The branch-and-bound technique [41, 28, 22] is a search technique developed for exploiting monotonicity in criterion functions. The technique was specifically formulated for selecting the optimal subset of *known* size s from a feature set of size F. The method is guaranteed to find the optimal subset of this *known* size for criterion functions that adhere to the set inclusion monotonicity principal described below (see also [22]).

If we denote a candidate feature set that contains s features as χ_s, then the monotonicity criterion implies that for nested sets related by

$$\chi_1 \subset \chi_2 \subset ... \subset \chi_s \subset ... \subset \chi_F \tag{2.7}$$

[6]The process can also be reversed by starting with the full set of features.

the criterion function $J(\chi_s)$ used for selecting the features must satisfy

$$J(\chi_1) \leq J(\chi_2) \leq ... \leq J(\chi_F)\,. \tag{2.8}$$

The branch-and-bound algorithm works by excluding from the evaluations all subsets of χ_s once this set was found to be worse than the previous best set.

In many cases even using this technique might become computationally expensive. For example, if the size of the feature subset is not known beforehand, the test must be performed for a number of different sizes. Another difficulty is that many criterion functions do not satisfy the criterion of Equation 2.8. This is true especially for the "Wrapper" approaches discussed in the next chapter.

2.3.3 Stochastic Search Methods

The field of stochastic optimisation brings to feature subset selection yet another technique of searching for the optimal set of features. Stochastic search techniques are based on the notion that, in many search spaces, the optimal set lies close to other sets that also perform well. The basic idea then is to randomly span the search space with evaluations, and then search more closely in areas that contain the most promising results.

Genetic Algorithms

One such stochastic technique comes from the field of Genetic Algorithms (GA's). These algorithms were developed by J. H. Holland in the 1970's. They stem from the concept of natural selection, the biological process by which stronger individuals have a better chance of survival in a competing environment. A short description of how these algorithms can be used in feature selection follows. For a more detailed discussion on GA's see [38].

Again, we represent a trial solution to the problem as a binary number, with a **1** representing a selected feature and a **0** representing a discarded feature.

The algorithm starts by randomly selecting a population of such binary numbers of length [7] F. Each member of the population is then evaluated using the criterion function. The next step is to select from these solutions a number of "parents" that will be used in finding new trial solutions. These "parents" are selected so as to represent the "best" solutions to the problem found in the current population.

Next, the "parents" are used in a recombination process to rebuild the population. This

[7] As before, F denotes the number of features in the full set.

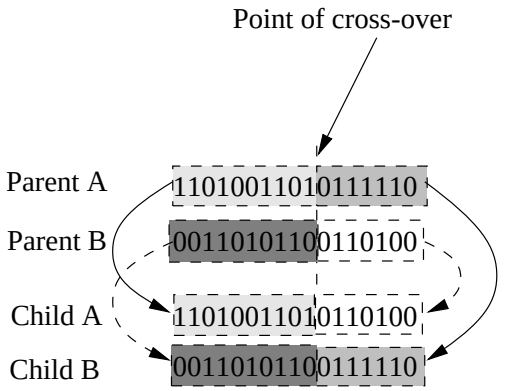


Figure 2.3: The single-point cross-over process used in standard genetic algorithms

is done by using the "parents" in pairs, and building new solutions as combinations of the "parents". An example of such a recombination shown in Figure 2.3 serves as a description of this process. The cross-over point is normally selected at random during the recombination process.

After recombination, a process called mutation is sometimes performed. This has the effect of altering, with a certain probability, some of the bits in the encodings.

The newly created population is then evaluated on the criterion function again, and the "best" members of the whole process forms the new population. From here the process starts afresh and is normally performed until the population has converged to some individual considered to be the optimal solution for that run. This solution is not necessarily the optimal solution for the test function, but forms a "good" solution to the problem[8].

Population Based Incremental Learning

In recent years a lot of criticism has been lodged against Genetic Algorithms, and other algorithms which stand in opposition to the GA have been developed. Much of this criticism come from the fact that the recombination operators used in GA's have certain biases which cause the search to be biased towards particular areas of the search space.

Efforts have therefore been made to remove the genetics from genetic algorithms. One such algorithm is Population Based Incremental Learning (PBIL) [4]. This algorithm takes a more probabilistic approach to the problem.

The algorithm as described in [5] is:

[8] As previously stated, the description of the GA given here serves only for illustrative purposes. For more detail on the exact algorithm see [38].

1. Initialise the probability vector to its equal state.

   The probability vector is a vector that contains, for every feature, a value stating the probability that the feature should be included. For example, if we have ten features then the initial probability vector will be:

   $P_f = \{0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5\}$

2. Generate a population using the probability vector and evaluate each member using the criterion function

3. Find the best solution χ_{best} in the population

4. Move the probability vector towards this best solution by adjusting the probability vector as follows:

   $P_f(i) = P_f(i) + (1.0 - \alpha) + \chi_{best} * \alpha$

   For example, if the best feature set found in the first population of the ten feature example was 1101010110, and the learning rate was $\alpha = 0.1$ then the new probability vector after the first execution would be

   $P_f = \{0.55, 0.55, 0.45, 0.55, 0.45, 0.55, 0.45, 0.55, 0.55, 0.45\}$

5. return to step 2

The algorithm works by searching with a higher and higher probability in the space around some good solution, and in the end converges when the probability vector contains only values very close to 1 and 0.

This algorithm has been shown to perform very well on problems for which genetic algorithms should have been more suited [5]. For this reason an implementation of PBIL is used in some of the feature set searches conducted on the criterion functions in this study[9]. However, it must be stated that the main content of the work presented here was not focussed on finding optimal search algorithms, but rather on finding applicable criterion functions for use in feature set evaluation. For this reason, many of the problems were chosen to have few enough features to be able to do exhaustive searches on the subsets.

The next chapter reviews some of the previous work done in the field of dimensionality reduction and introduces the standard ideas used when performing feature subset selection.

[9] More detail on the exact form of the algorithm can be found on the attached CDROM.

Chapter 3

Feature Extraction and Selection

The introduction and background have shown that the need exists for finding a lower dimensional representation from a set of input features. This representation should contain only the most relevant information needed for classification purposes. In this chapter some of the previous work from the literature on lowering input dimensionality is investigated.

Section 3.1 gives a brief introduction to the field of *feature extraction*. Standard methods are discussed and their advantages and drawbacks are shown.

Owing partly to some of the disadvantages of feature extraction and partly to engineering constraints, the next section discusses the need for feature selection techniques. It is shown that in some cases feature extraction is not an ideal solution. Feature selection techniques can then be used to build more robust classification systems.

Section 3.3 first presents a system diagram of the standard *feature selection* techniques, followed by a survey of previous work in the area of feature selection. Instead of being a fully chronological discussion of the different methods, the section first gives some historical details and then divides the methods into several different categories, depending on the approach employed in each method. The first broad categorisation is that of *wrapper* and *filter* approaches (as proposed by [30]). *Wrapper* approaches are those approaches that use the apparent error-rate of the classifier for determining the best subsets. *Filter* approaches use some other measure of class distance or separability to determine the best subsets. Both of these methods have their advantages and disadvantages, some of which are discussed.

The chapter ends of with a discussion of some of the properties considered to be important for the work presented here. This discussion leads to the three methods that were finally chosen for deeper investigation.

3.1 Feature Extraction

The field of feature extraction is concerned with analytic methods for finding *transformations* of the input space that lower dimensionality and improve classification accuracy. As will be shown in the next two sections, there exists a subtle difference between finding transformations that *represent* a certain data set well, and finding transformations that *discriminate* well between different classes. When trying to find transformations for pattern recognition purposes, they should be selected not to represent the data well, but to keep the discriminative information of the data intact.

In literature, however, both methods using representative biases and those having discriminative biases are used to build classifiers with smaller input dimensionality.

3.1.1 Principal Component Analysis

One method for finding reduced dimensionality input spaces that only contains the most expressive features is that of Principal Component Analysis (PCA) [27, 22, 56]. The analysis can be performed using a transformation known as the Karhunen Loéve transform (KLT) [36].

If we represent the feature vector as a random vector $\mathbf{x}$, the KL-transform of the input space is defined as the solution to the eigenvalue problem

$$\Lambda = \Phi^T \Sigma \Phi \tag{3.1}$$

where Σ is the covariance matrix of the random vector $\mathbf{x}$[1], Φ is the eigenvector matrix of Σ and Λ is the corresponding diagonal matrix of eigenvalues.

In PCA an approximate solution to this problem is found. Only those eigenvectors corresponding to the n largest eigenvalues are chosen. The new random feature vector can then be written as

$$\mathbf{y} = \Phi_n^T \tilde{\mathbf{x}} \tag{3.2}$$

Here $\tilde{\mathbf{x}} = \mathbf{x} - \bar{\mathbf{x}}$ is the mean-normalised input feature vector and Φ_n is a sub-matrix of Φ containing only the eigenvectors corresponding to the n largest eigenvalues. The value of n must be determined by the engineer. Several criteria for choosing n exist, some of which are

- the mean-square-error in representing the data
- using only eigenvectors of eigenvalues that are larger than some percentage of the of the maximum eigenvalue.

[1] The covariance matrix is calculated from the combined data of the different classes.

Principal component analysis can therefore be seen as a *linear* transformation $\mathbf{y} = \mathcal{T}(\mathbf{x})$ of variables to find the vectors that have the most influence on expressing the data.

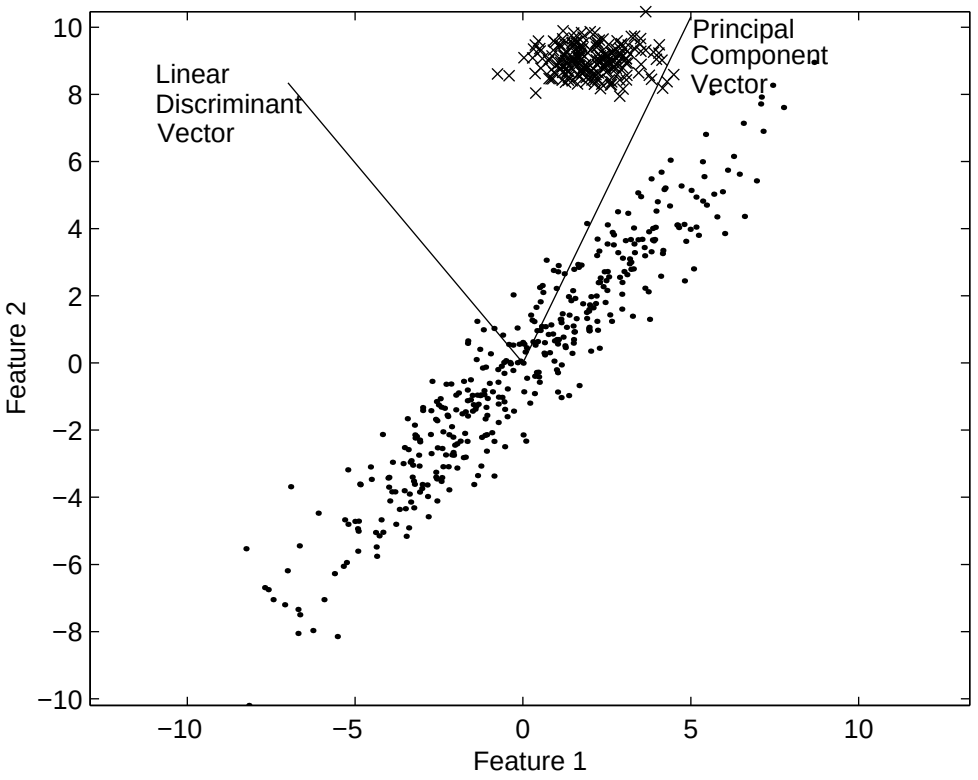


Figure 3.1: An example illustrating the use of principal component analysis and linear discriminant analysis on a simple two feature problem.

Figure 3.1 shows the first principal component that would be extracted using this method on a simple two-dimensional problem. From the figure it can be seen that the first principal component is that vector which accounts for most of the variance in the data. However, if the data is projected onto this principal component, it can be seen that it is impossible to find a point along this vector which would allow complete discrimination between the two classes. Thus the principal component does not necessarily give the best transformation of the data if the aim is to discriminate between the two classes. The procedure presented in the next section allows the optimal discriminatory features (in the sense of linear transformations on unimodal normal distributions) to be found.

3.1.2 Linear Discriminant Analysis

In Linear Discriminant Analysis (LDA) [18, 22, 56], the emphasis is shifted from finding representative features, to finding discriminatory features. By using the within-class, between-class and mixture scatter matrices, transformations can be found that maximise the discriminatory value of a feature projection according to some criterion.

Define Ψ as a projection matrix which projects a feature vector $\mathbf{x}_j$ into the most discriminating subspace yielding the new vector $\mathbf{w}_j = \Psi^t \mathbf{x}_j$. To calculate Ψ, first define the *within-class scatter matrix* of the input data $\mathbf{x}$ as

$$S_{wc} = \sum_{i=1}^{c} \sum_{j=1}^{n} (\mathbf{x}_j - \mathbf{m}_i) (\mathbf{x}_j - \mathbf{m}_i)^t \tag{3.3}$$

Here, $\mathbf{m}_i$ represents the mean-vector of the i-th class and n is the number of training samples in the data.

Also define the *between-class scatter matrix* as

$$S_{bc} = \sum_{i=1}^{c} (\mathbf{m}_i - \mathbf{m}_0) (\mathbf{m}_i - \mathbf{m}_0)^t \tag{3.4}$$

where $\mathbf{m}_0$ represents the grand mean vector[2].

In order to find Ψ it is now necessary to assign a number to these matrices which, in some way, will maximise the between-class distance whilst minimising the within-class distance. One such example [56] is the criterion function

$$\frac{\det S_{bc}}{\det S_{wc}} \tag{3.5}$$

This ratio can be maximised by using as the row vectors for Ψ, the eigenvectors of $S_{wc}^{-1} S_{bc}$ with the largest eigenvalues.

Figure 3.1 shows the best discriminating feature found in this way for the simple two-dimensional data set. From the figure it can now be seen that a projection of the data onto this feature yields a set of points for each of the two classes which are completely separable. The linear discriminant vector therefore allows a single thresholding function to be able to discriminate between the two classes. On this simple problem, and many other problems found in real life, linear discriminant analysis is much better than principal component analysis at finding projections that allow the classifier to discriminate between classes easily.

[2]The vector of means containing the mean value of every feature, regardless of class, over all data, that is $\mathbf{m}_0 = E(\mathbf{x})$

This discussion points out a fundamental difference between dimension reduction techniques used in pattern recognition, and compression techniques used in other signal processing applications. In most other applications, the aim is to find those values,parameters or projections that have inherent ability to describe the data, with as few numbers as possible. In pattern recognition, the aim is to find those features that best discriminate between the different classes contained in the data, possibly even discarding information that is common to all classes.

3.1.3 Other Methods

LDA is by far not the only method used in the literature for finding discriminative feature transformations. Other methods include the use of non-linear analysis, and some methods use no parametric approaches whatsoever. In one such method, Lee [34] uses the decision boundaries created by *any* classifier as a means of obtaining the necessary feature transformation, and then builds a more optimal classifier using his newly designed features. An extensive discussion on some other methods can also be found in Fukunaga [22].

3.2 The Role of Subset Selection

In view of the fact that there exist transformations which have the ability to effect substantial reduction of input space dimensionality, a question that can arise is:

"Why then is there a need for feature subset selection?"

Although feature extraction techniques are good for finding discriminative feature sets, these methods are still a far cry from being perfect for all problems. With LDA, the first restriction is that the method only finds linear transformations, and that non-linear ones make the approaches very difficult to optimise. LDA also relies on using estimates of the means and variances of the data, thus implicitly assuming a unimodal distribution for each class in the n-space of the inputs. This assumption might not be true and can thus yield very misleading results. (See, for example, some of the cases in Fukunaga [22] where it is shown how LDA can "misbehave".)

Some other problems from the engineering point of view are that LDA does not allow the classification engineer the luxury of still having the original meaning of the input features. Also, if cost and speed is a consideration in the design of the system, these methods have even more drawbacks:

- the reduced dimensionality comes at the cost of having more processing in the pre-classifier stage
- no single feature can be omitted totally from the inputs. This means that all the sensors that were used in the experimentation stage of the system design must still be used when the system is put to use.

Some of the present-day classifiers can also be considered to have, as their first stage, an implicit feature extraction phase. This is particularly true of almost any type of artificial neural network classifier. Sometimes using feature extraction can actually hamper the feature extraction stage of a neural network classifier.

Feature Selection techniques do not suffer from some of these disadvantages. The original meanings of features are retained, since only the irrelevant ones are eliminated. During operation, there are no extra burdens on the computational side (the selection is done only in the design stage of the classifier). The cost of the system is kept to a minimum by removing transducers that are not necessary for the classification task. A further advantage is that many feature selection algorithms do not make assumptions about the modality and distributions of the input data.

Feature selection techniques can sometimes also be employed before doing feature extraction. This is invariably necessary if the designer of the system does not have an idea of the relevance of every feature that was extracted from the problem domain. The interplay between different features may also result in peculiar results, where two features might seem to be irrelevant by themselves, but used together they form a good sub-space for classification.

These, and other, considerations show the necessity in classifier design, to evaluate and consider different subsets of the extracted feature set. Not only may this lead to higher classification rates, but perhaps also cost reduction in the implementation stage and more robust classification.

3.3 Feature Selection in the Literature

The field of feature subset selection has been studied widely for many years. References in the classical pattern recognition as early as the 1960's can be found (see for example the references contained in [28]). The subset selection problem shows a combinatorial increase in complexity as the number of features is increased, i.e. the number of subsets to search equals $2^n - 1$ where n is the number of features in the feature set. This fact made the problem very difficult to analyse in these early days due to the unavailability of high speed processing power.

Before discussing the literature, the different components of most feature selection methods should be pointed out. A block diagram showing the components and their interactions is shown in Figure 3.2.

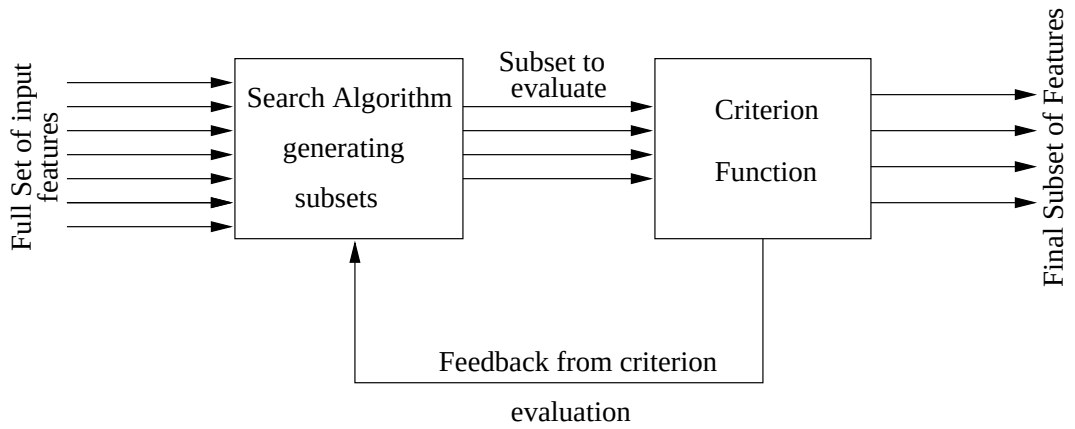


Figure 3.2: A block diagram illustrating the layout of most feature subset selection routines.

The two main components of most of the subset selection techniques are the search algorithm and the criterion function. The search algorithm successively generates the next subset to be evaluated, while the criterion function gives an indication of how suitable a specific subset is for use in the pattern recognition system. In many of the early systems, the desired dimensionality also formed an input to the feature subset selection system. In later research, however, this practice was changed since it is thought that the subset selection algorithm should be able to output the *best* embedding of *any* size.

The earliest methods were mostly concerned with using non-optimal selection/search strategies such as forward selection, backward selection, and floating selection methods[3] [28, 22]. Furthermore, they were mostly concerned with the parametric classification methods, using Gaussian distributions and some model-based distance measure as criterion function. A good introduction to these types of methods is found in Kittler [28]. In this article Kittler presents a comprehensive summary of the earliest work done in this area. He discusses some of the standard approaches, and categorises the different methods into sub-classes based on the criterion function used (often some kind of distance measure) and the search technique employed. The distance measures included probabilistic distance measures such as the Chernoff, Bhattacharya, Matusita and Patrick-Fisher distances, entropy measures and interclass distance measures.

The problem of minimising the search space was addressed by methods that use monotonic criterion functions and applied branch-and-bound [41] techniques. The branch-and-bound

[3]These were discussed in Chapter 2. The name floating selection is used sometimes to refer to the $+l-r$ selection techniques.

technique (as discussed in Chapter 2) was developed by Narendra and Fukunaga, and allows the search space to be considerably reduced when a criterion function satisfying the set inclusion monotonicity property exists. It was, however, soon realised that these methods are not able to deal with the problem of the so-called peaking phenomenon, where the full set of features does not give the highest value to the criterion function (and thus breaks this monotonicity assumption).

With the advent of high speed computing power and the immense increase in the size of databases, a more recent field of study, that of Machine Learning, also reached the stage where feature subset selection techniques became essential. This lead to renewed research in the area of subset selection, this time coming more from the area of soft computing. A survey of techniques from this area can be found in Dash and Liu [14].

Many of the methods presented in [14] are however not totally applicable to the problem of classification from the engineering point of view. These methods use a boolean perspective on the problem, and only work with features that have non-noisy binary, discrete, or nominal values. The survey in [14] shows many such examples. These methods include those that use boolean consistency measures such as Focus [2] and methods which uses probability estimation using binary histogram binning such as Lovell [37]. In fact, in the survey by Dash and Liu, it seems as though only two of the evaluated methods are able to handle noisy continuous data.

With this historical background stated, some of the relevant methods will now be discussed under the categories of *wrapper* and *filter* approaches as introduced by Kohavi in [30].

3.3.1 Wrapper Approaches

In the *wrapper* approach to feature selection, the apparent error rate of a classifier is used as a criterion function for the feature subset being tested. The apparent error rate is calculated by using a training sample from the data. The reason for the name "wrapper" approach comes from the fact that the whole selection process is now wrapped around the classifier, and almost becomes a little black box to be used (and abused) by anyone. Thus, instead of the criterion function being some invented measure of goodness for every subset, the apparent error rate is used to measure the performance of a specific subset.

The wrapper technique is the most intuitive and easily implemented method for doing feature subset selection, and has therefore been studied widely in literature. The biggest problem with wrapper methods are that they are very time consuming. The classifier must be trained or rebuilt for every subset being tested. If the selection strategy then needs to perform an exhaustive search of the feature space, it quickly becomes impossible to search more than about 10 or so features (which already implies building 1023 different classifiers). For this reason many of the wrapper approaches uses a search strategy other than exhaustive search

in obtaining the best feature set.

The wrapper approach does however have a distinct advantage in that the technique guarantees that the biases of the evaluation function and those of the final classifier being used are the same. Filter techniques (as discussed later) do not have this advantage in most cases.

A discussion of methods employing different search techniques follows.

Greedy Search Selection Techniques

Most selection techniques from the Machine Learning field employ instance based decision tree structure classifiers and greedy selection algorithms.

Caruana [10] employs a method that uses a hill-climbing search strategy and two *decision tree* type classifiers to find more optimal feature subsets. The search methods employed include forward selection, backward selection and floating selection techniques. It is widely known that these selection techniques might sometimes wrongly occlude a part of the search space because each step does not have access to all possible feature interactions.

Frasca [19] shows how Rough Set Theory can be applied to the problem. His method uses a different induction algorithm than that of Caruana but compares results to the ones used in Caruana. On data sets containing discrete features the results are better, but results appear significantly worse on data sets containing continuous features. This suggests that the method would not be suitable for continuous input pattern recognition problems such as those investigated here.

Aha and Bankert [1] show yet another implementation of a case-based decision tree type classifier and a greedy selection algorithm. They argue that non-parametric case base classifiers are good classifiers to use during wrapper tests because they do not need hand tuning of any parameters during the search. They also support the argument of Kohavi in [29] that wrapper approaches are superior to filter approaches because the inherent biases of the classifier to be used are taken into consideration. Sommerfield [54] uses notions of strongly and weakly relevant features also introduced by Kohavi in [29] and applies them to a best first search strategy using decision trees. He then extends the search strategy using compound operators that increase the chance of finding the optimal subset.

Decision tree induction algorithms are not the only induction algorithms used for feature selection. Thawonmas [57] describes a method in which a fuzzy logic classification system is used as the classifier being optimised. The number of exception regions contained in the classifier is used as the optimisation function. According to [57] this can be directly related to the increase in classifier recognition rates. A backward selection technique is employed, and selection is stopped when a significant increase in error rates would occur, where *significance*

is a user input to the system. The approach is novel in that it is the only one found in the survey which uses fuzzy classification as a method of finding the best feature subset. The employed backward selection search method does not ensure optimal subsets.

A reference on finding more optimal greedy search methods for feature set selection can be found in Pudil et al. [45].

Stochastic Search Techniques

Quite a few references exist on using stochastic search algorithms in finding subsets better than the full set. These techniques do not promise to give optimal results, but are ideally suited to the problem due to the binary nature of the search space.

Vafaie and de Jong [61] explore the use of Genetic Search strategies in finding optimal feature subsets for texture recognition problems. They develop an intuitive hypothesis explaining the brittleness of Greedy Search Methods. Using a rule-based classifier called AQ15 and Genetic Search they show that results more optimal than Sequential Backward Selection can be obtained. A similar example can be found in Cherkauer and Shavlik [12].

In Smith, Fogarty and Johnson [53], the use of genetic algorithms is shown in building k-Nearest Neighbour Classifiers. It is argued that sub-optimal criterion functions of much less complexity can be used in the feature selection problem, as long as the biases are the same as those of the final classifier being used.

The use of neural network classifiers as the criterion function is restricted by the long time taken to train most of these classifiers. Stochastic search techniques usually require a few thousand evaluations of the criterion function, which restricts the usage of neural networks as the black box in the wrapper approach. Yang and Honavar [63] was the only paper found in this survey which directly addresses this problem. They propose a neural network approach using a network construction technique that limits the training time of the network and present some promising results.

Other work showing the application of Genetic Algorithms to feature selection is that of Gaborski et al. [23], Schlosser et al. [48] and Whitley et al. [62].

Other Approaches to the Problem

As was stated earlier, artificial neural networks are mostly too slow to use in the wrapper search approach to feature selection. A different approach can however be taken. Instead of searching through the input space, the input weights of the neural networks can be used to find the importance of certain features. Setiono and Liu [50] show how the input weights of a

neural network can be used to select features. Related methods can also be found in Messer, Kittler and Kraaijveld [40] and in Kohavi, Langley and Yun [33]. A problem with many of these approaches is that features with low input weights cannot always be considered to be irrelevant. It must also be remembered that many neural network approaches implicitly use a search strategy (such as back-propagation) in the weight space to find the best set of weights. This implies that the set of weights is not optimal in most senses, since convergence to the optimal set cannot be guaranteed. The larger the size of the input space given to the network, the larger the size of this search space, thus making it even more difficult to find a measure of importance of any feature from the weights of a neural network.

Cherkauer and Shavlik [11] propose another approach that uses the model built from the data to decide which features are important. Their method uses a measure of the number of leaves built by a decision tree algorithm. They argue that this method captures the "transparency" of an input embedding, where "transparency" refers to the average complexity of accurate models under a specific embedding.

Another novel approach to the problem is that of doing *Context-Sensitive* feature selection. Here an extra degree of freedom is added, in that a feature is not merely added or deleted from the best set, but that different features are used depending on the region of feature space where the training example is found. The earliest work found is that of Turney [59]. He first defines a definition of context-sensitivity and argues that many real world problems such as medical diagnosis contain context-sensitive features. He defines five different ways of handling context-sensitivity and explores three of these methods on some real-world problems. Several types of classifiers are investigated and compared.

A second reference on context sensitivity is that of Domingos [17]. Domingos argues that many "lazy-learners" are affected by the problem of context-sensitivity. "Lazy-learners" are used to describe methods that delay the decision process by storing the learning samples instead of building models from them. Domingos develops a learning algorithm using a clustering-like approach that takes into account context of features during the learning phase. Promising results are obtained, but not compared to other classification methods.

Perhaps an appropriate end to the list of wrapper approaches to feature subset selection would be the comprehensive paper of Kohavi and John [30]. They compare different methods under different circumstances, and show several interesting examples on feature relevance. It is shown that the optimal feature subset does not always have to contain only relevant features, and that all features that are relevant do not have to be in the optimal subset. The paper also contains a rather detailed literature survey that might include references not mentioned here.

3.3.2 Filter Approaches

For classifiers that require a substantial number of parameters to be estimated or designed, the wrapper approach to feature selection is not a viable option. For these types of classifiers, the usual practice is to find some other criterion function that (hopefully) correlates well with the obtainable classification rates. These feature selection techniques then either use the same types of search strategies as discussed for the wrapper approaches or, if possible, more optimal search strategies are implemented that do not have to search the whole space of possibilities. This is especially true if the criterion function adheres to the set inclusion monotonicity property.

A distinct advantage of these methods is that many of them are designed to be quite fast, and full searches of the whole feature space can therefore be made to find the optimal subset according to the criterion imposed.

Many of these methods date back to the late 70's and early 80's, but new methods have also been added to those through the years. The work presented in Kittler [28] is a good reference on the early methods. Kittler classifies the different methods into five different categories and discusses each of these. Most of the methods require in some way the estimation of the underlying probability density functions of the input distributions. Different distance measures are then used to decide how well a specific feature subset can distinguish between the different classes. Some of the more recent filter approaches will now be added to that of the study given by Kittler.

Information Measures and Probabilistic Distance Measures

The methods coming from the field of information theory and from probabilistic distance notions are closely related. Both these areas use some measure of the density functions of the inputs to find distances in probability space between different distributions, and then use these distances to give some measure of how easy it will be for a classifier to separate the classes.

The method on which the first filter approach, presented in the next chapter, is based is that of Battiti [6]. He uses the concept of *mutual information* to find the relationship between the output variable and the input features. The paper first defines how the whole notion of mutual information is applicable to the problem, and then goes on to discuss the practical problems of implementing mutual information measures in practice. These problems are mostly related to the problem of estimating densities in n-dimensional space. An approximate solution is then suggested which is similar to the concept of forward selection, but which also looks at the dependence of already chosen features to features being added next. The work of Battiti

was later extended by Bonnlander and Weigend [9] and finally presented by Bonnlander in his doctoral thesis [8]. He argued that the method used by Battiti for estimating the density functions was incorrect and proposes using non-parametric kernel density estimation techniques. He also makes the assumption that the number of training instances are "plentiful", and argues that the only method of finding the best subset when features are not independent is either a full search, or some heuristic search approach.

Lovell et al. [37] present a method called Expected Attainable Discrimination in which the Receiver Operating Characteristic Curves are used to find a measure of attainable discrimination for two-class problems. Their method is designed specifically for the area of nominal variables such as those obtained by questionnaires and is not suited to the problem of continuous random variables such as is found in the types of classification problems investigated here.

Novovicova, Pudil and Kittler propose a method in [43] that is based on the Kullback J-divergence measure. This divergence measure is similar to the notion of mutual information. They employ a method of estimating the density functions by finite mixtures of parameterised densities. The method is in a sense similar to neural network techniques for finding the optimal features by analysing the input weights of the classifier. In selecting the feature being used, the method also yields the desired decision boundaries for doing the classification of future examples.

In their work, Koller and Sahami [31] also investigate a method based on information theory. As a distance measure they use the Kullback-Leibler distance. They argue that information measures are better than divergence measures in high-dimensional spaces. They then present from the field of probabilistic reasoning a Markov-blanket criterion which, according to them, removes the need for a full search of all subsets. They argue that their approach has the ability to find, by backward selection only, the most necessary features for classification. Unfortunately the concept of the Markov-blanket is a difficult one to implement in practice and several approximations have to be made which reduces the effectiveness of the method. It is argued that their method is well-suited to very large input spaces (in the order of 1000 features) where, according to [31], most wrapper approaches become too computationally expensive.

Methods using Separability Criteria

In his work with determining optimal embeddings for neural networks used in control applications, Končar [32] uses a method called the *Gamma Test* which reportedly gives an estimate of the variance of the model when building a *continuous* model from a sample data set. The idea was first published by Stefánsson, Končar and Jones [55]. In their work, they do not directly

apply the problem to classification, which in the end builds a continuous model, but discretise the output. Application of this method to classification problems is analogous to methods that uses within-class and between-class distance/separability measures. The method utilises the average distance between nearest neighbours both in input space and in output space, and uses a linear model derived from this as a measure of goodness for any particular embedding. The method is similar to earlier interclass distance methods proposed in [28], but differs in the respect that it does not directly use within-class and between-class measures.

In the technical report by Scherf and Brauer [47] a method similar to that of Stefánsson et al. is used. This method, called EUBAFES, is based on a similar distance metric of inter-class and intra-class distances. They argue that the normal inter- and intra-class methods fail to find relevant features in problems of the XOR-type, and propose using a nearest neighbour approach to solve this problem, which is very similar to the approach in [55]. However the connection between model variance and the output of the Gamma Test is not found explicitly in the work by Scherf and Brauer. In the work presented in Chapter 5, it is argued that if this relationship does indeed exist, it would have profound effects, not only on feature subset selection, but as a method for finding stopping criteria for artificial neural network training.

3.4 Discussion

The advent of high-speed computing has brought to the feature subset-selection problem the advantage (or disadvantage) of a multitude of methods. The number of combinations between evaluation criteria and search functions are endless. This makes it very difficult for anyone working in the field to decide which methods to focus on.

The field of pattern recognition on noisy continuous variables does however bring with it criteria that eliminate some of the methods. Methods based on discrete, non-noisy data can not be used directly, and this disqualifies much of the work done in recent years by the Machine Learning community.

The advantage of having a measure of the expected worth of features is one that is both intuitively appealing and highly necessary. In many pattern recognition problems features are designed by hand, and methods of knowing when the information provided by them is enough to handle the problem would be of great assistance. This was one of the criteria used upon deciding which types of feature selection methods to investigate. The fact that the concept of Mutual Information satisfies this criterion makes it an obvious choice for finding not only optimal subsets, but also an idea of the relative worth of different features. The concept of mutual information feature selection is therefore investigated in Chapter 4.

Another method from the literature that seems to satisfy the above criteria is the Gamma

Test method employed by Končar in his work on feature selection for neuro-control. If proven to be correct, his method does not only allow a measure of the usefulness of any specific feature, but would give an estimate of the expected error incurred in building a continuous model classifier when using a specific input embedding. This would be directly applicable to most types of classifiers used in the literature. For these reasons, Chapter 5 investigates the application of the Gamma Test procedure to the problem of feature subset selection for classification.

The fact that most of the approaches seen do not explicitly allow the search of input spaces for artificial neural networks is another area that calls for investigation. The scarceness of literature on these methods is due partly to computational costs of training these neural networks, and partly to some other considerations such as the availability of applicable stopping criteria and network size. In Chapter 6 it is shown that there does indeed exist a form of the neural network which is applicable and has training speeds high enough to do full searches of a number of features.

Comparisons between these different methods are made, and the suitability of every method to the problem of feature selection is investigated. Similar to most other literature, the methods are first shown to work on some hand created (and very easy) data sets, and are then applied to real data sets from standard Machine Learning problems.

Chapter 4

Mutual Information

In this chapter it will be shown how the information-theoretic concept of mutual information can be applied to the feature subset selection problem. The work presented here is based in part on work by Battiti [6] that was later extended and used by Bonnlander [9].

The mutual information method was chosen for investigation since it was one of the few methods found in recent feature selection literature on filter approaches that seem to have sound theoretic background. The method also has an intuitive appeal for the feature selection problem. In the early literature [28], similar information-theoretic methods were applied to parametric classifiers for the multivariate normal case.

Section 4.1 describes the theoretical concepts of mutual information. Thereafter some considerations when implementing these concepts in the practical case are highlighted. This is followed by a description of the mutual information feature rating algorithm. Before concluding, a simple example is given to illustrate the concepts discussed in the chapter.

4.1 Applicability of Mutual Information to Feature Selection

The concept of mutual information (MI) was developed in the field of information theory (although some very similar probabilistic distance measures were also developed in the field of statistical estimation theory). The derivation of the equations that describe how to calculate MI is developed from the concept of *entropy*. It will be derived here using similar notation to Cover and Thomas [13], but with the same emphasis on classification concepts as in Battiti [6].

Starting with a discrete random variable X over an alphabet $\mathcal{X}$, the entropy $H(X)$ of this variable can be defined as:

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x) \tag{4.1}$$

with $p(x)$ being the probability mass function defined on $x \in \mathcal{X}$. This measure represents the average uncertainty of the random variable X.

Also of interest here is the definition of conditional entropy. If another discrete random variable Y is used as a measurement to reduce the entropy of the random variable X, the definition of the conditional entropy $H(X|Y)$, which represents the uncertainty in X given Y, is defined as

$$\begin{aligned} H(X|Y) &= -\sum_{y \in \mathcal{Y}} p(y) H(X|Y=y) && (4.2) \\ &= -\sum_{y \in \mathcal{Y}} p(y) \sum_{x \in \mathcal{X}} p(x|y) \log p(x|y) && (4.3) \\ &= -\sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} p(x,y) \log p(x|y) && (4.4) \end{aligned}$$

Adapting this to the process of classification, the random variable X can be viewed as the class label and each event as being a different class label. If we denote this new random variable with C and write (4.1) for the two-class problem it becomes

$$H(C) = -\sum_{c=1}^{2} p(c) \log p(c) \tag{4.5}$$

where the sum is now taken over the events of the two class labels[1].

$H(C)$ is now a measure of the uncertainty of the class label. By extracting features from the process, an attempt is made to decrease the uncertainty about the class label. Let $\mathbf{f}$ represent the vector of features that is extracted from the process, and N_f denote the number of samplings made. We then have (from Equation 4.3)

$$H(C|F) = -\sum_{f=1}^{N_f} p(\mathbf{f}) \left(\sum_{c=1}^{2} p(c|\mathbf{f}) \log p(c|\mathbf{f}) \right) \tag{4.6}$$

which represents the average uncertainty about the class label given the measurements $\mathbf{f}$. (Here, $p(c|\mathbf{f})$ represents the conditional probability for class c given the input vector $\mathbf{f}$.)

The reduction of the uncertainty of the class label C that is obtained by making the mea-

[1] Although only the two-class problem is discussed, the derivation would clearly be extendable to the multi-class problem.

surements $\mathbf{f}$ is given by

$$I(C;F) = H(C) - H(C|F) \tag{4.7}$$

This reduction in uncertainty is also the definition of the mutual information $I(C;F)$ between the class label c and the measurement vector $\mathbf{f}$. The equation for the mutual information between the class label and the input feature vector can now be written as

$$\begin{aligned}
I(C;F) &= H(C) - H(C|F) && (4.8)\\
&= -\sum_{c=1}^{2} p(c)\ \log\ p(c) - \left(-\sum_{f=1}^{N_f} p(\mathbf{f}) \left(\sum_{c=1}^{2} p(c|\mathbf{f})\ \log\ p(c|\mathbf{f}) \right) \right) && (4.9)\\
&= \sum_{c=1}^{2} \sum_{f=1}^{N_f} p(c,\mathbf{f})\ \log\ p(c) + \left(\sum_{f=1}^{N_f} p(\mathbf{f}) \left(\sum_{c=1}^{2} p(c|\mathbf{f})\ \log\ p(c|\mathbf{f}) \right) \right) && (4.10)\\
&= \sum_{c,f} p(c,\mathbf{f}) \log \frac{p(c|\mathbf{f})}{p(c)} && (4.11)\\
&= \sum_{c,f} p(c,\mathbf{f})\ \log \frac{p(c,\mathbf{f})}{p(c)\,p(\mathbf{f})} && (4.12)
\end{aligned}$$

If the input feature becomes a continuous random variable, the sum on the feature vector $\mathbf{f}$ becomes an integral.

$$I(C;F) = \sum_{c} \int p(c,\mathbf{f})\ \log \frac{p(c,\mathbf{f})}{p(c)\,p(\mathbf{f})} d\mathbf{f}. \tag{4.13}$$

Some interesting properties of mutual information can also be derived (see [13] for complete derivations). One property of interest here is that of symmetry; after some manipulation [13] it can be shown that for two random variables X and Y, the mutual information measurement is symmetric, i.e.

$$I(X;Y) = I(Y;X) = \sum_{x,y} p(x,y)\ \log \frac{p(x,y)}{p(x)\,p(y)}. \tag{4.14}$$

A qualitative explanation of mutual information is that it can be thought of as a measure of the "lumpiness" of the joint probability density function of c and $\mathbf{f}$. From Equation 4.13 it can be seen that the contribution to the integral would be large if the distribution $p(c,\mathbf{f})$ is uneven. In this case, the product of the marginals is not equal to the joint density function, and therefore the class is not independent of the measurements made. The more dependent the measurement and the class becomes, the higher the mutual information measure. This makes mutual information a good measure to determine which features of the input feature set have the greatest effect on output entropy. If a vector has little relevance with the output, the mutual information between the two will be very low. On the other hand, if they are

mutually informative then the measure will be high.

4.2 From Theory to Practice

Although the concept of mutual information looks like a novel idea in theory, some problems arise in obtaining the estimates of $I(C;F)$ in practice. These problems mostly arise because the underlying density functions of the input vectors (the *a priori* and *conditional* information from the Bayesian classifier) are not known, and only a finite amount of data is available from which to estimate these density functions. This prevents us from making estimates of mutual information in high-dimensional spaces, since the number of data points needed grows very rapidly with the dimensionality of the space[2].

To overcome these problems, Battiti [6] suggests an approximation for the feature selection case. Instead of calculating MI from the joint density functions in the N_f-dimensional space, he reduces the dimensionality by calculating it between only two vectors at a time. Firstly, MI calculations are made between each input feature and the desired output vector i.e. $I(C, f_i)$. This gives a indication of the relationships of each feature vector with the output. Secondly, the MI is calculated between different features i.e. $I(f_j, f_k)_{j \neq k}$. This second measure is used in an attempt to remove highly mutually dependent features from the feature set. It is argued, and also well-known, that the mutual information between features and output alone is not able to take into account dependencies between different features. It could happen, for example, that two highly correlated features have high mutual information with the output class. Including both these features at the cost of some other feature which is uncorrelated with the first two (but which has only slightly less mutual information with the output) would result in a less optimal feature subset. In such a case, it would probably be better to use only one of the two correlated features, and then use the third independent feature as well.

In Bonnlander [9] it is argued that this method will still be suboptimal, and that the joint density calculations have to be made in the correct dimensionality to take into account dependencies between different features. However, it is agreed here, and has been shown in many cases [49, 51], that the number of required data points for estimating the density functions is exceedingly high when the number of feature becomes large. Bonnlander aims his method at problems were sample data size is very large (more than 10000 points). The data sets tested in the work presented here do not fall into this class of problems, since most of these problems have a sample size of less than than a thousand points. For this reason it was decided to follow Battiti's method, but with some changes in the density estimation technique.

Battiti uses a method introduced by Fraser [20] for obtaining the necessary density estimates.

[2]Refer to Section 2.2 for more details. A standard reference on density estimation will also give deeper insight into this problem e.g. [51, 49].

This method is a form of equal mass histogram binning which, according to Battiti, can greatly reduce the time needed to make the density estimates. For details of this method, the reader is referred to the original paper by Fraser [20] and the appendix in the paper by Battiti [6].

Bonnlander points out a simple example case where Fraser's density estimation technique does not give correct results. Initial experimentation in the work here also supported this view. Fraser's method is based on a partitioning of the space into equi-probable areas (thus the so called *equal mass binning*). The fact that Fraser's method breaks the regions into rectangular areas of equal probability, using a χ^2 test as criterion for further splitting, has some serious disadvantages. This can easily be seen when applying Fraser's method to normally distributed data, where the form of the binnings would now be rectangular rather than curved in shape. Bimodal data with a main mode and some less probable second mode is another example that would not be handled well by Fraser's method. It was decided to opt rather for a density estimation technique that has more support in the density estimation literature.

In most of the literature, density estimation is done either by some form of histogram binning, or through the use of some kind of kernel density estimation technique. Both of these methods have their advantages and disadvantages. In histogram binning, the number of bins used in the estimation is of critical importance. Having too many bins will cause some of them to be void of data points. If the number of bins is too few, the estimated function would not show local changes in density very well.

For the case of kernel density estimation, a similar trade-off is used. Here, the density estimation is done by placing a kernel function, which decreases monotonically from its centre points, at every data point in the space. The width of the kernel then determines the approximation of the density function. If the kernels are made too wide the local structure is lost, and if the kernels are very narrow the density function will have numerous unwanted peaks.

In Bonnlander's initial work [9] he uses a method similar to that of Fraser and argues that it is more optimal than kernel density estimation. However, in the final work on his doctoral thesis two years later, this method is not discussed at all, and he argues that a kernel density estimation technique would suffice for performing the density estimates. The conclusion drawn from his sudden change in perspective can be that some problems in using equal mass binning must have convinced him to use kernel density estimation techniques instead.

Kernel density estimation has some advantages over histogram binning. If the kernel used has infinite support, then even open spaces that contain no data will in the end have some (very small) probability of occurrence. This is more desirable than having areas with zero probability which is commonly found in histogram binning (unless, of course, it is believed that the true nature of the density function is such that it only has compact support). The

fact that most of the features considered in this work are of continuous nature, supports a choice of kernel density estimation. In theory it is possible to show that both histogram binning and kernel density estimation tends to the correct density as the number of samples increase [18].

With all of these factors taken into consideration, it was decided to use kernel density estimation techniques, since the number of samples considered in our case would be ample but not abundant, and the estimates would only be made in two-dimensional space.

The estimation was done using kernel density estimation software for Matlab obtained from Beardah [7]. This software was created using theory partly from the book by Silverman [51], and uses a method reported in Silverman for adaptively determining the width of the kernel. The method is based on multivariate normal data, but seemed to give rather good results on the data used here.

A point that must be mentioned is that the *precise* mutual information measurement is not of critical importance in this work, as long as the *relative* information content stays ordered in the same way. This concept was addressed by Battiti, and forms another supporting reason that precise kernel width is not important when comparing mutual information measurements. The fact that the measurements are not to be used as stand alone, precise, measurements but rather in an comparative study, alleviates the need for accurate determination of the kernel width.

4.3 The Mutual Information Ranking Algorithm

The algorithm used to rank the features in order of importance using the mutual information technique are discussed below. This algorithm differs from the work by Battiti in the fact that no pre-specified value for the number of features have to be chosen. Rather, if it is possible, the evaluation of the different subsets created as features were added, are encouraged.

Let C represent the desired output, f_k the kth input feature, n the number of features and $\mathcal{F}_r$ an empty feature set, which in the end will contain the ranked set of selected features. To find the ranking of the features using the mutual information method, the following algorithm was used:

1. Append to $\mathcal{F}_r$ that f_k which maximises $I(C, f_k)$ for $k = 1..n$
2. Find $I(f_j, f_k)_{j \neq k}$ for all features not yet contained in $\mathcal{F}_r$
3. Append to $\mathcal{F}_r$ the f_j that maximises $I(C, f_j) - \alpha \sum_{f_s \in \mathcal{F}_r} I(f_j, f_s)$
4. if $\mathcal{F}_r$ does not yet contain all features jump to 2.

This algorithm ranks features in their order of importance. Features with high information content about the output obtain better rankings, but are penalised if they show too much common information with already selected features. The constant α yields the importance of not selecting features which have high mutual information with already selected features. Battiti suggests that in practice a value for α somewhere between 0.5 and 1 must be used.

This algorithm falls into the class of forward selection techniques. Battiti [6] demonstrates that using either forward or backward selection does not yield vastly different results on final classification rates. It was decided in this work to first test the forward selection method against other selection techniques, and if the method hinted at superior performance compared to other methods, to investigate this matter further.

4.4 An Illustrative Three Feature Example

As a first example of how mutual information can be utilised as a feature selection criterion, it was decided to test the mutual information method on a simple tri-variate two class problem. A simple data set was created that contained two features drawn from normal distributions with differing means for the different classes, and a third feature which was distributed uniformly over all classes.

A plot of the data set is shown in Figure 4.1. From the plot it can clearly be seen that the inclusion of feature 3 into the set could have bad effects on the possible classification accuracy. The final problem however, still is almost linearly separable.

Figure 4.2 shows the mutual information measurements obtained during this test. As expected, results for the experiment show that features 1 and 2 have high information content with the output while feature 3 has a very low MI value. It can further be seen that features 1 and 2 have a high information content with each other. The final ranking obtained for these features was in the order: feature 1 first, feature 2 second and feature 3 third.

4.5 Other Experiments and Conclusion

The experiments to validate whether mutual information can be used for doing feature selection, as well as seeing how the mutual information method compared to other techniques are shown in Chapter 7.

The next chapter discusses another filter approach found in the literature, which can be applied to the problem of feature selection. This method seemed rather promising and also has some relevant theoretical background.

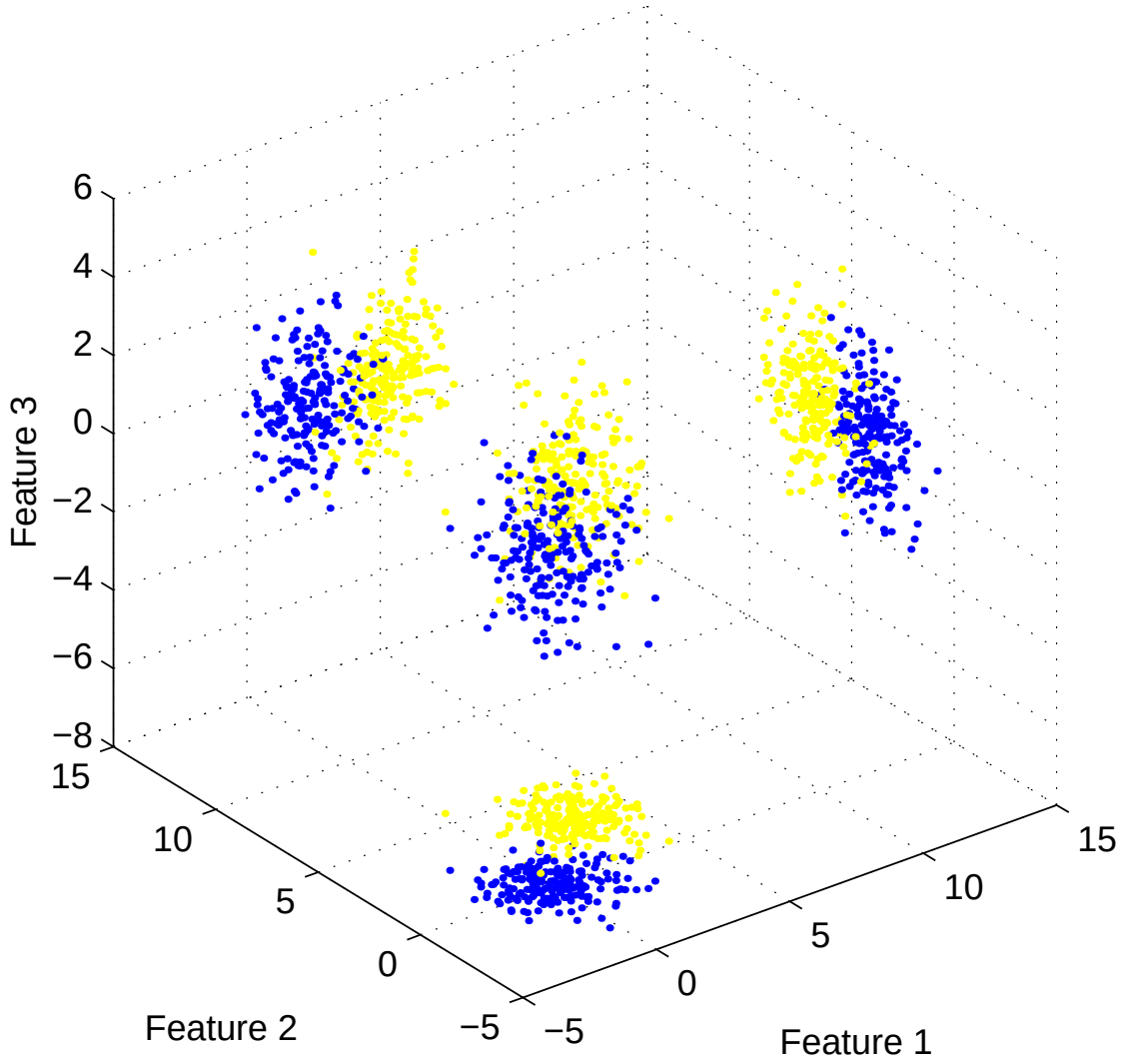


Figure 4.1: The three feature example data. Projections onto different planes are also shown. Features 1 and 2 are normally distributed and have differing means for the two classes. Feature 3 is drawn from a uniform distribution.

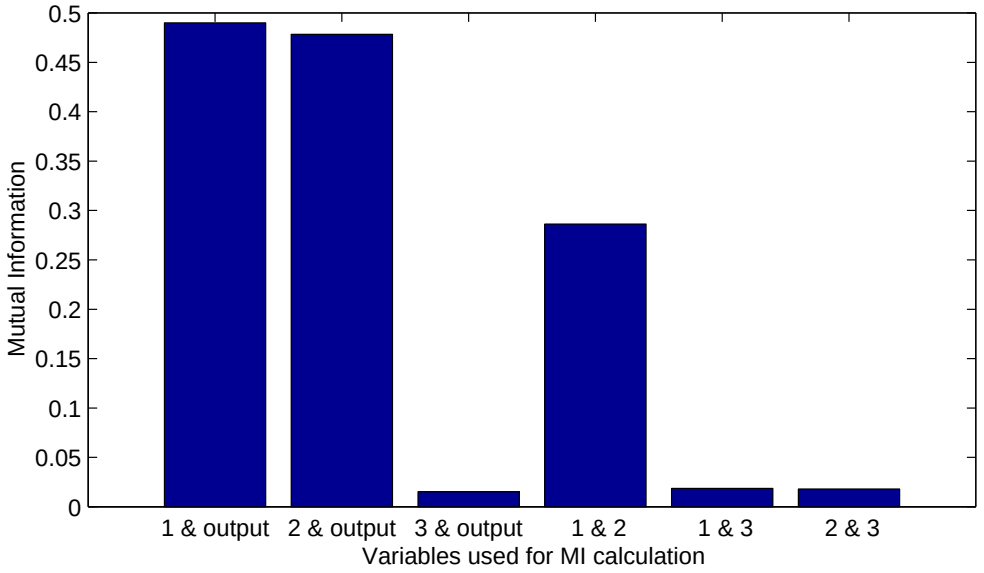


Figure 4.2: The measurements made by the mutual information ranking algorithm

Chapter 5

The Gamma Test

A reasonable assumption when working with most classification problems, is to assume that when two points are close in the feature space, they probably should have the same label. This is the basis on which simple nearest neighbour techniques base their classification[1].

In this chapter a feature selection method called the *Gamma Test* based on this intuitive idea is discussed. The method was not originally developed for use in feature selection for classification purposes, but it is shown here to be very similar to other selection techniques used previously in literature (with some distinct advantages as well).

In the first section of the chapter, the theoretical concepts and origins of the Gamma Test will be described. Next, the applicability of this method to pattern recognition is discussed and it is shown that the method indeed can be used for feature selection in this case. An explanation of the inner workings of the test follows, which should give the reader good insight into why the method works the way it does. Section 1.4. discusses several illustrative examples, and the chapter concludes with some ideas on the similarity of, and differences between, this method and other methods of its class.

5.1 Theoretical Concepts

The Gamma Test [32] is a simple statistical test to find a data-derived estimate of the variance of the error in fitting a smooth model to any continuous mapping. It was developed by Končar as part of his doctoral thesis on optimisation techniques for neuro-control systems.

Suppose $(\mathbf{x}, y)$ is a data sample with $\mathbf{x}$ denoting the input data and y representing the output data. Suppose, furthermore, that the data was generated by a suitably smooth process with

[1]Nearest neighbour techniques can be shown to converge to no more than twice the optimal Bayes classification rate as the number of data points are increased [18].

bounded first and second derivatives. This input-output mapping can be defined as $y = \mathcal{F}(\mathbf{x})$, and can be written as the expression

$$y = f(\mathbf{x}) + r \tag{5.1}$$

with r denoting the indeterminable/noisy part. The noise may be pure noise, or may be due to a lack of representation ability of the functional mapping.

The Gamma Test was designed to derive a data based estimate of the variance of r, and thus an estimate of the ability to build a good model using any particular set of input vectors. The assumptions under which this estimate holds is discussed in [32].

From the data sample $\mathbf{x}$, we now find the sample $(\mathbf{x}', y')$, which has the property that it minimises $|\mathbf{x}' - \mathbf{x}|$ (this is the sample of nearest neighbours of $\mathbf{x}$). The Gamma Test can then be defined as

$$\gamma = \frac{1}{2M} \sum_{i=1}^{M} (y'(i) - y(i))^2, \tag{5.2}$$

where M is the number of data points in the sample.

According to [32] it can be shown that this value tends to the variance of the error r as the nearest neighbour distances tends to zero. In order to make these distances tend to zero, the ideal solution would be to make the data size M tend to infinity. Since we cannot always increase data size in order to investigate the Gamma value as the nearest neighbour distances decrease, an approximation must be used in order to find the estimate on practical data sets. Thus, due to the availability of only a finite number of samples the effect of letting M tend to infinity in the above equation is approximated using the following technique.

We have $(\mathbf{x}(i), y(i))$ with i ranging from $1..M$ where M is the number of samples, and $\mathbf{x}(i) = (x_1(i), ..., x_f(i))$ with f being the number of input features. We now let $\mathbf{x}(N(i,p))$ represent the pth nearest neighbour to $\mathbf{x}(i)$ and define

$$\Delta(p) = \frac{1}{p} \sum_{j=1}^{p} \frac{1}{M} \sum_{i=1}^{M} |\mathbf{x}(N(i,j)) - \mathbf{x}(i)|^2 \tag{5.3}$$

and

$$\Gamma(p) = \frac{1}{p} \sum_{j=1}^{p} \frac{1}{2M} \sum_{i=1}^{M} (y(N(i,j)) - y(i))^2. \tag{5.4}$$

To understand Equation 5.3, evaluate the term inside the sum over i first. This forms the square of the distance between $\mathbf{x}(i)$ and its jth nearest neighbour. The normalised sum over i forms the mean of this squared distance for all points in the data sample. $\Delta(p)$ then represents the average nearest neighbour distance up to the pth nearest neighbour for all sample points.

From Equation 5.4, $\Gamma(p)$ is an estimate of the statistic in (5.2) based on these $1..p$ nearest neighbours.

The next step, is to build this estimate for different values of p. Thus, let p be the range of integer values from $1..P_{max}$ where P_{max} has to be chosen. By forming the coordinates $(\Delta(p), \Gamma(p))$ and plotting these estimates for increasing values of p, a curve of the values of $\Delta(p)$ and $\Gamma(p)$ is obtained. It is argued [32] that if enough data samples are present, this curve would be of linear nature. Performing a least squares linear fit to this curve we find $\Gamma = \alpha\Delta + \gamma$ with γ representing the estimate of the value of (5.2) as the nearest neighbour distances approach zero. An example of the technique is given later in the chapter.

5.2 Applicability to Pattern Recognition

The question arising out of this analysis would now be why the variance of the error is at all useful. It is shown and discussed in [32] that the variance of the error can be used to form a bound on the error rates of a neural network modelling system. It is further shown that the neural network, used in modelling a continuous mapping from input to output, is a bounded first and second order system, and that the Gamma Test theory therefore applies to these types of networks. Furthermore it is known, and has been discussed in Section 2.2, that variance in estimation techniques has specific relationships with the expected error bounds of these techniques. Any method that has the ability to quantify the variance of the error would therefore be able to give bounds on the expected attainable performance of the system.

A problem in applying this method to classification and pattern recognition arises from the fact that these recognition systems normally have an output stage which does not have bounded first and second order partial derivatives, since a threshold function is used to discretise the output to one of many fixed class values. Končar discusses this issue briefly and states that even though the final output might not be bounded in its derivatives, the input mapping produced by the neural network or other classification system is mostly a continuous model, and thus the bound found in this way can still be applied to the problem of pattern recognition. This fact, though dubious and certainly not proven here at all, is investigated by some experimentation. The fact that other techniques found in the feature selection literature use methods similar to this, coupled with some of the results obtained in our experiments, definitely shows that this method deserves some more theoretical work.

The next question to ask then, is how to use this test in order to perform feature selection, and what other properties of this method makes it attractive for feature selection systems.

The Gamma test method can be implemented in $O(M \log M)$ time, and thus is much faster than most wrapper approaches to the feature selection problem. Should this method be

found to have high correlation with classification accuracy it would be very useful as a filter technique for feature selection. The speed of the test enables it to exhaustively scan through all different input embedding combinations for up to 20 features. The best subsets found in this way can then be tested to find the optimal subset for use with a particular classification system. Another attractive property is that the method does not require density estimation techniques which can be very time consuming and also very prone to error (especially in high-dimensional spaces).

5.3 The Inner Workings of the Gamma Test

To employ this method in a pattern recognition case, it must first be decided how the distance in the output space is to be handled. For the two-class problem the easiest solution is to just assign some integer value to the one class and another to the second class. The distance in output space is then simply calculated as the difference between the labels. This was the approached followed in the work here, where only two-class problems ere dealt with.

In the case of multiple labels, a different measure will have to be used, since a difference between more than two output labels cannot be expected to correspond in any way to distance in input space. An acceptable technique would be to use a difference of 0 for similar class labels, and a 1 otherwise (also called the discrete metric).

Taking a closer look at the equations of the Gamma Test yields some insights into the ideas behind the test, and why it can be expected to work in pattern recognition problems. Shown in Figure 5.1 is a plot created using the Gamma Test algorithm[2]. The figure shows both the nearest neighbour average distances in input and output space, as well as a linear approximation to this. The data used was the WDBC data set from the UCI data repository [60].

The x-axis represents values calculated using Equation 5.3 and the y-axis the Gamma values from Equation 5.4. Investigating Equation 5.3, it can be seen that it represents the average distance between nearest neighbours of a certain degree[3]. This distance is monotonic on the degree. As the degree of the neighbour used for the distance calculation is increased, so too must the distance between points in the input space. (It might not be be *strictly* monotonic, since neighbours of differing degrees can have the same distance, but it is highly unlikely that a data set would have nearest neighbours which on average did not increase as the degree of neighbours increased.)

[2] An optimised implementation of the Gamma Test was written by Steve Margetts [39] as part of the work by [32]. The code is implemented in C and has sufficient flexibility for almost direct application to any problem.

[3] The closest point is a neighbour of the $1st$ degree. The second closest point is a neighbour of the $2nd$ degree and so forth.

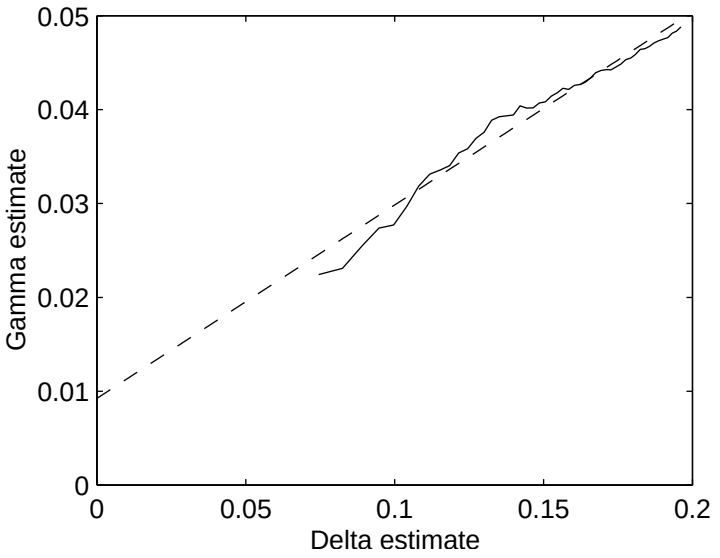


Figure 5.1: The Gamma Test regression estimate for one of the data sets

If the premise of closeness in input space yielding likeness in output label is applicable, it can be expected that most of the output values between neighbours will have the same label. However, at the borders of the classes, there will be some points which very quickly start adding to the output difference measure of Equation 5.4. If the data is not multi-modal then this distance too would be expected to increase monotonically as the nearest neighbour degree is increased.

In the Gamma estimation technique, the input distance and output distance are now plotted against each other. The monotonicity of both of these variables imply that this plot will be a monotonic function under the assumptions made above. This can be seen in Figure 5.1. The plot also shows the linear approximation made on this data. This approximation is made by fitting a line to the graph utilising the criterion of minimum mean square error[4].

It can be seen from the graph that the linear model for the approximation of the cut-off seems relevant. Other experiments were also performed to investigate this approximation. These experiments are detailed later in the chapter.

To select the best subsets using the Gamma Test now becomes a simple matter of finding that subset for which the Gamma Test yields the lowest value. As a safeguard against estimation error, the alternative of finding a number of best subsets and then using some other criteria to decide on the best of these can be used. The test is normally fast enough to handle a exhaustive search of up to 20 features. If more features than this have to be tested, some other sub-optimal search routine must be used. The speed of this method far exceeds most

[4]As discussed in Section 5.1.

of the wrapper approaches using neural network classification error as criterion function.

5.4 Initial Experiments

5.4.1 An illustrative Example

As an initial and informative experiment on the Gamma Test, it was decided to investigate the effects of difference in class mean on a simple bivariate normal two-class example. The first class was centred on the origin of a two-dimensional feature space. The second class had its mean shifted from the origin by some distance, and the effects on Gamma value were investigated as this difference in mean was changed.

Figure 5.2 shows the data set in one of its positions. The estimated Gamma values as the distance changes are shown in Table 5.1

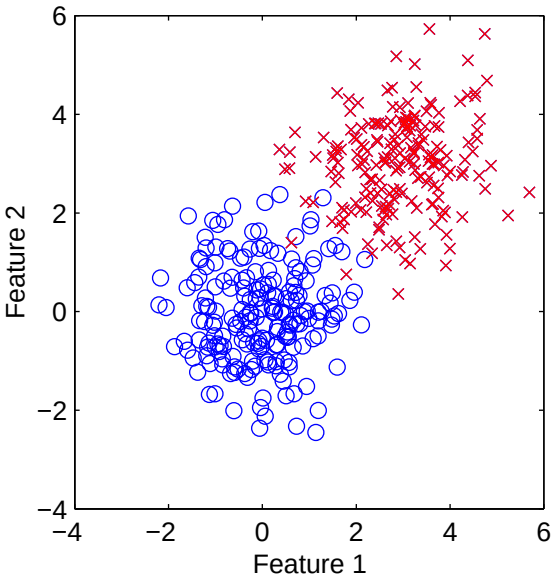


Figure 5.2: The illustrative data set plotted with the mean between the two classes shifted by a distance of 3 units in the direction of feature 1 and 3 units in the direction of feature 2

As expected, the Gamma estimate becomes smaller and smaller as the two classes become more and more separable. This indicates that the Gamma measure may, in some way, be related to the expected error rate of the classifier. The negative result for the last value in the table might be unexpected, but in fact is a result of estimation error in performing the straight line fit.

This does, however, raise a question about how to handle negative values. Another factor

Table 5.1: The Gamma estimates as the distance between the two classes are increased.

Difference in Mean	Gamma value
2.82	165×10^{-3}
4.24	12.7×10^{-3}
4.95	4.90×10^{-3}
5.65	1.29×10^{-3}
6.36	0.116×10^{-3}
7.07	-0.040×10^{-3}

that might sometimes lead to negative estimates is when the gradient of the line becomes very high, and the average output distance is very low. The work by Koncar [32] does not indicate with certainty whether bigger negative values can be treated as indicating an even more separable data set, or whether these values should just be taken to indicate zero variance and thus perfect separability. In the comparative experiments of Chapter 7, the estimate of the Gamma value was used as part of a criterion function for rating different feature sets, and more negative values were used to indicate better subsets. However, the experiments were performed over multiple tests, so as to minimise the effects of estimation error in the estimates, and, as is explained in Chapter 7, a number of the best subsets were evaluated to finally find the optimal subsets.

5.4.2 The Three Feature Example

The same three feature example that was used in testing the mutual information method was also applied to the Gamma Test. Results of the test on the different subsets are shown in Figure 5.3. From the figure the method clearly indicates that inclusion of the third feature has bad effects when only one of the other two features is used. This is shown by the high cut-offs of the linear approximation estimates. However, it can be seen that the method does not indicate that the inclusion of feature 3 would have a bad effect if both features 1 and 2 are used. The estimated cut-off point is only slightly less when using features 1 and 2 then when using all three features. Although this might be seen as a negative effect in the sense of feature selection, it need not be. The fact is that this dataset is still almost linearly separable using all three features and the noise in the third feature does not yet increase the error rate of the classifier too much. This fact however supports the idea that the Gamma Test by itself might not be sufficient to discriminate between different subsets that show similar ability to separate the classes. The method must therefore be supported by a second stage of comparing a few of the best subsets found (as in Chapter 7).

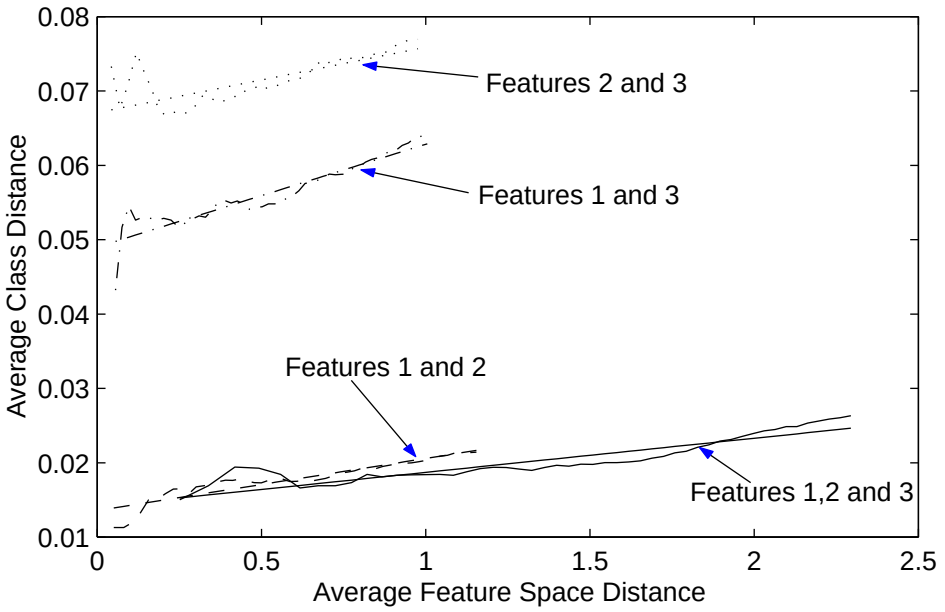


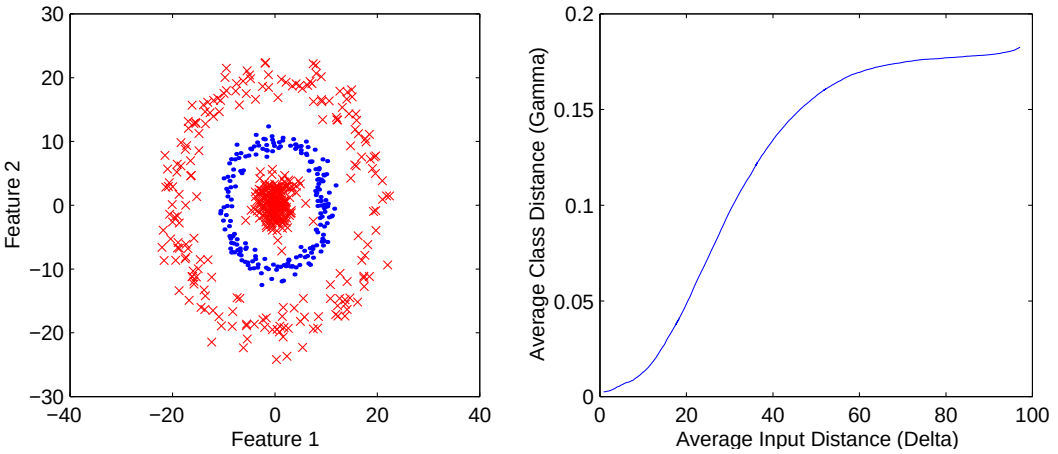
Figure 5.3: The Gamma Test regression estimate for the three feature example

5.4.3 Effects of Unexpected Structure

One of the areas in which it can be expected that the Gamma Test would not perform well is when there exist multiple modes or strange structure in the data, and these modes are separated by another class. In this case the monotonicity of the distances would inevitably not hold, and the average output distance would first increase and then start decreasing as the number of nearest neighbours is increased. The output distance would first include many samples of the same class. This would gradually increase as samples of the other class start to appear in the distance measure. However, when the number of neighbours used becomes too many, the output labels would start corresponding again and thus the output distance would start decreasing. This invalidates the monotonicity assumption discussed earlier, and thus cripples the linear approximation used in the Gamma Test technique.

To illustrate this, a two-class problem was created in which one class is surrounded by two modes of a different class. This specific dataset might never occur in practice, but serves well to illustrate the problem. The data set is shown in Figure 5.4(a).

A plot of the Gamma estimate is shown in Figure 5.4(b). From the plot it can be seen how the problem manifests itself. The strange structure of the data causes the linear approximation to be invalid. This can clearly be seen by the s-shape of the curve. The problem discussed here, however, would be the exception to the rule, especially if the number of nearest neighbours



(a) Sample data set for which the Gamma Test estimate might give unexpected results. Crosses indicates class 1. Dots indicates class 2.

(b) Sample data set for which the Gamma Test estimate might give unexpected results.

Figure 5.4: Problems with unexpected structure in a data set

used to find the estimates are kept at a minimum[5]. This example thus serves as a warning that the test should not be blindly applied to any data set and used without any investigation into possible multi-modality in the data set.

In this sense, however, the method can actually serve a useful purpose, since any strange structure in the plot of the Gamma distances could serve as an indication of unexpected structure or multi-modality in the data, which might then be used to build better estimates of the underlying density functions, or might be used as knowledge in the classifier. If such structure is found in the data set the method can also be adapted to it by changing the model fitted to the data, and using some other function estimation technique instead of a simple linear model. In the practical data sets investigated in the work here, this problem was not really encountered, and the regression estimates seemed valid in these cases.

5.5 Similarity to Other Methods

The feature selection literature has shown many approaches in which some heuristic measures of class separability have been used to find the "optimal" feature sets [28]. Many of these methods cannot be backed at all by any theoretical background. In this sense the Gamma Test has some advantage, and it might be worth thoroughly proving these theoretical grounds.

[5]The linear approximation would be valid for points with input distance between 0 and 10.

Some of the comparable methods use measures of inter-class and intra-class distances in the input space. These methods, however, work strictly on a per-class basis. The Gamma Test, on the other hand, allows the use of both input and output distance simultaneously. One method which notably compares with the Gamma Test method is that of Scherf and Brauer [47]. A problem with the method they present is that it is based on heuristic ideas of class separation.

The two approaches discussed so far have both been *filter* approaches to the problem. Next a wrapper approach for neural network classification techniques is discussed. This technique is novel in the specific application of this type of neural network, and has not, to the author's knowledge, been used for feature selection before.

Chapter 6

The Random Artificial Neural Network

The two methods presented in the previous chapters do not take into consideration at all the type of classifier that would be used for the final classification of the data using the selected feature sets. It is therefore not known whether the biases that these methods impose coincide with the biases that a specific classifier imposes. For this reason it was decided to also investigate a wrapper (classifier based) approach to the problem.

It was mentioned in Chapters 2 and 3 that many classifiers are either too complicated, take too long to train, or require too much user interaction for use in a *wrapper* approach to feature selection. Parametric techniques sometimes require a lot of user interaction, especially if the densities are of non-standard form, and might not have enough adaptivity to be used as a criterion function in the feature search. Techniques that *do have* high adaptivity to the problem situation without the necessity of user interaction at every step, such as neural networks, suffer from other disadvantages. For example, most neural network classifiers require lengthy training processes in order to set the weights of the network. Another problem found in these networks are that they suffer from over-training, and thus have to employ some form of early stopping criterion. These stopping criteria can sometimes be problematic.

This chapter focuses on one method that addresses some of the problems of feature selection for neural network systems. Many other efforts have been made to reduce the input dimensionality of neural network classifiers. These method do so by pruning the inputs that have low weights (see for example [50]). This becomes in effect a type of greedy search strategy which, as pointed out in Chapter 2, has some drawbacks. The removal of a single feature prevents any subset containing that feature from being evaluated later on, thus implying sub-optimal performance. If this fact is not taken into account in the pruning algorithm, the

performance of the feature selection algorithm can be erratic.

As in the previous two chapters, the theory behind the RANN is presented first. This is followed by a discussion of some practical implementation issues. The use of the RANN in feature selection is discussed in Section 6.3. To be consistent with the previous two chapters, the method is also tested on the simple 3-feature example. As conclusion to the chapter, some remarks about the validity of using classifier training error as a measure for feature selection are made.

6.1 The Theory

There exists a form of the neural network capable of rapid training and evaluation without much user interaction and, without the need to specify early stopping parameters. This network was presented by Anderson [3]. The structure of the network, which is the same as most fully-connected feed-forward neural networks, is shown in Figure 6.1

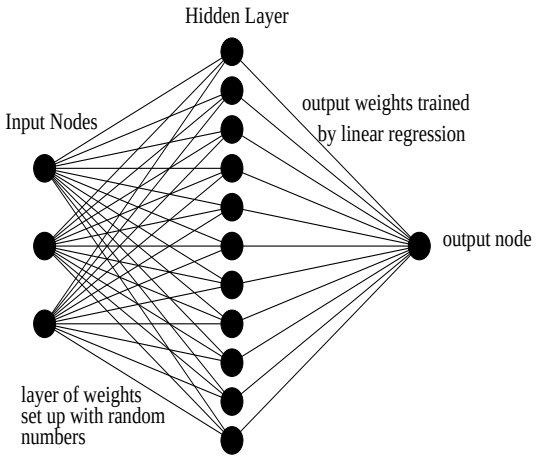


Figure 6.1: The structure of the random neural network.

The biggest difference between this network and other ANNs is that it does not make use of back-propagation or one of its variants for training. Instead a method similar to linear regression is used to train the network.

Like other networks, this one too is initialised to have random or quasi-random valued input weights. Training is then done by performing a matrix inversion to obtain the outputs of the hidden layer. This neural network has been shown to perform almost as well as most back-propagation methods [3], and its method of training is fast gaining recognition in the neural network community.

Using matrix notation, the Random Artificial Neural Network (RANN) can be described as

follows:

Let $\mathbf{X}$, $[N \times F]$ denote the inputs of a classifier[1], and $\mathbf{Y}$, $[1 \times N]$ the class labels of every observation. The matrix of input weights are represented by $\mathbf{W}$, $[F \times H]$ where H is the number of neurons in the hidden layer.

It was seen in Chapter 2 that the neural network hidden nodes normally have some non-linear transfer function. In this case, the tanh function is used. Thus every hidden node takes the sum of its inputs[2] and passes it through a tanh function.

The output of a hidden node in the network is thus given by

$$y_j = \tanh\left(\sum_{i=1}^{F} x_i w_{ij}\right) \tag{6.1}$$

where, x_i refers to the i-th input of every hidden node, w_{ij} refers to the weight between input node i and hidden node j and y_j refers to the output of the j-th hidden node.

Using matrix notation, the output $\mathbf{O}$ of the hidden layer can be written as

$$\mathbf{O} = \tanh(\mathbf{X}\,\mathbf{W}) \tag{6.2}$$

with $\mathbf{O}$ an $[N \times H]$ matrix.

To obtain the solution for the output weights of the network, a linear approximation to the problem is formulated. Representing the output weights by $\mathbf{V}, [1 \times H]$, the problem for which a solution is sought can now be defined as

$$\mathbf{V}\,\mathbf{O} = \mathbf{Y} \tag{6.3}$$

This forms a general set of linear equations that can be solved by finding a matrix inverse and post-multiplying. The solution for the output weights $\mathbf{V}$ then becomes

$$\mathbf{V} = \mathbf{Y}\mathbf{O}^{-1} \tag{6.4}$$

Since the matrix $\mathbf{O}$ would be (in most cases) non-square and thus not invertible, an approximate solution can be found using the pseudo-inverse or any other known technique for finding inverses of non-square matrices [3].

[1] N denotes the number of data points, and F the number of features. Thus the matrix is arranged with each feature represented in a different column and every sample from the data as a another row.

[2] A bias input is assumed to be added for every hidden node.

6.2 Practical Implementation

The problem of inverting matrices is, in a sense, an art in its own right. There do however exist several packages which use different techniques, such as singular value decomposition, in order to solve the problem quite effectively without too much computational cost. This does, of course, depend on the size of the training data.

In the experiments conducted here, the calculations were performed using the Matlab package, which provides the necessary tools for the problem. Training of these neural networks can be done under the Matlab environment in a sufficiently short time to allow exhaustive search of up to 18 features using up to 30 hidden neurons[3]. This process is usually more than an order of magnitude faster than most back-propagation routines, and also has the advantage of being less susceptible to over-training. This is due partly to the fact that no stopping criterion for the training process has to be specified, since training is done as a one-off process.

The RANN has been tested widely in the literature, and performs well on some standard problems[4] that are known to be difficult for even the best modified back-propagation algorithms.

6.3 Using the RANN for Feature Selection

Due to the speed and low user interaction requirements of the RANN, it can be used directly as a feature selection criterion function. In most of the tests done in the work presented here the method was fast enough to perform an exhaustive search of the input space using the RANN classification rate on the training data as a indicator for whether a certain subset performs well or not. The error rate of the classification can be used directly since, if the classifier is not over trained, this error rate would correspond closely to the error rate of the classifier on test data. The fact that the network is not trained until the mean-square-error reaches some user-specified value, as in other neural networks, cause these networks not to be over-trained. To appreciate this, the training and testing errors on some of the tests showed in Chapter 7 can be investigated.

Previous work by the author on the subject of classifier optimisation using this method [42] on a simple rotation-invariant character recognition problem showed promising results, and even yielded highly increased classification performance on that problem.

When the number of inputs become too high to handle exhaustive searches of the input space, other sub-optimal search techniques have to be resorted to. As mentioned in Chapter 2, the

[3] Again, this depends on the amount of training data as well.

[4] For example, tests on the difficult two-spiral problem [3] have shown the network to be very effective.

search strategy that was employed in this case was that of PBIL. It is expected that this search strategy would suffice for feature sets up to 100 or so features. When the number of features becomes much higher than this, the probability of finding a solution close to the optimal solution will become exceedingly small, and this could have the effect of not even reaching a very good local optimum.

6.4 The Three Feature Example

The RANN method was evaluated on the simple three feature example. It would be expected that the classifier would find a very good approximation to the data even using only single features for classification, and that almost perfect classification can be expected when using only features 1 and 2. The results of the experiment indicated that this was in fact the case. The uniform noise feature yields a classification close to 50% as expected, and decreases the performance of this classifier when added to the problem.

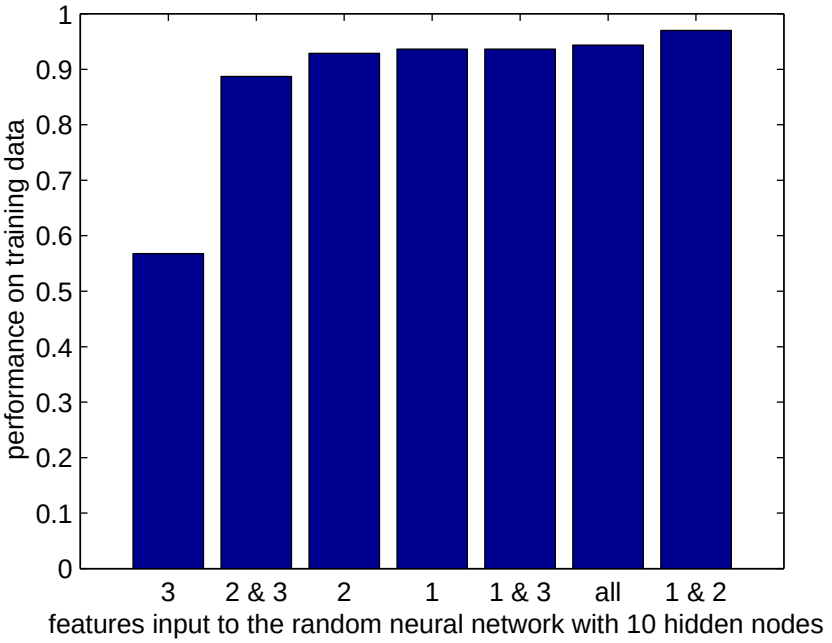


Figure 6.2: The classification results on all subsets of the three feature problem

6.5 Concluding Remarks

One of the assumptions made when using the training error rate as an indication of the usefulness of a particular subset is that this value is actually correlated to a high degree with

the expected error rate of the classifier. If the RANN method is to be used as a general feature selection tool for other types of neural networks or other types of classifiers, then the validity of this assumption would have to be tested.

It is known that most neural network techniques suffer from the problem of over-fitting, and that the training error rates of these classifiers do not correlate well with the final expected error. The RANN however, does not suffer from this problem. This is one of the main advantages of this type of neural network classifier above other neural network approaches.

From the experiments discussed in Chapter 7 it will be seen that this correlation between training error and test error of the RANN holds to some degree. Chapter 7 also compares the classification rates of this classifier to that of the k-Nearest Neighbour approach.

Chapter 7

Experiments and Results

In order to investigate each of the three methods described in the previous chapters, several tests and experiments were carried out. This chapter details these experiments and discusses some of the results obtained.

The chapter begins by discussing the data sets that were used in performing the experiments. The reasons for using these specific data sets are given, and details pertaining to each set are discussed.

Next, some initial experiments are reported that investigate the correlation between mutual information measurements, Gamma Test values, and apparent classification rates. The initial experiments were followed by a detailed search for the "best" feature subsets of the different data sets using all three selection methods. This is reported in Section 7.4. The tests use a complete search of the possible subsets for the Gamma Test and the RANN method[1]. The section that follows, shows how the PBIL approach can be used when the number of features becomes too high to allow a complete search of the input space. Finally, the chapter ends of with a discussion of the presented results.

7.1 The UCI data sets

Most of the tests and experiments were performed using data sets from the UCI Machine Learning Data Repository [60]. These data sets were chosen since they have enough data points to make probability density estimates in two dimensions valid to some extent. Furthermore, these data sets have also been used in other literature on feature selection and classification. This allows for the comparison of results on the work presented here and those

[1]The mutual information method never does an exhaustive search. This was explained in Section 4.3 and is elaborated upon in Section 7.3.2.

previous methods that used the same data sets. The data sets are briefly described below:

- The Pima-Indian diabetes (PIMA) data set has the goal of identifying whether a certain patient of Pima-Indian origin has diabetes or not. The majority class (no diabetes) in the data set has an occurrence of 65.10%. The set contains 768 data points and has 8 different input features.
- The Wisconsin Database on Breast Cancer (WDBC) is aimed at diagnosis of breast cancer. The data set contains 569 data points and has 30 input features. The majority class is 62.74% (no breast cancer). For some of the tests this data set was divided into 2 random partitions of the feature-space. These are called WDBC1 and WDBC2 and each of these data sets contain 15 features[2]. The partitioning was done in order to allow full searches of the feature-space on these sets. Strangely enough, the subdivisions made on the input-space still have the ability to give very high classification rates. Tests using the full WDBC data set and a PBIL search for the best embedding were also performed.
- The BUPA Medical Research Liver (LIVER) database contains 345 instances of 8 measurements each indicating whether a patient suffers from liver disorder or not. Majority class is 57.97%.
- The Ionosphere radar returns database (ION) contains 351 instances of classifications made from radar returns from the ionosphere. 34 features are present and the majority class label has an occurrence of 64.10%. This data set was used only in the PBIL search methods of Section 7.4.

Another reason for using these particular data sets is that all of the sets form two-class problems containing mostly continuous features. Before performing the different tests, the features were all scaled to range between values of -1 and 1. This scaling is normal practice when using neural network techniques, and does not degrade separability of the data significantly. More information on the data sets and the meanings of every feature can be found by downloading the information from [60] and also by looking at the data provided on the attached CDROM.

7.2 Correlation between Feature Selection Techniques and Classification

In order for any feature selection criterion to perform well, it must be highly correlated with the true obtainable error rate of the classifier. However, in most practical cases, the

[2]The WDBC1 data set consists of features 4,5,7,8,9,10,11,14,15,21,22,25,27,28,29 randomly chosen from the full set.

true obtainable error rate cannot be calculated. This leads to the approximation of using apparent classifier error rate as a measure of whether a particular selection criterion performs well or not. For this reason, it was decided to test the mutual information method and the Gamma Test method against the apparent classifier error rates of the RANN and k-Nearest Neighbour techniques.

7.2.1 Correlation between Mutual Information and the Random Artificial Neural Network

This experiment was aimed at testing correlation between mutual information values obtained from feature-to-output measurements of the different data sets, and the classification of these features using a RANN technique. The RANN was employed to infer the class labels of every data set using only a single feature at a time. The classification obtained in this way was then correlated with the mutual information content between every feature and the output.

The RANN was trained using different training and test sets chosen on a standard $\frac{2}{3}$ training, $\frac{1}{3}$ validation principle. The training set was also used in every case to determine an "optimal" number of neurons for each feature[3].

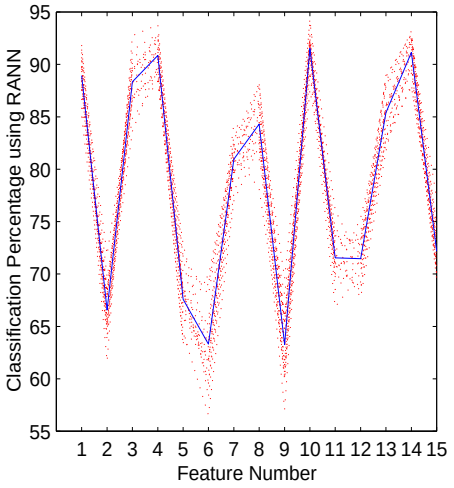
Mutual information was calculated by using only the training samples. This was done in order to make the tests as close to the real scenario as possible where, in general, the training set would represent the labelled data, and the test set would represent the new data not yet seen by the selection method. Thus, neither the mutual information measurements nor the training of the RANN was based on the test data.

The tests were performed twenty times on differing training and testing samples and the correlations between the mutual information values and the classifier estimated error rates on the *test sets* were calculated for each test. The average correlation was then calculated, and is reported in Table 7.1. Plots of these tests on the WDBC1, WDBC2, PIMA and LIVER data sets is shown in Figures 7.1 and 7.2.

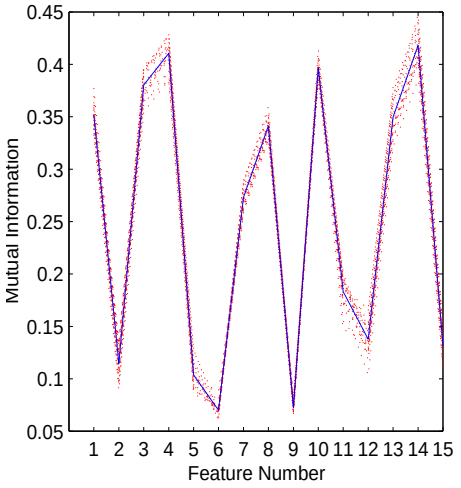
The classification rates as well as the mutual information measurements were also averaged over all test runs for every data set. These mean values are plotted as solid lines in Figures 7.1 and 7.2. The correlation of these mean values are shown in the third column of Table 7.1.

From Figures 7.1 and 7.2, and from Table 7.1 it can be seen that for the data sets with high classification rates on single features, the mutual information method shows almost perfect correlation with the classification accuracy. On the data sets for which the classification accuracies on the single features become very low however, the correlation between mutual in-

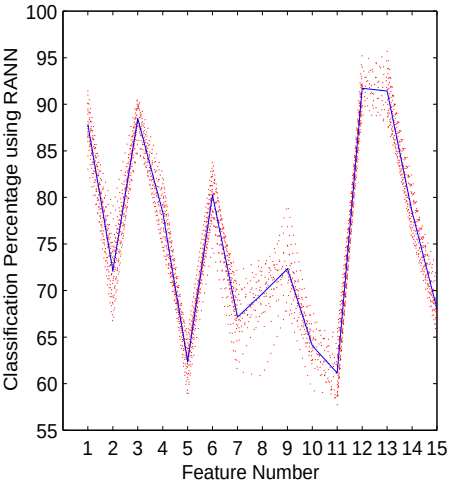
[3]This was done by again using several (between 30 and 50) search and evaluation sample sets obtained from the *training* data. The number of neurons that reported the highest classification results on the evaluation sample sets was then used to perform final classification.



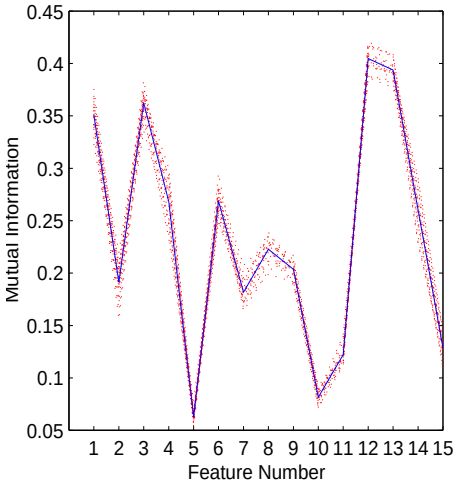
(a) RANN Classification on the WDBC1 data set features



(b) Mutual information on the WDBC1 data set features

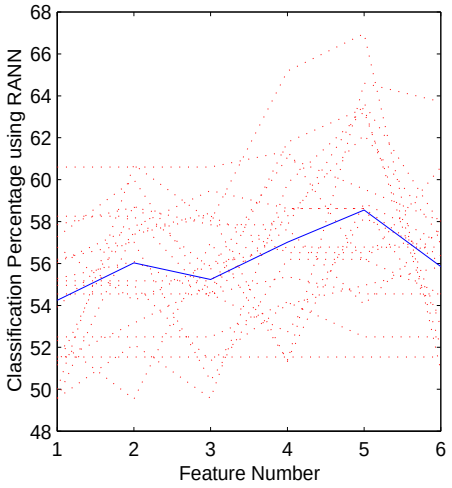


(c) RANN Classification on the WDBC2 data set features

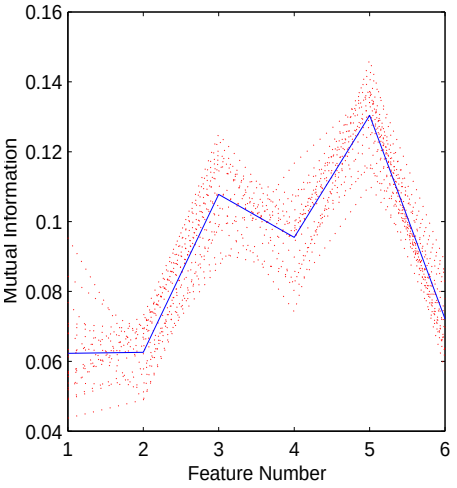


(d) Mutual information on the WDBC2 data set features

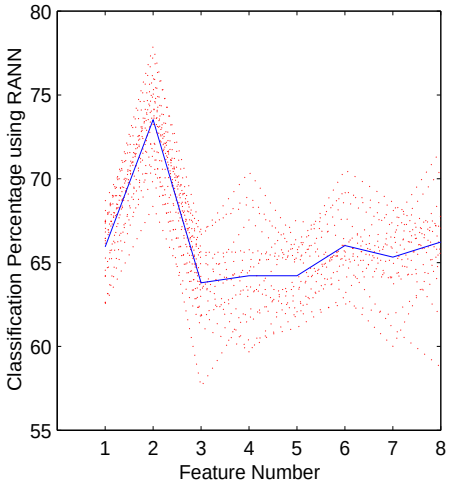
Figure 7.1: Comparison of mutual information with classification accuracy on single features from the WDBC1 and WDBC2 data sets using the RANN classifier.Plots on the left show classification accuracy and plots on the right show mutual information measurements. Points were joined with dotted lines to highlight the general tendency. The vertices of the dotted lines show the measured value. The solid line in every graph indicates mean values.



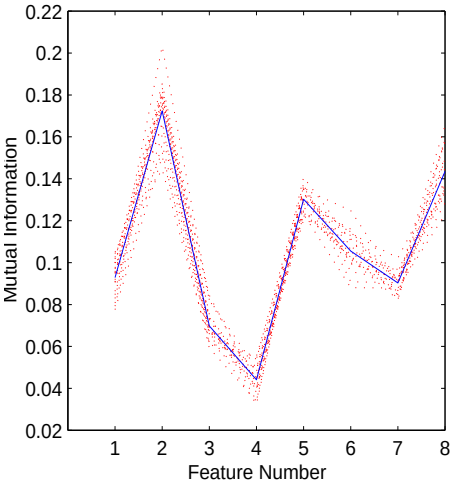
(a) RANN Classification on the LIVER data set features



(b) Mutual information on the LIVER data set features



(c) RANN Classification on the PIMA data set features



(d) Mutual information on the PIMA data set features

Figure 7.2: Comparison of mutual information with classification accuracy on single features from the LIVER and PIMA data sets using the RANN classifier.Plots on the left show classification accuracy and plots on the right show mutual information measurements. Points were joined with dotted lines to highlight the general tendency. The vertices of the dotted lines show the measured value. The solid line in every graph indicates mean values.

Table 7.1: The Correlation values obtained between mutual information measurements and RANN classification accuracies on single features from different data sets.

Data set	Average correlation between MI and classification	Correlation of average MI and average classification.
WDBC1	0.97	0.99
WDBC2	0.95	0.97
LIVER	0.18	0.72
PIMA	0.61	0.74

formation measurements and RANN accuracies are much lower (see, for example, the LIVER data set plots in Figure 7.2). This might suggest either that the mutual information technique performed poorly on these data sets, or that the classifier was not able to find consistent decision boundaries using only single features from the data sets. The mutual information measurements seem to be more consistent than the classifier error rates (as indicated by the lower per feature variance of the mutual information plots). This might indeed indicate that classifier inconsistency gave rise to these low correlations.

Another interesting observation is that the mutual information measurements show a general tendency to decrease as the classification rates decrease. This can be seen from the fact that the data sets that have high classification rates on the single features have much higher mutual information values, on average, than the ones with the lower classification rates.

The fact that three of the average correlation values from column two in Table 7.1 were above 0.6 is encouraging. The high correlation in these cases, coupled with the general tendency discussed in the previous paragraph are empirical justification that the mutual information method might have some inherent correspondence to classification rates obtained by a classifier. In previous tests, the RANN classifier has shown high correspondence in classification rates to other neural network classification methods, and thus this result might be generalisable to most neural network techniques[4].

Another encouraging result is that the mutual information estimates seem to be consistent over different samplings from the data sets. This means that the mutual information measure is not very badly affected by the different samplings made form the data, as long as these samplings contain enough data points from both classes to allow estimation of the underlying densities.

[4]The generality of this result is not investigated here. Investigation of this issue would entail detailed study and comparison of whether the correlation holds for many different classifiers. The aspect was therefore left for future research.

Table 7.2: Correlation between mutual information measurements and k-NN classification on single features from the UCI data sets.

Data set	Average Correlation between MI and classification	Correlation between average MI and average classification
WDBC1	0.97	0.99
WDBC2	0.95	0.98
LIVER	0.32	0.59
PIMA	0.56	0.72

7.2.2 Mutual Information and the k-Nearest Neighbour technique

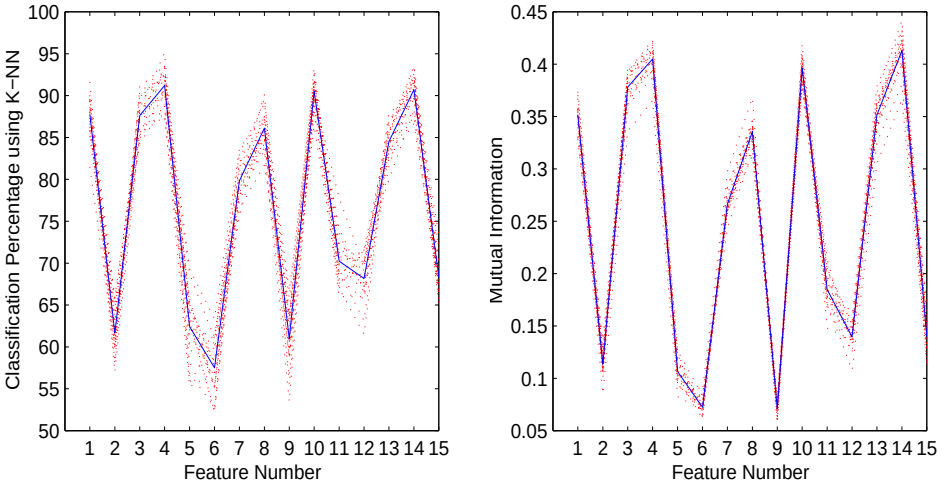
To investigate whether the discrepancies in the correlation between the RANN classification results and the mutual information estimates for the LIVER and PIMA data were due to inabilities of the RANN classifier, it was decided also to experiment with the correlation between mutual information and k-Nearest Neighbour techniques. If the k-Nearest Neighbour technique was to indicate high variance on these data sets, it would support the theory that the single feature data do not allow good separation of the classes.

The k-Nearest Neighbour classifier was implemented using a leave-one-out strategy on the training data to find the best number of neighbours to use for every evaluated feature[5]. Thereafter the k-Nearest Neighbour classifier was used to classify the test data. Mutual information measurements were made as discussed in the previous section.

The test was repeated over twenty training and testing samplings from the data. The reported correlation values were obtained by averaging over the different correlations found in each test. Plots of the results of this experiment are displayed in Figures 7.3 and 7.4. The average correlation values for every data set are shown in Table 7.2. As was done in the previous experiment, the measurements were also averaged over the different test sets, and the correlation between the averaged values was calculated.

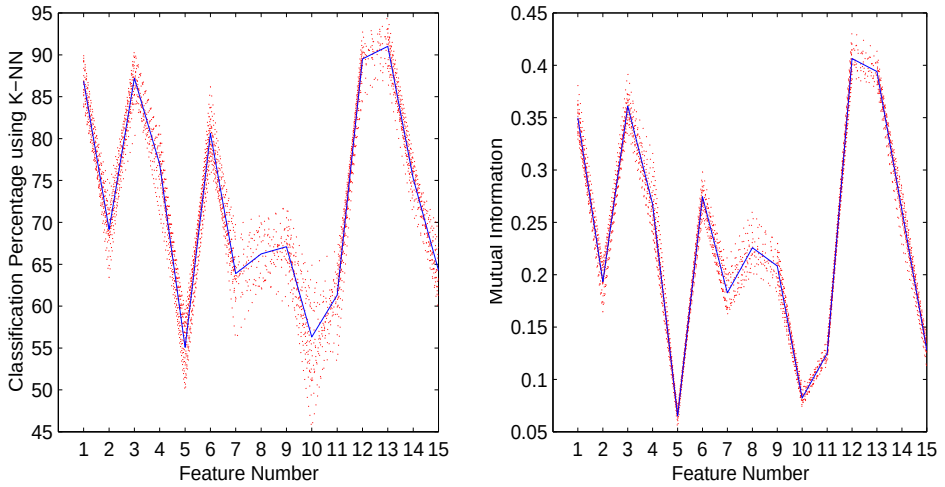
These experiments on the correlation of mutual information and k-Nearest Neighbour classification error yielded results much similar to that of the tests on the RANN. The mutual information estimates were again more consistent than the classification accuracies. The similarity of the results could indicate that the inconsistencies in the classification rates of the RANN were not solely due to inabilities of the neural network, but that using the single features on the LIVER and PIMA data sets does not lend itself to consistent classification of different data samplings.

[5]The range of neighbours considered were odd numbers between 1 and 11. Testing many more of these values makes the technique very computationally intensive.



(a) k-NN Classification on the WDBC1 data set features

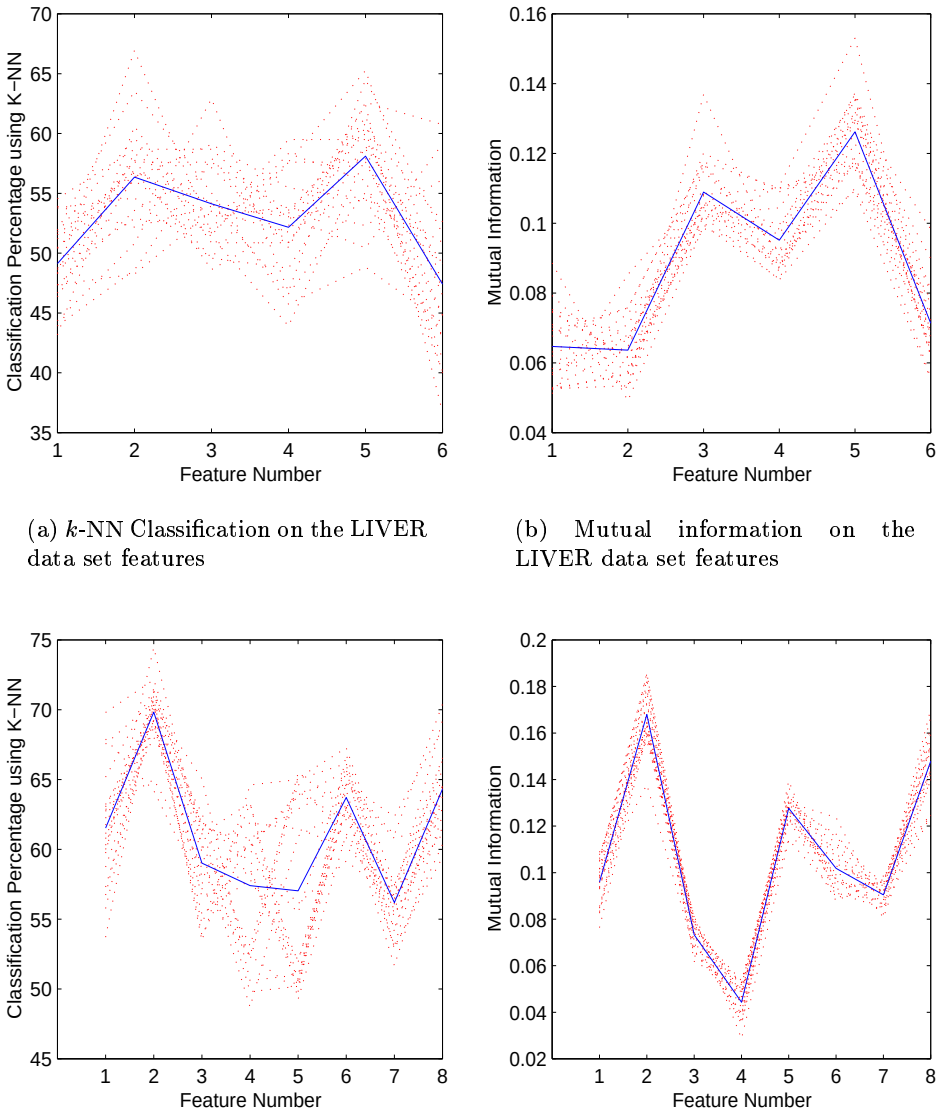
(b) Mutual information on the WDBC1 data set features



(c) k-NN Classification on the WDBC2 data set features

(d) Mutual information on the WDBC2 data set features

Figure 7.3: Comparison of mutual information with classification accuracy on single features from the WDBC1 and WDBC2 data sets using the k-NN classifier.Plots on the left show classification accuracy and plots on the right show mutual information measurements. Points were joined with dotted lines to highlight the general tendency. The vertices of the dotted lines show the measured value. The solid line in every graph indicates mean values.



(a) k-NN Classification on the LIVER data set features

(b) Mutual information on the LIVER data set features

(c) k-NN Classification on the PIMA data set features

(d) Mutual information on the PIMA data set features

Figure 7.4: Comparison of mutual information with classification accuracy on single features from the LIVER and PIMA data sets using the k-NN classifier.Plots on the left show classification accuracy and plots on the right show mutual information measurements. Points were joined with dotted lines to highlight the general tendency. The vertices of the dotted lines show the measured value. The solid line in every graph indicates mean values.

The fact that the mutual information calculations are based on a single feature premise, makes it difficult to investigate whether the method correlates with classification if more than one feature is used. It remains to be seen whether the heuristic approach of the mutual information ranking algorithm that uses only the feature-to-output and feature-to-feature based measures would suffice for handling inter-dependencies between different features. Section 7.3 investigates this more deeply by comparing the mutual information technique with the Gamma Test and RANN feature selection techniques.

7.2.3 Correlation between Gamma Test and Classification

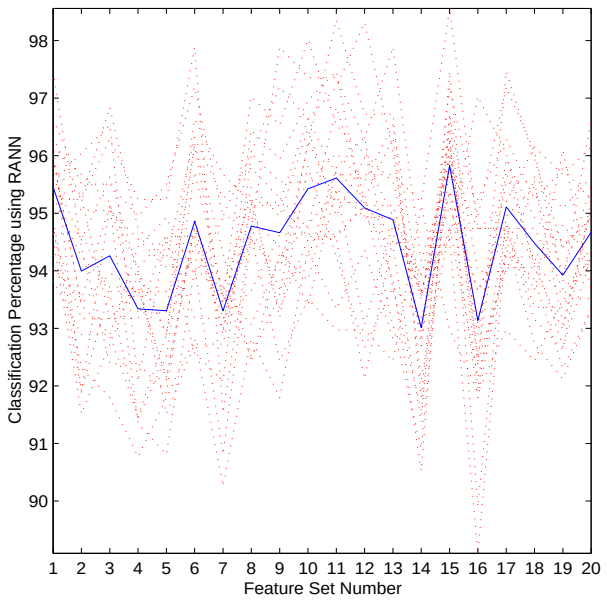
To investigate whether there exist correlation between the Gamma Test estimates and the RANN and k-Nearest Neighbour classifier error rates, it was decided to undertake experiments similar to those presented in the previous two sections. The fact that the Gamma Test is based on a nearest neighbour technique could make it more suited to the biases of the k-Nearest Neighbour classifier.

The Gamma Test takes any embedding as input (not just feature-output pairs like the mutual information calculation), and it was therefore decided to use 20 random embeddings from every data set to calculate correlation with both the RANN and the k-Nearest Neighbour classifier. The method used for classification was similar to that of the previous two sections except that the single features were now replaced with different subsets from each data set. Correlation was drawn between the Gamma estimate of every subset and the classification accuracy of the subset. An ideal correlation value of -1 was expected since the Gamma value should be small when the classification rate is high and vice versa.

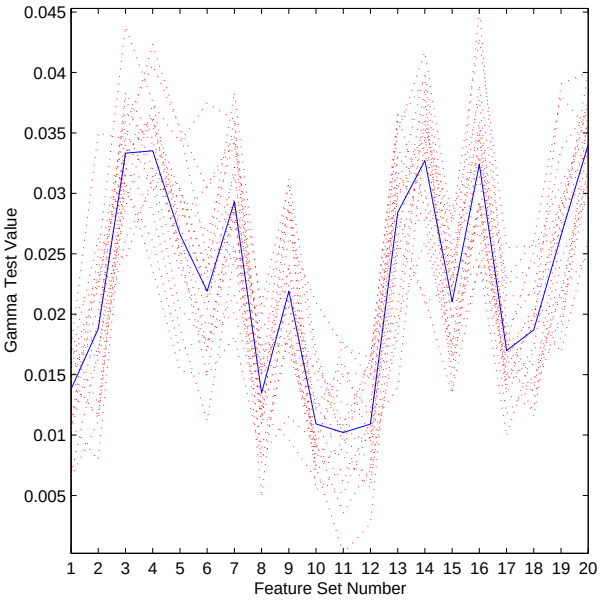
The classification values were correlated with the Gamma estimates for every test sampling. The average values of these are reported in the second columns of Tables 7.3 and 7.4. As with the tests of the previous two sections, the classification rates and Gamma values were also averaged over the different test samplings and then correlation of these averaged values were calculated. This is reported in the third column of each of these tables. Plots of the different data sets are shown in Figures 7.5, 7.6, 7.7 and 7.8.

The results of the experiments confirm that the Gamma Test is negatively correlated with the classifier apparent error rate for the RANN and k-Nearest Neighbour classifiers. On the WDBC1 data set the k-Nearest Neighbour classifier showed inconsistent performance over the different samplings made from the data for some of the feature sets. This is the cause of the very small negative correlation values obtained here. However, the WDBC2 data set showed surprising high correlation values. This was due to the fact that the last subset contained only three features[6], resulting in significantly worse classification when compared to the other

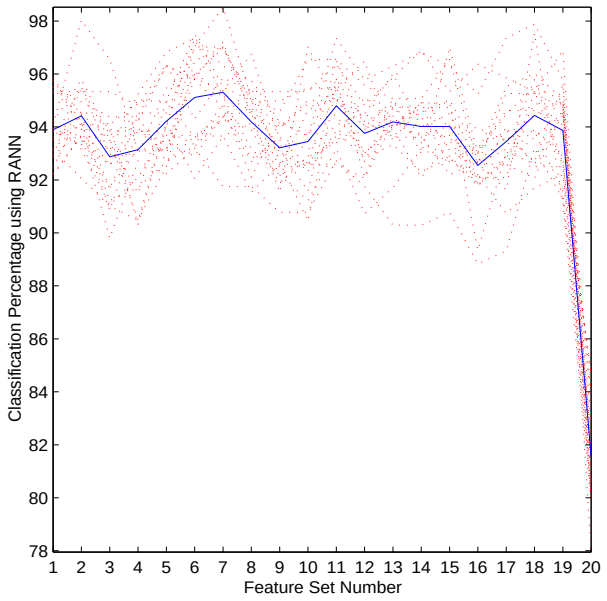
[6] A list of the different subsets used in these tests can be found in Appendix B.



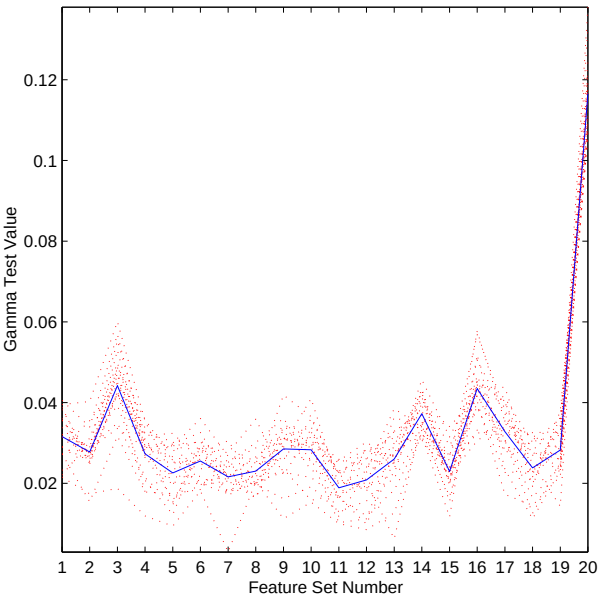
(a) RANN Classification on the WDBC1 data set feature subsets



(b) Gamma estimates on the WDBC1 data set feature subsets

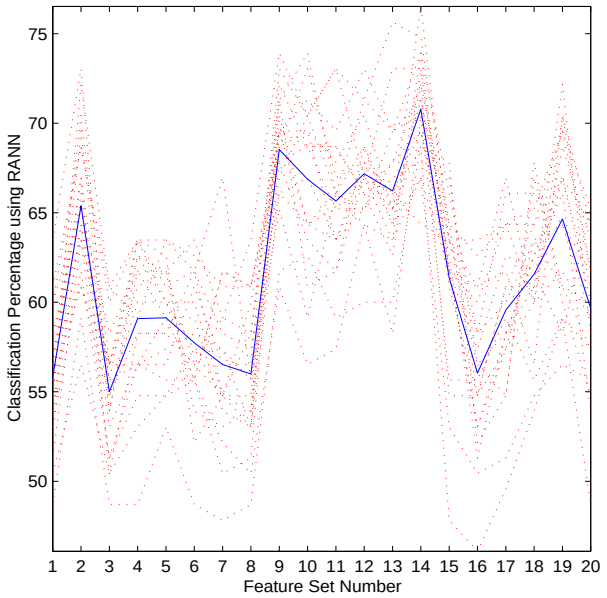


(c) RANN Classification on the WDBC2 data set feature subsets

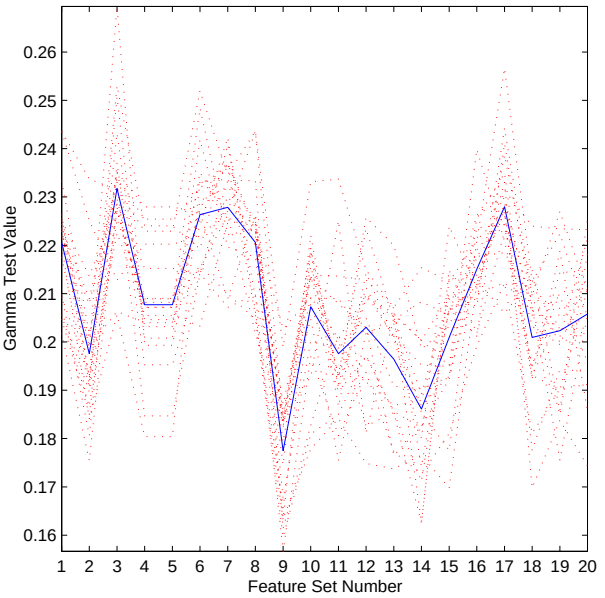


(d) Gamma estimates on the WDBC2 data set feature subsets

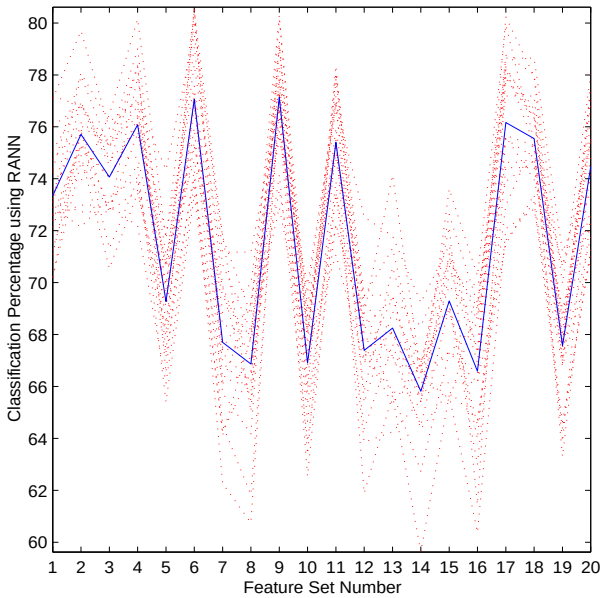
Figure 7.5: Comparison of Gamma Test with classification accuracy on random feature subsets from the LIVER and PIMA data sets using the RANN classifier. Plots on the left show classification accuracy and plots on the right show Gamma estimates. Points were joined with dotted lines to highlight the general tendency. The vertices of the dotted lines show the measured value. The solid line in every graph indicates mean values.



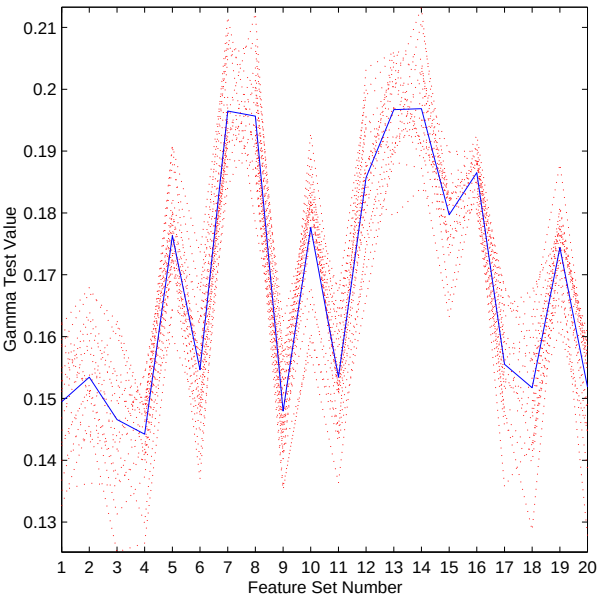
(a) RANN Classification on the LIVER data set feature subsets



(b) Gamma estimates on the LIVER data set feature subsets

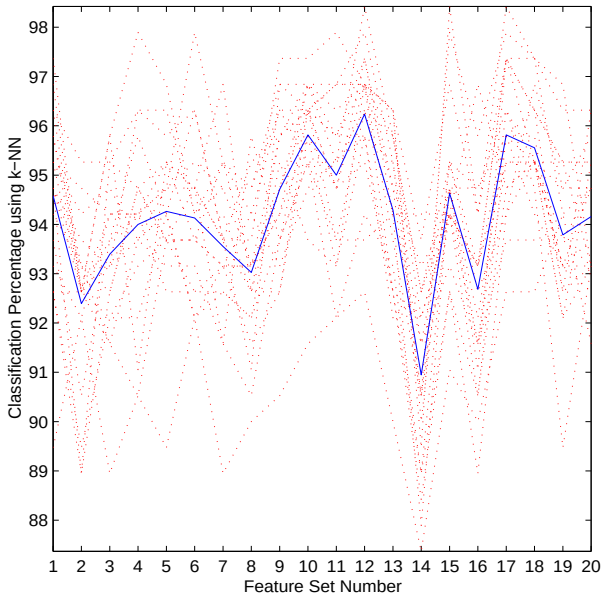


(c) RANN Classification on the PIMA data set feature subsets

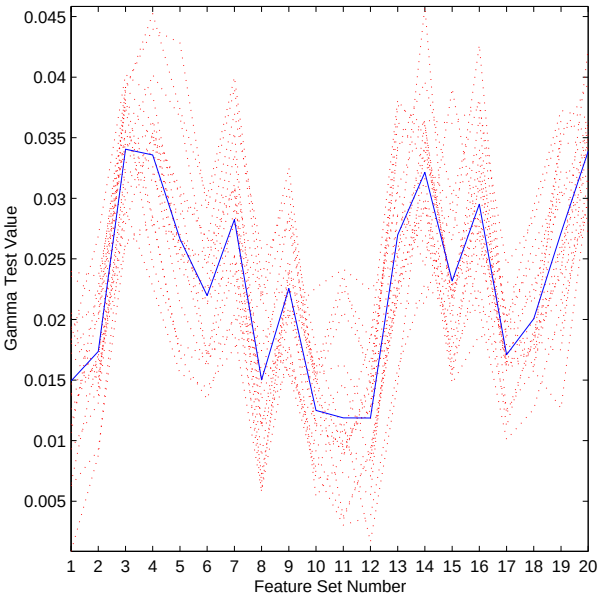


(d) Gamma estimates on the PIMA data set feature subsets

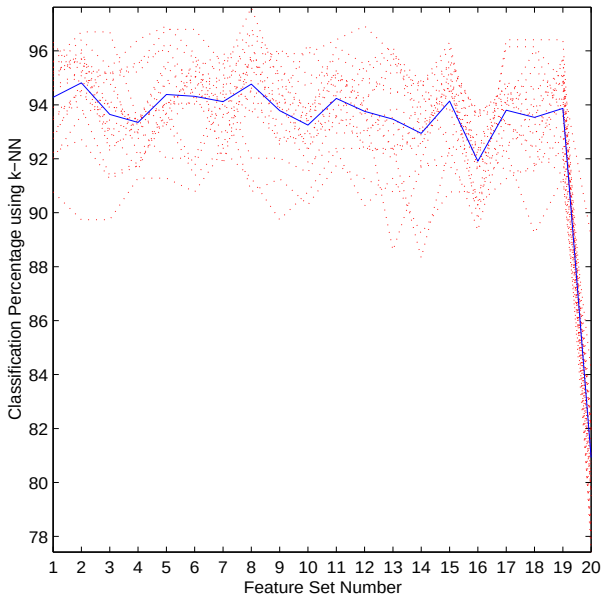
Figure 7.6: Comparison of Gamma Test with classification accuracy on single features from the LIVER and PIMA data sets using the RANN classifier.Plots on the left show classification accuracy and plots on the right show Gamma estimates. Points were joined with dotted lines to highlight the general tendency. The vertices of the dotted lines show the measured value. The solid line in every graph indicates mean values.



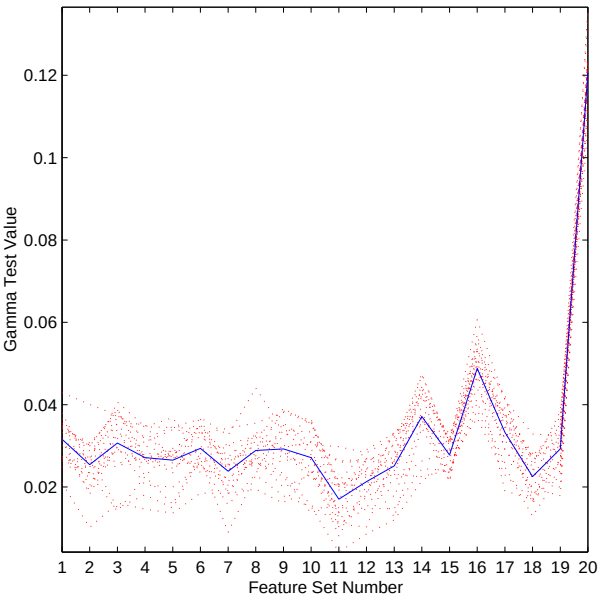
(a) k-NN Classification on the WDBC1 data set feature subsets



(b) Gamma estimates on the WDBC1 data set feature subsets

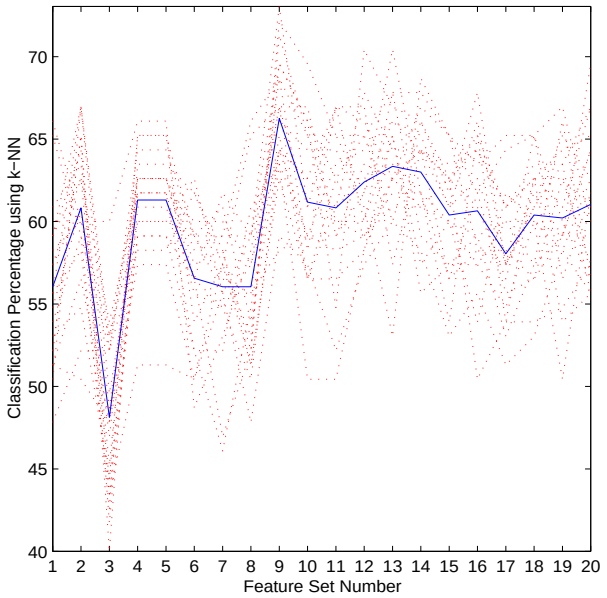


(c) k-NN Classification on the WDBC2 data set feature subsets

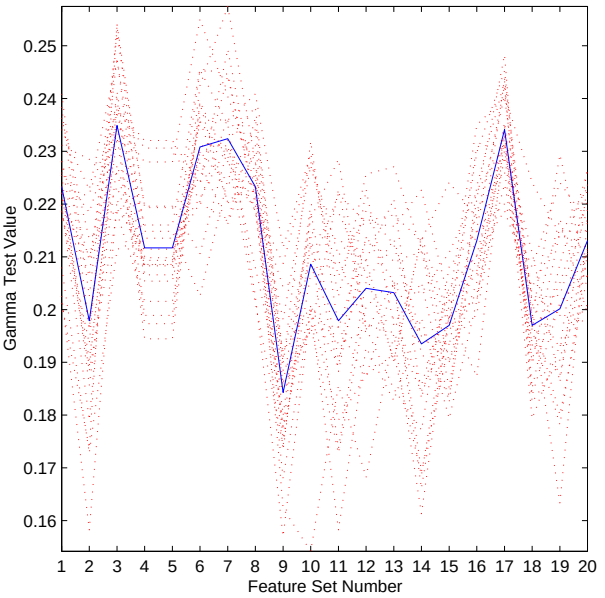


(d) Gamma estimates on the WDBC2 data set feature subsets

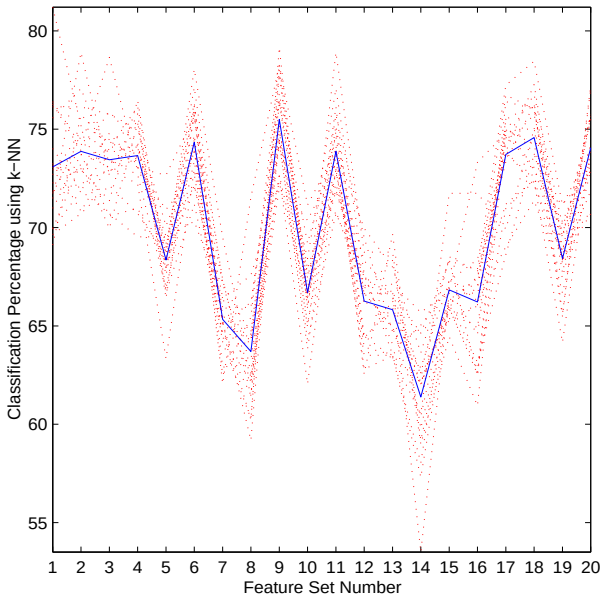
Figure 7.7: Comparison of Gamma Test with classification accuracy on single features from the WDBC1 and WDBC2 data sets using the k-NN classifier.Plots on the left show classification accuracy and plots on the right show Gamma estimates. Points were joined with dotted lines to highlight the general tendency. The vertices of the dotted lines show the measured value. The solid line in every graph indicates mean values.



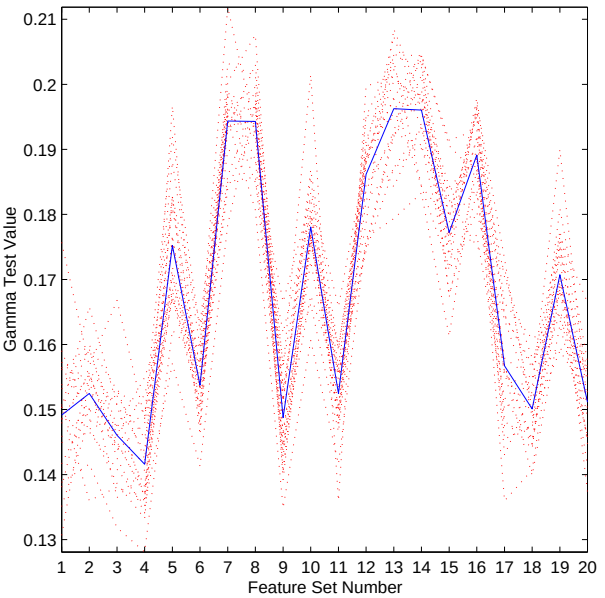
(a) k-NN Classification on the LIVER data set feature subsets



(b) Gamma estimates on the LIVER data set feature subsets



(c) k-NN Classification on the PIMA data set feature subsets



(d) Gamma estimates on the PIMA data set feature subsets

Figure 7.8: Comparison of mutual information with classification accuracy on single features from the LIVER and PIMA data sets using the k-NN classifier.Plots on the left show classification accuracy and plots on the right show Gamma estimates. Points were joined with dotted lines to highlight the general tendency. The vertices of the dotted lines show the measured value. The solid line in every graph indicates mean values.

Table 7.3: Correlation between Gamma Test and RANN classification on random feature subsets from the UCI data sets.

Data set	Average correlation between Gamma estimate and classification	Correlation between average Gamma estimate and average classification
WDBC1	-0.35	-0.72
WDBC2	-0.87	-0.97
LIVER	-0.55	-0.85
PIMA	-0.82	-0.92

Table 7.4: Correlation between Gamma Test and k-NN classification on random feature subsets from the UCI data sets.

Data set	Average correlation between Gamma estimate and classification	Correlation between average Gamma estimate and average classification
WDBC1	-0.25	-0.57
WDBC2	-0.89	-0.97
LIVER	-0.34	-0.81
PIMA	-0.81	-0.96

tested subsets. The high Gamma Test value for this subset clearly indicates that this set would not result in high classification results. The absolute correlation between the mean Gamma Test values and mean classification values (column 3 in Tables 7.3 and 7.4) were all above 0.5, indicating that the Gamma Test, on average, seems to be related to classifier error rates, and as such would be useful as a feature selection criterion function.

In comparing the correlation values for the k-Nearest Neighbour classifier and the RANN technique, it is seen that the values do not differ significantly. This indicates that the method is consistent for different types of classifiers, and confirms that the Gamma estimates capture some inherent properties of the data which allow the estimates to be related to classifier performance. For this reason, the Gamma Test method would be usable as a general filter feature selection tool.

In the next section, the three methods are used as feature selection techniques on the different UCI data sets, and the resulting subsets are compared.

7.3 Comparison between the methods: Selecting Features on the UCI Data Sets

The three methods were tested using complete search on four of the data sets from the UCI data repository. This section details the experimental method for these tests and reports and discusses the selection results obtained.

7.3.1 Experiments on the Gamma Test

The aim of this experiment was to find, using the Gamma Test, those subsets that were expected to give the best classification on the UCI data sets. The subset found would then be tested on a classifier to see whether they really perform well. The classification values for every subset are also compared to those of the full set, to investigate whether any of the subsets actually outperform the whole data set.

The four data sets were each partitioned into 10 sample sets of training and testing data. Each of the training sample sets was then tested using a maximum number of nearest neighbours (P_{max} from equation 5.4) varying in steps of 5 from 20 to 50. This test was performed over all subsets of features for every training data sample from each data set.

From every specific value for P_{max}, the best *ten* embeddings were found by choosing those subsets that resulted in the lowest estimates of the Gamma values. This was done for each different training sampling that was made for every data set[7]. The embeddings were then ranked according to

$$rank_{embedding} = \sum_{k=1}^{freq} \frac{1}{0.1 + \gamma} \tag{7.1}$$

where $freq$ is the number of times a specific embedding was placed amongst the top 10 embeddings.

Thus, embeddings that occurred more frequently amongst the top ten in the 70 tests were considered to be better, but embeddings occurring the same number of times were rated according to the average estimate of the γ value. The embeddings found were then subjected to rigorous classification tests using the RANN classifier.

For the above mentioned classification, the training set was first used to estimate the best number of neurons needed to classify the data for every embedding. This was done by sampling the *training* data into $\frac{2}{3}$, $\frac{1}{3}$ partitions again, and training the classifier with a different number of neurons each time on the $\frac{2}{3}$ partitioning of the training data while testing the performance

[7]Thus there were 70 evaluations done for every subset of features from a specific data set: 10 for the different training samples and, on these, 7 for each value of P_{max}.

of the classifier on the $\frac{1}{3}$ test portion[8]. Once the best number of neurons had been decided on, a classifier using this number of neurons was trained, and then used to classify the final test set associated with that training set. The top *five* subsets for every data set found in this experiment are reported in Table 7.5, ranked in order of best results on the final test sets. The full embedding is also added for comparative study. The average Gamma estimate that was obtained for each set is also shown.

Table 7.5: Top 5 embeddings obtained using the Gamma Test.

Data set	Features used	Rank	% avg. training performance	Std. dev of training %	% avg. test set performance	Std. dev on test sets	Avg. γ estimate $\times 10^{-3}$
	101111	205.22	73.19	2.40	**73.19**	2.49	181.49
L	011111	114.38	73.21	2.05	73.19	3.72	203.53
I	001111	236.65	73.07	2.45	72.87	3.16	183.09
V	101110	203.44	72.77	2.13	71.70	3.30	202.18
E	000111	222.17	71.57	2.42	70.41	3.47	178.83
R	111111	n/a[9]	72.96	1.69	**71.82**	2.85	212.29
	11100111	53.49	78.90	1.40	**76.48**	2.57	142.4
P	11101111	54.65	79.14	1.40	76.37	2.58	126.5
I	11001111	35.10	79.26	1.81	76.22	2.56	140.1
M	11000111	39.50	78.89	0.98	76.05	2.73	143.1
A	11000101	39.52	78.07	1.10	75.06	1.10	142.5
	11111111	36.00	78.55	1.51	**75.99**	2.19	142.2
W	001011111111001	130.21	96.89	0.72	**96.49**	1.64	4.66
D	101011111111001	111.52	96.93	0.49	96.32	1.65	-5.09
B	001011101111001	110.9	96.99	0.66	96.25	1.34	4.90
C	110011111111101	126.37	97.18	0.77	95.97	1.68	-9.23
	110011101111001	135.42	96.35	0.51	95.74	1.73	-10.76
1	111111111111111	n/a	97.37	0.87	**96.04**	1.78	9.20
W	010011111110101	72.46	97.61	1.06	**95.62**	1.14	10.05
D	010010111110101	72.52	97.35	0.70	95.40	1.24	9.33
B	010010011110101	214.04	97.08	0.77	94.93	1.21	10.46
C	011011110111111	84.62	96.54	0.59	94.45	1.23	13.04
	010111011110111	74.15	96.67	0.71	94.44	1.12	5.59
2	111111111111111	n/a	96.89	1.06	**95.22**	1.29	14.32

From Table 7.5 some interesting observations can already be made. It is seen that in most cases the sets provided by the Gamma Test, performs equally well or better than, the full set of features. This happens even though in some cases a much smaller number of features are utilised by the classifier.

A second observation confirming the ideas of Chapter 5 is that the average gamma values tend to decrease as the classification rates increase. This can be seen by comparing the Gamma values for every data set with the classification rates of the data sets. The tests therefore

[8]These tests were carried out 30 times for every hidden layer size. The hidden layer sizes ranged from 5 to 50 neurons.

supports the notion that the Gamma value is in some way related to classification error.

However, the tests were not without hick-ups. The average Gamma values for the WDBC1 data set show some negative values. The sets showing these negative averages however do not seem to perform better then the other data sets. As was discussed in Chapter 5, these negative values are due partly to regression error in estimating the straight line fit, and partly due to finite data sample size, giving rise to non-linear effects in the curves from which the estimation must be made. The results obtained here might indicate that negative values for the Gamma Test should just be treated as zeroes, possibly indicating that the subset should allow perfect classification. Another aspect that could shed some light on the problem might be to investigate the slope of the line as well and include this in the ranking criterion, since a set with high slope indicates big class difference with small input distance change, and thus less separability in the data. This was not investigated here and is left for future research.

Comparing the Results to a k-Nearest Neighbour Classifier

In comparing the classification obtained in the previous section to the k-NN technique, the training set was first used to estimate the best value of k to use for every embedding. This was done by using a leave-one-out strategy on the *training* data and searching through increasing values for k from a predefined set of values[10]. Using the best value found in this way the classifier was then used to classify the unseen test data to estimate the classifier error for the subset data.

The whole process was done for each of the top 5 sets obtained in the previous section in order to compare the performance of the two types of classifiers. Results are reported in Table 7.6. The full embedding is again added for comparative study.

It is clear that the subsets found by the Gamma Test do not only perform well on the RANN but also performs well on the k-Nearest Neighbour approach. In a sense this could have been expected, since the basis of the Gamma approach is closely related to the k-NN technique. A surprising fact is that for the LIVER data set, the k-NN technique shows very high increases in classification rates when using the Gamma Test feature sets. This supports arguments of Chapter 2, that the k-NN classifier can be highly affected by noisy features or inconsistency in features.

Another interesting observation is that the RANN classifier seems to outperform the k-NN technique on almost all of the sets reported. This seems to indicate that the RANN classifier can, on average, outperform the k-NN technique. For the RANN, the drop in classification rate from training to test data are also not significant, indicating that the RANN classifier

[10] As in previous sections, the different values for k were odd values ranging between 1 and 11.

Table 7.6: Top 5 embeddings from Gamma Test, evaluated on the k-NN classifier.

Data set	Features used	% avg training performance	Std. dev of training %	% avg test performance	Std. dev of test sets
	101111	65.27	1.32	65.53	2.29
L	011111	66.17	2.46	66.58	3.39
I	001111	67.05	1.57	67.57	2.37
V	101110	66.66	2.48	**68.52**	2.09
E	000111	63.61	1.52	62.97	3.08
R	111111	63.31	2.42	**62.97**	3.44
	11100111	76.60	1.60	74.96	2.76
P	11101111	76.80	1.69	74.20	3.03
I	11001111	76.84	1.85	**75.35**	2.23
M	11000111	77.03	1.54	74.64	3.12
A	11000101	76.64	1.54	73.52	2.40
	11111111	75.97	1.64	**74.22**	2.54
W	001011111111001	96.77	0.75	96.05	1.67
D	101011111111001	96.83	0.82	**96.43**	1.58
B	001011101111001	96.83	0.80	95.94	1.60
C	110011111111101	96.62	0.51	95.88	1.31
	110011101111001	95.85	0.94	95.09	1.53
1	111111111111111	96.88	0.62	**95.81**	1.33
W	010011111110101	94.23	0.68	93.44	1.68
D	010010111110101	93.96	0.75	93.53	1.46
B	010010011110101	94.66	0.72	93.58	1.76
C	011011110111111	95.33	0.65	**94.39**	1.22
	010111011110111	94.16	0.32	92.63	1.79
2	111111111111111	95.35	0.43	**93.79**	1.49

was not affected too much by over-fitting problems[11].

7.3.2 Tests Using Mutual Information

For the mutual information tests, it was decided to use a value for α similar to that used by Battiti. The value used in the tests are 0.5. The fact that the tests are carried out over multiple data samplings combined with the experimental method used, made the subset selection a bit more immune to the particular choice of this value.

The mutual information ranking algorithm was applied to all 4 of the different data sets. Each ranking was performed on every one of the 10 different training samplings made for every data set. This resulted in ten rankings of feature importance for every data set. From these ten (possibly different) rankings the subsets used for evaluation on the RANN and k-NN classifiers were then chosen by selecting from the rankings all subsets of 2 features up to all

[11] The reader might be interested to know the number of neurons used for the RANN classifier. This cannot be easily reported, as the number of neurons varied in most of the test runs as was explained earlier. In most cases however, the number of neurons would not have exceeded 40.

subsets of $n-1$ where n is the number of features in the data set. Thus for example, if a ranking of 5,2,6,1,3,4 was obtained in one of the runs, the subsets for evaluation would be {5,2},{5,2,6},{5,2,6,1} etc. If the rankings differed amongst the ten different samplings, then those subsets would be added to the ones found previously.

The subsets found by the above method were then tested on the RANN classifier using the same tests as described for the Gamma test. The results of the 5 subsets with the highest classification rates on the RANN tests are shown in Table 7.7.

Table 7.7: Top 5 embeddings obtained using mutual information measurement ranking, and RANN classification.

Data set	Features used	% avg training performance	Std. dev of training	% avg test performance	Std. dev of test test sets
L	111011	69.10	4.87	**67.83**	3.52
I	010010	65.22	3.79	63.25	3.25
V	110111	65.73	2.57	62.91	4.87
E	110011	68.93	1.99	62.08	3.64
R	100010	64.08	2.44	61.21	3.23
	111111	74.08	4.63	**73.25**	4.42
	01000111	78.17	1.38	**76.55**	2.58
P	01000101	78.32	0.99	76.36	2.24
I	01001111	78.48	1.27	76.34	1.92
M	11101111	78.62	1.37	76.32	2.34
A	01101111	78.69	1.74	76.24	2.60
	11111111	78.92	1.70	**76.44**	2.25
W	111011111011011	97.08	0.83	**96.96**	1.09
D	110011111011111	97.04	0.71	96.60	1.31
B	111011101011011	96.91	0.72	96.60	1.26
C	110011011011011	96.95	0.55	96.52	1.29
	110011101010011	96.86	0.48	96.52	1.63
1	111111111111111	97.02	0.58	**96.57**	1.52
W	110011001111011	96.59	0.54	**95.50**	1.30
D	110111011111001	96.59	0.52	95.34	1.17
B	010000000001001	96.17	0.66	95.21	1.00
C	110011011111111	96.64	0.43	95.19	1.76
	110111011111011	96.59	0.81	95.12	1.50
2	111111111111111	96.45	0.67	**94.80**	1.53

The mutual information method was not always able to find subsets that performed as well as the full set of features. In the case of the LIVER data set, this is particularly clear. The highest classification rates of the mutual information method is more than 6% lower than that of the full set. The problem can probably be ascribed to the fact that the mutual information method uses a heuristic ranking of features, and does not allow a full search of all subsets. In the case of the LIVER data set, this resulted in the fact that none of the subsets found in the previous Gamma Tests, were ever evaluated on the classifier.

In order to investigate whether the poor classification accuracy of the mutual information subsets on the RANN was not due to inability of the random weights of the RANN, it was again decided to compare the classification of the subsets from Table 7.7 to the k-Nearest Neighbour technique. As previously this was done by first finding good values for k on the training set, and then classifying the test sets using these values.

Table 7.8: Classification of Top 5 embeddings from Table 7.7 on a k-Nearest Neighbour classifier.

Data set	Features used	% avg training performance	Std. dev of training	% avg test performance	Std. dev of test test sets
L	111011	63.50	1.50	**62.26**	3.52
I	010010	61.40	2.37	59.75	2.67
V	110111	61.09	2.21	61.35	4.75
E	110011	61.52	1.99	59.23	2.60
R	100010	62.50	2.20	60.05	6.16
	111111	63.31	2.42	**62.97**	3.44
	01000111	77.98	1.19	76.39	2.41
P	01000101	77.58	1.41	74.88	2.28
I	01001111	77.81	1.19	76.44	2.25
M	11101111	76.80	1.69	74.20	3.03
A	01101111	77.95	1.15	**77.24**	1.56
	11111111	75.97	1.64	**74.22**	2.54
W	111011111011011	96.08	0.66	**95.50**	1.44
D	110011111011111	95.58	0.74	94.09	1.50
B	111011101011011	95.90	0.72	95.20	1.56
C	110011011011011	95.19	0.71	94.49	1.44
	110011101010011	95.69	0.63	94.81	1.55
1	111111111111111	96.88	0.62	**95.81**	1.33
W	110011001111011	94.89	0.66	93.88	1.65
D	110111011111001	94.91	0.68	93.20	1.45
B	010000000001001	95.54	0.66	**94.78**	0.81
C	110011011111111	95.36	0.61	94.40	1.44
	110111011111011	94.75	0.59	93.52	1.25
2	111111111111111	95.35	0.43	**93.79**	1.50

The k-Nearest Neighbour technique showed that the subsets chosen by the mutual information algorithm for the LIVER data set do not perform well. None of the chosen sets outperforms the full-set, which happened to significant extent for most of the Gamma Test subsets on the LIVER data set. For the other data sets, the two methods showed much similar results.

An interesting fact is that the methods do not seem to find the same subsets for the data sets, yet these subsets seem to perform well. At first glance, this appeared strange, but it was soon realised that the none of these data sets had any known significantly optimal value. In most data sets from real life problems this would be the case, and the problem does not end up to be that of finding the optimal solution, but that of finding a solution that is acceptable. This notion might lead the reader to believe that that any random choice of features would

be good. However this is not correct. From the fact that the mutual information method showed subsets that performed significantly worse than the full set on both the k-NN and the RANN classifiers, even though they contained almost all features, it can be gathered that a simple random selection of features would not be acceptable. However, the notion of random search is one worth investigating, and is performed later in the chapter by using the PBIL stochastic search approach.

7.3.3 Wrapper Experiments using the RANN

In using the Random Artificial Neural Network to select feature subsets, the method followed was to classify each of the training sets using a network that had a low number of hidden nodes and employ the classification rate obtained as the criterion function for selection. Because a search for the optimal number of nodes for every subset would be too time consuming, it was decided to keep the number of hidden nodes low so as to prevent over-fitting of the data, giving a false estimation of how easy the embedding would be to classify on[12].

From these sets, the 10 ones that on average performed best on the training data were then selected for further testing on the final classifier. Each of these sets, as with the Gamma test, were then tested to find the best number of neurons for the classification and finally used to classify the test sets. The top 5 embeddings found this way is reported in Table 7.9.

From Table 7.9 it can be seen that the technique of using the training error of the RANN classifier as the criterion for feature selection shows some merit. The training rates of the classifier and the final test errors do not show perfect correlation, but at least the subsets found in this way never show significantly poor performance. The fact that the RANN does not need a stopping criterion in the training stage, serves to its advantage, and makes it useful for feature selection for artificial neural networks. If the training error of a normal back-propagation network was to be used, this method would be very difficult to implement, since the back-propagation method could be allowed to over fit the data, and thus end up having close to 0% error on the training set.

Comparing the above results with that obtained by the Gamma Test and the mutual information method on RANN classification, it can be seen that the RANN selection technique did not vastly outperform the other two methods. The fact that the RANN wrapper technique had included to it the biases of the classifier, proves not to be such a big advantage over the other methods. This indicates that the Gamma Test, and mutual information techniques does capture some of the inherent properties in the data that makes a specific subset either

[12]Typically a value of 20 hidden neurons or lower was used. When the number of features become very high this technique would have to be adapted since it might become impossible to build descriptive decision boundaries with so little neurons.

Table 7.9: Top 5 embeddings obtained using the RANN training error rate in a wrapper approach.

Data set	Features used	% avg training performance	Std. dev of training	% avg test performance	Std. dev of test sets
	001111	73.63	1.63	**73.35**	2.60
L	101110	72.40	1.88	72.74	2.87
I	011111	74.94	2.21	72.41	4.93
V	001110	70.57	1.80	72.12	2.96
E	101111	73.95	2.07	72.01	3.10
R	111111	74.20	1.65	**74.68**	3.11
	11101110	78.00	1.72	**76.96**	2.89
P	01101111	79.33	1.47	76.84	2.29
I	11111110	78.00	1.43	76.82	2.47
M	01000111	78.03	1.59	76.72	2.98
A	11100110	77.84	1.50	76.71	2.96
	11111111	78.55	1.28	**75.99**	2.17
W	101101110111011	97.42	0.54	**96.55**	1.56
D	001101110111001	97.23	0.60	96.54	1.71
B	100101100111011	97.23	0.67	96.53	1.51
C	001111110111010	97.65	0.57	96.45	1.62
	100000011110010	97.18	0.49	96.43	0.93
1	111111111111111	97.37	0.58	**96.04**	1.46
W	010000001111101	96.883	0.51	**95.89**	0.87
D	110000011111111	97.088	0.66	95.63	0.66
B	010100001111001	96.431	0.48	95.51	1.03
C	110000101101110	97.198	0.72	95.50	1.24
	110000001111001	96.597	0.61	95.47	0.63
2	111111111111111	97.07	0.51	**95.03**	1.39

good or bad for classification. Furthermore, it confirms the high correlation values obtained in Sections 7.2 and 7.2.3.

To be consistent with the previous tests and for comparative reasons, the top five subsets of Table 7.9 was again tested using a k-Nearest Neighbour technique. Results from these tests are tabulated in Table 7.10.

Table 7.10: Classification of top 5 embeddings from Table 7.9 using a k-Nearest Neighbour technique.

Data set	Features used	% avg training performance	Std. dev of training	% avg test performance	Std. dev of test sets
	001111	67.05	1.57	67.57	2.35
L	101110	66.66	2.48	68.52	2.09
I	011111	66.17	2.47	66.58	3.39
V	001110	68.87	1.92	**69.94**	2.26
E	101111	65.27	1.32	65.53	2.29
R	111111	63.31	2.42	**62.97**	3.44
	11101110	76.41	1.25	75.56	2.30
P	01101111	77.95	1.15	**77.24**	1.56
I	11111110	75.21	1.29	72.64	3.01
M	01000111	77.98	1.19	76.39	2.41
A	11100110	76.66	1.37	75.57	2.32
	11111111	75.97	1.64	**74.22**	2.54
W	101101110111011	96.88	0.45	95.94	1.55
D	001101110111001	97.04	0.60	**96.58**	1.64
B	100101100111011	97.14	0.38	96.00	1.22
C	001111110111010	96.83	0.53	95.92	2.10
	100000011110010	96.70	0.41	96.35	1.10
1	111111111111111	96.88	0.62	**95.81**	1.33
W	010000001111101	95.77	0.57	**94.70**	1.05
D	110000011111111	95.51	0.48	94.66	1.23
B	010100001111001	95.04	0.59	93.96	1.37
C	110000101101110	95.07	0.76	94.13	1.26
	110000001111001	95.44	0.57	94.75	1.18
2	111111111111111	95.35	0.43	**93.79**	1.50

The results from Tables 7.9 and 7.10 is seen to be much similar to the Gamma Test method. The k-NN neighbour classifier again performed worse than the RANN. The subsets selected by the RANN was however also able to find good discriminatory features for the LIVER data set. Again it is seen that the k-NN classifier performs poorly when too many features of this data set is used. For the other three data sets, this problem does not seem to affect the k-NN classifier that much.

The fact that the RANN method is able to select, using only the apparent error rates from the training sets, good features for final classification is encouraging. For both the RANN classifier, as well as the k-NN classifier the results on the final test sets support the theory that this classifier do not suffer so much from over-training, and that therefor it would be

useful to use a feature selection tool.

7.3.4 Summarising Discussion of the Full Search Selection Results

The experiments on the different techniques using the full search method resulted in some interesting observations. The Gamma Test and RANN selection techniques were both able to find subsets that, on average, performs as well or better than the full set. This is true for using both an RANN and k-NN classifier to generate final test set error estimates.

The mutual information ranking technique method performed worse than the Gamma Test and RANN technique on these data sets. This was especially true for the LIVER data set, where the mutual information subsets never performed nearly similar to the full embedding.

The results of this chapter also confirms that the RANN classifier is able to outperform the simple k-NN technique on most of these data sets. This makes the RANN a good classifier to use in situations where training speed is of importance, since training of these networks are much faster than that of the normal back-propagation feed forward neural networks.

The fact that some of the subsets selected by the selection techniques contains less than two thirds of the full feature set, shows that for these standard data sets, classification results could be based on much less information than was extracted from the problems. This lower dimensionality might also be more robust in that the distance measures used in these classifiers are less affected by noise in the input features.

The next section investigates selection of features on larger subsets. For these cases the Gamma Test and RANN technique are not able to perform full searches of the input space.

7.4 Data Sets with More Features

7.4.1 PBIL Search on the Gamma Test and RANN Technique

The ION data set and full WDBC data set contains too many features to allow full searches of the input space. For this reason it was decided to use the PBIL search algorithm of Chapter 2 to find "good" subsets for these data sets[13].

The PBIL routine was only suited for use with the Gamma Test and RANN technique because these techniques allow criterion evaluations for any subset. For these techniques, the experiment was again performed by dividing the data sets into *ten* different samplings of training and testing sets. The PBIL routine was used to generate subsets for evaluation. For

[13]For implementation Al details of the PBIL optimisation routine, refer to the CDROM.

each training sampling, the top subsets emerging from the the PBIL routine was found. This was repeated *ten* times on each training set, to allow the routine to possibly report different local maxima[14]. In this way, the PBIL tests on every training sampling ended up reporting the top ten values found for a specific training sampling. From this the best solution was selected, yielding the top ten solutions on all training samplings of a specific data set.

The next step was to evaluate these ten solutions using the RANN and k-NN classifier. In a similar manner to the tests of the previous section, these best sets were first evaluated by searching for the best number of hidden neurons to use in the RANN classifier, and then final classifying the left-out test sets using the different subsets. The best 5 subsets found by this experiment for both the Gamma Test routine as well as the RANN selection technique is reported in Table 7.11.

Table 7.11: Top 5 embeddings obtained on RANN classifier using the PBIL search selection technique

selection method	data set	feature subset	% avg training performance	std. dev of training	% avg test performance	std. dev of test sets
RANN	ION	11110111010000010000000010001000101	93.38	1.07	**90.17**	2.97
		00111111100001110110000101000100001	94.57	1.63	89.06	2.91
		11101001000001010000100100101000000	92.95	2.43	88.72	2.22
		10001001010100100001110000000001001	93.38	1.90	88.55	1.49
		11101001101101000011010000000000000	91.11	2.09	88.38	2.65
		11111111111111111111111111111111111	91.58	2.94	**86.32**	1.87
RANN	WDBC	01111101001001001000111001000	97.92	0.44	**97.00**	1.03
		11010010001011011100101100100	97.55	0.78	96.05	1.27
		10001011001110011011011010101011	97.36	0.85	96.00	1.38
		11011101111001110001011100111	97.44	0.39	95.79	0.91
		01011011000000010000111010100	97.44	0.46	95.74	1.34
		11111111111111111111111111111	97.55	0.75	**96.37**	1.06
GAMMA	ION	10101101000101010100011101010100000	92.82	2.23	**88.46**	2.33
		10011101011001010101011011010101010	93.85	1.66	87.69	2.24
		10111111000101010101010100010001010	93.89	2.46	87.26	2.69
		11011111101010010101101100000111010	93.25	2.32	87.18	3.61
		11011111110101010101011111010101000	93.68	1.51	87.18	2.99
		11111111111111111111111111111111111	91.62	2.19	**86.84**	1.09
GAMMA	WDBC	01010001101101110110100001001	97.36	0.62	**96.05**	1.09
		01101000101111111111010000100	97.55	0.66	96.00	1.62
		01001001111111101011001010000	97.31	0.74	96.00	1.13
		11111010110101110110011110001	97.10	0.66	95.89	1.17
		01101010110011111111000001101	97.23	0.72	95.84	1.26
		11111111111111111111111111111	97.89	0.58	**96.16**	1.18

Similar to previous experiments, the comparative study of classification using the k-NN technique was also performed and these results are reported in Table 7.12.

[14]The routine does not guarantee an optimal subset, but due to the random nature of the search routine, different evaluations runs might return different local maxima in the search space

Table 7.12: Classification of subsets from Table 7.11 on a k-NN classifier selection technique

selection method	data set	feature subset	% avg training performance	std. dev of training	% avg test performance	std. dev of test sets
RANN	ION	11110111010000010000000010001000101	89.10	2.10	89.32	2.45
		00111111110000111011000010100010001	85.51	1.46	87.01	3.22
		11101001000001010000100100101000000	91.62	1.00	**90.43**	1.61
		10001001010100100001110000000001001	88.93	1.07	87.86	2.29
		11101001101101000011010000000000000	88.89	1.63	88.03	1.71
		11111111111111111111111111111111111	85.90	0.88	**85.98**	1.84
RANN	WDBC	01111101001001001000111001000	97.10	0.26	96.37	0.76
		11010010001011011100101100100	97.39	0.40	96.37	1.23
		10001011001110011011011010101011	97.47	0.43	96.79	1.04
		11011101111001110001011110011	97.60	0.34	95.79	1.10
		01011011000000010000111010100	97.55	0.24	**97.16**	0.82
		11111111111111111111111111111	97.57	0.48	**96.63**	1.06
GAMMA	ION	10101101000101010100011101010100000	89.70	1.25	89.74	1.25
		10011101011001010101011011010101010	88.55	1.81	88.55	1.81
		10111111000101010101010100010001010	90.85	0.96	90.26	0.96
		11011111101010010101101100000111010	89.74	1.55	**90.68**	1.55
		11011111110101010101011111010101000	87.86	1.73	88.63	1.73
		11111111111111111111111111111111111	85.90	0.88	**85.98**	0.88
GAMMA	WDBC	01010001101101111011010000100	96.99	0.54	95.37	1.12
		01101000101111111111010000100	96.65	0.76	95.47	1.11
		01001001111111101011001010000	96.78	0.44	95.58	1.72
		01001001111111101011001010000	97.39	0.48	**96.47**	1.15
		11111010111010111011001111000	96.20	0.50	95.00	1.09
		11111111111111111111111111111	97.57	0.48	**96.63**	1.06

Using PBIL, both the Gamma Test and the RANN selection technique where able to find feature subsets that reduced the dimensionality of these data sets tremendously without decreasing, or even by substantially increasing, the classification rates of both of the classifiers. For the ION data set, subsets are found that contain only a third of the features of the whole set yet still increasing the final classification rate of the data set.

The ION data set are also the only data set come across in this study, where the RANN classifier suffered significantly from the peaking phenomenon. Classifier error was decreased by 33% for this classifier using the best subset.

It is seen that the Gamma Test technique in this case performed marginally worse than the RANN technique when using the RANN classifier in the final tests. However, on the k-NN technique the top 5 subsets of the Gamma Test still contain subsets that perform as well as the RANN technique. The fact that the Gamma Test found good subsets for the RANN classifier without any knowledge of the classifier to be used, again shows empirically that the Gamma Test as a filter approach has inherent properties well suited to the task of feature selection.

7.4.2 Mutual Information Tests

In the case of the mutual information method, finding a decent experimental procedure was more difficult. This came about from the fact that the mutual information procedure does not by itself give the specific number of features to use, but just ranks the features according to their importance. This problem is common to all forward/backward selection techniques. For the data at hand, this would imply that for every sampling made from the data, the mutual information method would find possibly different rankings, resulting in a multitude of possible subsets to evaluate.

An heuristic solution for this problem might be to use the mutual information measurements in some way to find a stopping criterion for adding features. This criterion might be to stop when features that are left have only e.g. 5% of the information contained in the best feature (Here, the specific value chosen might have a huge impact on the number of features selected).

In previous research this problems was mostly addressed by simply choosing a specific number of features to select. However, this is also just an heuristic solution, and it seems as though a remedy for this problem has not yet been found.

For the ION and WDBC data sets it was decided to first find the different subsets resulting from the ranked sets before making decisions on the path to follow. This was done by building these subsets in a manner similar to that discussed in Section 7.3.2. This approach resulted in the need to evaluate 254 subsets for the ION data set and 181 subsets for the WDBC data set.

Although this entailed huge computational power, it was decided to train and evaluate all the different subsets on the two different data sets. Thus each of these subsets were used to train a RANN in a manner similar to previous experiments[15] and then to classify the testing data[16].

From these tests, the best 5 subsets are reported in Table 7.13. For more detail, Appendix C reports the 40 best embeddings obtained. Similar to the previous section Table 7.14 shows the comparative results for the k-NN classifier.

Similar to previous experiments, the comparative study of classification using the k-NN technique was again calculated, and the results are reported in Table 7.14.

For the ION data set the mutual information method displayed very interesting results. Firstly, it is seen that only 6 of the features are used in the subset that performs the best on

[15] Again the best number of neurons to use was first found by using a repeated search and evaluation approach.

[16] The tests took several days of parallel evaluation on 8 CPU's to complete, and would not be possible if the classifier was a normal feed-forward network trained with back-propagation.

Table 7.13: Top 5 embeddings obtained on RANN classifier using subsets found by the mutual information selection technique

data set	features subset	% avg training performance	std. dev of training	% avg test performance	std. dev of test sets
ION	01111000000000000000000000011000000	93.16	0.66	**90.60**	2.26
	01101001000000000000000000011000000	93.25	1.34	90.43	2.70
	11101001000000000000000000011000010	92.14	0.90	90.34	2.02
	11111000010000000000000000011001001	92.48	1.41	89.83	2.34
	11101110110000000000000010011101100	92.14	1.00	90.00	1.95
	11111111111111111111111111111111111	92.82	2.24	**87.86**	3.37
WDBC	11001000111111111111010101110111	97.18	0.959	**96.11**	1.16
	11001000001100110010010000011 1	97.15	0.484	96.11	1.29
	11011001111111111111010111111111	97.15	0.781	96.05	1.44
	11000001101100100011010010001 1	97.39	0.559	95.95	1.20
	11001001111110111111010111111111	97.52	0.690	95.95	1.33
	11111111111111111111111111111111	97.31	0.633	**96.32**	1.63

the RANN classifier. Furthermore, the classification rate seems to decrease when the number of features are made more or less then this number. A more detailed table of the 40 best feature subsets given in Appendix C clearly shows this relationship. From the Table in the appendix it is seen that some of the subsets containing a very small number of features still perform very well. This would mean that for this data set at least, dimensionality can be reduced to 20% of the original size and still robust classification (and even better classification) can be obtained. In the case of the k-NN neighbour classifier, a significant decrease in classification results result when using the full subset of features rather than the best set found by evaluation of the mutual information generated subsets on the ION data set.

From the results however it can be seen that the problem does not manifest itself in all data sets. For the full set of the WDBC data set, the final classification rates seem to be better when using almost the whole set of features. This was true for the PBIL tests on this data set as well, and thus the dimensionality problem seem to be very data dependent.

7.5 Summary of Experimental Work

In this chapter several comparative experiments were performed to investigate the success of the three different selection techniques. These results and experiments are now elaborated upon for each of the three different techniques.

In Section 7.2.1 it was shown that there exist high correlation between the measurements made with mutual information and classification rates of a classifier when using only single features. A problem with the mutual information strategy is that it cannot be used to directly

Table 7.14: Classification of subsets from Table 7.13 on a k-NN classifier

data set	features subset	% avg training performance	std. dev of training	% avg test performance	std. dev of test sets
I O N	01111000000000000000000011000000	91.54	0.74	90.17	1.59
	01101001000000000000000011000000	91.37	1.03	89.74	1.75
	11101001000000000000000011000010	91.03	1.13	**91.03**	2.07
	11111000010000000000000011001001	89.53	1.07	87.86	2.32
	11101110110000000000010011101100	89.66	1.18	90.00	1.91
	11111111111111111111111111111111	85.90	0.88	**85.98**	1.84
W D B C	11001000111111111110101110111	96.33	0.36	95.47	1.00
	11001000001100110010010000111	96.15	0.39	94.42	1.38
	11011001111111111110101111111	97.20	0.43	**95.68**	1.12
	11000001101100100011010010011	96.33	0.53	94.53	1.81
	11001001111110111110101111111	97.04	0.56	95.58	1.18
	11111111111111111111111111111	97.57	0.48	**96.63**	1.06

evaluate any subset due to problems of density estimation in high dimension. However, the high correlation and low variance in the mutual information estimates makes the method suited as an approach to decide whether any particular feature on its own is relevant or not.

The heuristic selection strategy of finding inter-feature mutual information seemed not to give very good results in the experiments of Section 7.3.2. It was found that for the LIVER test data set this method performed poorly when compared to the other selection techniques. In the previous section it was seen that the mutual information technique can however be useful when the number of features are to be kept very low, and the full subset contains many features. In these tests the best subsets found by the mutual information tests contained surprisingly little features in the case of the ION data set.

Correlation between classification rates and the Gamma Test values were also investigated. The experiments of Section 7.2.3 in particular, but also the correlation between error rates in all the different tests and Gamma values for these tests indicated that the test indeed seems to be highly related to classifier error. The subsets found by the test were generally as good as or even better than the RANN wrapper technique and mostly outperformed the mutual information technique. The fact that this test can be performed on any subset in a very short time, makes this method very attractive.

Another interesting observation was that subsets found by the Gamma Test selection technique performed equally well on the two different classifiers. This was very encouraging, since a drawback of most filter approaches to the selection problem is that they are sometimes not able to capture the biases of the classifier, and thus end up selecting subsets that are not ideal for the classifiers use. In the tests here however, this seemed to have little affect on the performance of the different subsets.

The RANN wrapper approach to the problem also performed very well. In most of the tests, subsets found by this technique also performed well on the k-Nearest Neighbour classifier. The fact that the evaluation time of this technique is short enough to allow fast testing, makes it one of only a few neural network methods suitable to vast input space searching.

Even though the data sets chosen for these experiments were not chosen because they were known to be particularly affected by dimensionality problems, some of these data sets showed significant increase in classification rates when using less features than the full set. The experimentation also confirms ideas discussed in Chapter 2 that the k-Nearest Neighbour classifier could be very badly affected by having too many input features. This was especially the case in the ION and LIVER data sets.

However, the experiments performed did not indicate the three techniques to be perfect. Although the RANN technique are very fast in terms of other neural networks, it still takes a very long time to perform a full search of the subsets of 15 input variables. Another problem with the approach is that the number of hidden neurons to be used is somewhat heuristic, and a search for the best number of neurons for every different subset would be impossible when the number of features become more than 8 or so.

The most serious drawback of the mutual information technique is that it does not give the user an idea of when to stop adding features, thus making it difficult to implement in practice when a very large number of inputs are available. The method are also prone to error due to the fact that a heuristic selection technique is applied and in the experiments done here, was the only method that did not yield good results on all tests.

Chapter 8

Conclusions

The problem of high dimensionality in pattern recognition systems is a very real one. In the work presented here, some origins of this problem were discussed and some previous work on countering the effects of this problem was highlighted. The background study showed that most classifiers are affected by the problem, and experimentation on some standard data sets confirmed this hypothesis for two different classifiers.

The three feature selection techniques that were chosen for investigation all seemed to be able to reduce the dimensionality of some standard machine learning data sets significantly. The experimental results showed that the application of these three selection techniques was able to find subsets that perform equally well, or even better than, the full set of features on two different types of classifiers.

Of particular interest for the neural network feature selection problem is the use of the RANN selection technique. This method was shown to have the ability of searching through a large number of subsets in significantly less time than most feed-forward artificial neural networks. This could make the method particularly suited for use in neural network approaches since it might have the same inherent biases as other neural network techniques. It was argued that the RANN has a distinct advantage above other neural networks because it does not need the explicit definition of a stopping criterion, which makes other neural networks difficult to use in direct application to feature selection.

The Gamma Test values showed surprising correlation with classifier error rates, and were able to select feature subsets that performed very well. The method was also able to pick up subsets that indicated the peaking phenomenon in some of these standard data sets. The number of features selected by the Gamma Test was sometimes more than those selected by other methods. However, the Gamma Test is aimed at finding the best subset in which to perform classification and is not therefore biased toward finding the minimum number of

features This bias could however be included as part of the PBIL selection routine criterion function.

The mutual information method was the only method that did not perform well in all cases. This indicates that, as explained in Chapter 2, the forward selection technique in feature subset selection might sometimes occlude important parts of the search space. Due to the use of the forward selection technique, the mutual information method also has the disadvantage of requiring a stopping criterion which, usually, is found by heuristic techniques. To circumvent the problem all different subsets generated by the test was evaluated in this work, but for more than 30 or so features, this method would soon become too computationally expensive.

An important realisation was that most data sets contain more than one particular subset that performs well. This casts the problem into that of finding a "good" local maximum rather than an optimal value. This fact makes selection by using stochastic search approaches attractive when the number of features become too high for exhaustive search.

The work presented here indicates a distinct need for feature selection and input dimensionality reduction in pattern recognition systems. Although all systems might not suffer from these problems, the use of feature reduction techniques on data sets with many features and only a small number of data points, might lead to improved classification results.

Areas for Future Work

The high correlation found between the Gamma Test and classification rates might have some other uses in the neural network field. The problem of finding a stopping criterion for neural network training techniques might be addressed by finding an analytic relationship between the Gamma test value and expected classifier error. Research into finding this relationship might have profound effects in this area.

The problem of intrinsic dimensionality is one that is not addressed in this work. There exist methods in the statistics literature to estimate (though not perfectly) the intrinsic dimensionality of a specific data set. This might be used as a method for deciding on the number of features to select when using a forward selection technique such as was used by the mutual information method.

The idea of including the cost features in the selection test is one that is not addressed here. The cost of some medical tests for example might prohibited it to be used if a number of other features with much lower cost can find a solution to the problem. Other promising extensions to the standard feature selection technique that was not investigated here, is that of using context sensitive feature selection techniques. These techniques might prove to be even more effective in cases where features are very dependent on each other.

Bibliography

[1] D. W. Aha and R. L. Bankert. Feature selection for case-based classification of cloud types: An empirical comparison. In *Proceedings of the 1994 AAAI Workshop on Case-Based Reasoning*, pages 106–112, Seatlle, 1994. AAAI Press.

[2] H. Almuallim and T. G. Dietterich. Learning boolean concepts in the presence of many irrelavant features. *Artificial Intelligence*, 69(1):279–305, 1994.

[3] P. Anderson. Using quasi-random numbers in neural networks. Invited lecture at the congress of non-linear analysis, July 1996. Athens, Greece.

[4] S. Baluja. Population based incremental learning. Technical Report CMU-CS-94-163, Carnegie Mellon University, 1994.

[5] S. Baluja. Genetic algorithms and explicit search statistics. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*. MIT Press, 1997.

[6] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5(4):537–550, July 1994.

[7] C. C. Beardah. Kernel density estimation toolbox for Matlab, version 1.1. http://maths.ntu.ac.uk/ccb/html/densest.html, May 1997.

[8] B. Bonnlander. *Non-parametric Selection of Input Variables for Connectionist Learning*. PhD thesis, University of Colorado, 1996.

[9] B. Bonnlander and A. Weigend. Selecting input variables using mutual information and nonparametric density estimation. *Proceedings, International Simposium on Artificial Neural Networks*, 1994.

[10] R. Caruana and D. Freitag. Greedy attribute selection. In *Proceedings of the 11th International Conference on Machine Learning*, pages 28–36, 1994.

[11] K. J. Cherkauer and J. W. Shavlik. Rapidly estimating the quality of input representations for neural networks. In *Proceedings of the IJCAI Workshop on Data Engineering for Inductive Learning*, 1995.

[12] K. J. Cherkauer and J. W. Shavlik. Growing simpler decision trees to facilitate knowledge discovery. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, 1996.

[13] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* Wiley, 1991.

[14] M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis*, 1(3), March 1997.

[15] H. Demuth and M. Beale. *Matlab Neural Network Toolbox.* Math Works Inc., 1992.

[16] P. A. Devijver. Advances in nonparametric techniques of statistical pattern recognition. In J. Kittler, K. Fu, and L. Pau, editors, *Pattern Recognition Theory and Applications*, volume 81 of *Mathematical and Physical Sciences*, pages 3–5. Reidel, 1982.

[17] P. Domingos. Context-sensitive feature selection for lazy learners. *Artificial Intelligence Review*, 11:227–253, 1997.

[18] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley-interscience, 1973.

[19] B. Frasca and R. Kohavi. Useful feature subsets and rough set reducts. In *Proceedings of the Third International Workshop on Rough Sets and Soft Computing*, 1994.

[20] A. M. Fraser and H. L. Swinney. Independant coordinates for strange attractors from mutual information. *Physical Review*, 33(2):1134–1140, February 1986.

[21] J. H. Friedman. On bias, variance, 0/1 - loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1(1):55–77, 1997.

[22] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, 2nd edition, 1990.

[23] R. Gaborski, P. Anderson, D. Tilley, and C. Asbury. Genetic algorithm selection of features for handwritten character identification. In *Proceedings of the 1993 International Conference on Artificial Neural Networks and Genetic Algorithms*, Insbruck, Austria, April 1993.

[24] S. Haykin. *Neural Networks: A comprehensive foundation.* Prentice Hall, 2nd edition, 1998.

[25] A. K. Jain and B. Chandrasekaran. Dimensionality and sample size considerations in pattern recognition practice. In P. Krishnaiah and L. Kanal, editors, *Handbook of Statistics, Classification Pattern Recognition and Reduction of Dimensionality*. North-Holland, 1982.

[26] J.-S. Jang, C.-T. Sun, and E. Mizutani. *Neuro-Fuzzy and Soft Computing*. Prentice-Hall, 1997.

[27] I. Jolife. *Principal Component Analysis*. Springer-Verlag, New-York, 1986.

[28] J. Kittler. Feature selection and extraction. In T. Y. Young and K. sun Fu, editors, *Handbook of Pattern Recognition and Image Processing*, chapter 3, pages 59–82. Academic Press, 1986.

[29] R. Kohavi. Feature subset selection as a search with probabilistic estimates. In *AAAI Fall Symposium on Relevance*. AAAI Press, 1994.

[30] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence Journal, special issue on relevance*, 97(1 and 2):273–324, 1997.

[31] D. Koller and M. Sahami. Toward optimal feature selection. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 284–292, 1997.

[32] N. Končar. Optimisation methodologies for direct inverse neurocontrol. Master's thesis, University of London, 1997.

[33] P. Langley, R. Kohavi, and Y. Yun. The utility of feature weighting in nearest-neighbour algorithms. In *ECML*, 1997.

[34] C. Lee and D. A. Landgrebe. Decision boundary feature extraction for nonparametric classification. *IEEE Transaction on systems, man, and cybernetics*, 23(2):433–444, March/April 1993.

[35] C.-T. Lin and C. G. Lee. *Neural Fuzzy Systems*. Prentice Hall, 1996.

[36] M. Loève. *Probability Theory*. Van Nostrand, Princeton, 1955.

[37] D. Lovell, C. Dance, M. Niranjan, R. Prager, and K. Dalton. Using upper bounds on attainable discrimination to select discrete valued features. *Networks for Signal Processing*, VI:233–242, 1996.

[38] K. Man, K. Tang, and S. Kwong. Genetic algorithms: Concepts and applications. *IEEE Transactions on Industrial Electronics*, 43(5):519, October 1996.

[39] S. Margetts. Gamma test research distribution package. Downloadable from http://www.cs.cf.ac.uk/ec, e-mail:S.Margetts@cs.cf.ac.uk, 1997.

[40] K. Messer and J. Kittler. A comparison of colour texture attributes selected by statistical feature selection and neural network methods. *Pattern Recognition Letters*, 18:1241–1246, 1997.

[41] P. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, C-26(9):917–922, Sept 1977.

[42] W. Nel, G. de Jager, and J. Greene. Improving pattern recognition systems using stochastic search methods. In *Proceedings of the 8th annual South African Workshop on Pattern Recognition*. IAPR, November 1997.

[43] J. Novovicova, P. Pudil, and J. Kittler. Divergence based feature selection for multimodal class densities. *IEEE transactions on pattern analysis and machine intelligence*, 18(2):218–223, February 1996.

[44] B. Pharinger. Compression based feature subset selection. In *Proceedings of the IJCAI-95 Workshop on Data Engineering for Inductive Learning*, pages 109–119, Montreal, Canada, 1995.

[45] P. Pudil, F. J. Ferri, J. Novovicova, and J. Kittler. Floating search methods for feature selection with nonmonotonic criterion functions. *Pattern Recognition Letters*, 15:119–125, 1994.

[46] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[47] M. Scherf and W. Brauer. Feature selection by means of a feature weighting approach. Technical Report FKI-221-97, Institute for Informatics, University of München, 1997.

[48] S. G. Schlosser, J. M. Trenkle, and R. C. Vogt. Feature set optimization for the recognition of arabic characters using genetic algorithms. In *Proceedings of the 21st Applied Imagery Pattern Recognition Workshop*, pages 64–73. SPIE, October 1992.

[49] D. W. Scott. *Multivariate Density Estimation. Theory, Practice and Visualization*. Wiley, 1992.

[50] R. Setiono and H. Liu. Neural-network feature selector. Technical report, Department of Information Systems, National University of Singapore, 1996.

[51] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

[52] D. B. Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Proceedings of the 11th International Machine Learning Conference*, pages 293–301, 1994.

[53] J. Smith, T. Fogarty, and I. Johnson. Genetic selection of features for clustering and classification. In *IEE Collequim on Genetic Algorithms in Image Processing and Vision*, volume 4 of *Digest no.1994/193*, pages 1–5, 1994.

[54] D. Sommerfield and R. Kohavi. Feature subset selection using the wrapper method: Overfitting and dynamic search space topology. In *First International Conference on Knowledge Discovery and Data Mining*, 1995.

[55] A. Stefánsson, N. Končar, and A. Jones. A note on the gamma test. *Neural Computing and Applications*, 5:131–133, 1997.

[56] D. Swets and J. Weng. Using discriminant eigenfeatures for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):831–836, 1996.

[57] R. Thawonmas and S. Abe. A novel approach to feature selection based on analysis of class regions. *IEEE Transactions on Systems, Man, and Cybernetics*, 27:196–207, April 1997.

[58] P. Turney. Feature selection bibliography. http://ai.iit.nrc.ca/bibliographies/feature-selection.

[59] P. Turney. Robust classification with context-sensitive features. In *Industrial and Engineering Applications of AI and Expert Systems (AIE93)*, pages 268–276. Gordon and Breach, 93.

[60] UCI. Machine learning data repository. http://www.ics.uci.edu/~mlearn.

[61] H. Vafaie and K. de Jong. Robust feature selection algorithms. In *Proceedings of the 5th International Conference on Tools for Artificial Intelligence*, pages 356–363, Boston, MA, 1993. IEEE Press.

[62] D. Whitley, J. Beveridge, C. Guerra-Salcedo, and C. Graves. Messy genetic algorithms for subset feature selection. In *International Conference on Genetic Algorithms*, 1997.

[63] J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. *IEEE Expert : Intelligent Systems and their Applications*, 13(2), March/April 1998.

Appendix A

CDROM

This appendix details some of the files residing on the CDROM. These files include the commented code used for some of the experiments of the previous chapter. Details are also given for the indexing of the PostScript files containing papers referenced in the thesis.

A.1 PostScript Files

The CDROM contains many of the papers referenced in the thesis. These files are stored in the */papers* directory. The files are named by the first author from the references. If more than one paper of the same author is present, the files are distinguished by the reference numbers from the bibliography as well.

The directory */theses* contains the theses by Bonnlander [8] and Končar [32].

A.2 UCI Data Sets

The directory */uci-data* contains the input data and associated class labels for all of the data sets used in the thesis in Matlab format. Data are always arranged with features represented by columns, and rows representing subsequent data points.

A.3 Matlab Code

The directory */matlab-code* contains commented Matlab code for most of the experiments of Chapter 7. Of particular interest might be the the following:

- *mutualinfoselect_kde.m* showing the implementation of the mutual information ranking routine.
- *mi_kern.m* that calculates the mutual information between two vectors of values.
- *compare_mi_classRANN.m* which shows the experiment for performing the correlations between the mutual information technique and the RANN classification.
- *compare_mi_classknn.m* which shows the experiment for performing the correlations between the mutual information technique and the k-NN classification.
- *compare_gamma_classRANN.m* used for finding correlation between the Gamma Test and the RANN.
- *compare_gamma_classknn.m* used for finding correlation between the Gamma Test and the k-Nearest Neighbour classifier.
- *makegamma.m* which converts Matlab data to a format readable by the Gamma Test programs.
- *mlprand.m* showing the implementation of the random artificial neural network.
- *mlptest.m* that classifies new data after training a random neural network.
- *classifyon1.m*,*classifyonallsets.m* and *mlp_class_embed.m* which classifies using a RANN all single features, all subsets of features and any specific embedding respectively. *mlp_final_embed.m* which classifies (using RANN) a certain set of embeddings using a very thorough method of searching for the best number of hidden neurons to use.
- *pbiloptimise.m* which shows the PBIL optimisation routine used.
- *gammafitnessfunc.m* and *mlpfitnessfunc.m* that were the fitness functions used for the gamma tests and the RANN tests respectively.
- *pbilgammafindtop1.m* which performs a PBIL search using the Gamma Test method as criterion function.
- *pbilrannfindtop1.m* that performs a PBIL search using the RANN wrapper technique.
- *create_3_feature_exampledata.m* that creates data for the simple three feature example used in Chapters 4,5 and 6.
- *classknn.m* and *classknn_test.m* showing the implementations of the k-Nearest Neighbour classifier for a leave-one-out training evaluation and a new test data evaluation.
- *knnclass_gammaranntop5.m*,*knnclass_miranntop5.m* and *knnclass_rannranntop5.m* which were used to test the 5 best sets reported in the different tests of Section 7.3. on a k-Nearest Neighbour classifier.
- *partitiondata.m* which divides a data set into a random 2/3 training, 1/3 testing partitioning.

Most of the Matlab files contains a short header of information stating the purpose of the program.

A.4 Gamma Test Binaries

The Gamma Test binaries resides in the directory */gammatest*. These binaries were obtained from Steve Margetts (S.Margetts@cs.cf.ac.uk). The directory contains binaries running under Solaris 2.4, Windows NT/95 and DOS.

A.5 Kernel Density Estimation Software

The kernel density software obtained from [7] is found in */kdetoolbox*. The toolbox contains its own documentation. For more information see the website at [7].

A.6 Electronic version of this document

Lastly an electronic version of this thesis can be found in the */document* directory.

A.7 A Note on the Matlab Code

Some of the code used in this work might make use of the Matlab Neural Network Toolbox. Unfortunately this toolbox is Copyrighted by Mathworks, and the files used could not be included on the CDROM.

Appendix B

Feature Sets used in Section 7.2.3

The 20 random feature sets that were chosen for the tests in Section 7.2.3 is tabulated below. It can be seen that for the LIVER data set, subsets four and five are the same. It was decided to keep these values equal in order to see whether the different techniques performed consistently in these cases. Furthermore, it is seen that the same subset masks were used for the two 15 feature data sets (WDBC1 and WDBC2).

Table B.1: Random feature subsets chosen for experiments in Section 7.2.3. The leftmost bit indicates the first feature, the second bit from the left indicates the second feature, and so forth. In the Matlab code the first feature was always placed in the first column of the data matrix.

feature subset number	LIVER	PIMA	WDBC1 and WDBC2
1	101001	11101001	101010100011001
2	101101	01011101	000001111011101
3	000001	11000001	101001100100001
4	000111	10100111	000101010100111
5	000111	00100111	010101010100111
6	110110	01001110	110001111001110
7	000110	00100110	010010110100110
8	101001	00011001	110010110011001
9	001111	01101111	001101011101111
10	011011	10111011	101110001111011
11	101101	01011101	011111111011101
12	011010	10111010	010111101111010
13	001011	00101011	001110110101011
14	111110	10000001	110010101000001
15	100111	10010111	110010001010111
16	111000	10000100	010001101000100
17	010110	11010110	011100000110110
18	100111	01010111	100110111010111
19	111011	10000111	011011001000111
20	000011	11000011	000000000100011

Appendix C

Mutual Information Selection on the ION Data

The Table below is used in Section 7.4.2 to point out some effects of dimensionality taking place in this specific data set.

Table C.1: Best 40 feature subsets from ION data found by RANN evaluation of the mutual information tests in Section 7.4.2. It is interesting to see how the final test set error seem to decrease when more than/or less than 6 features are used. Also note that there seem to exist a subsets of only 2 features that give comparable performance to the total of 34 features.

feature subset	Avg. training set classification	Std. dev on training sets	Avg. test set classification	Std. dev on test sets
01111000000000000000000011000000	93.16	0.66	90.60	2.26
01101001000000000000000011000000	93.25	1.34	90.43	2.70
11101001000000000000000011000010	92.14	0.90	90.34	2.02
11101110110000000000001011101100	92.14	1.00	90.00	1.95
11111000010000000000000011001001	92.48	1.41	89.83	2.34
11111000010000000000000011000011	90.64	1.71	89.74	1.91
11111000110000000000000011000011	91.79	1.21	89.66	3.47
01001000000000000000000011000000	90.90	0.60	89.57	1.42
01011000000000000000000000000000	90.43	0.88	89.57	1.94
00011000000000000000000000000000	90.43	0.88	89.57	1.94
11111000010000000000000011011000011	91.92	1.45	89.57	2.29
11101001110000000000000011000110	92.09	1.71	89.49	2.78
11101011110000000100000110110001110	92.26	1.27	89.49	1.99
11111000010000000000000011000011	92.69	1.11	89.32	2.30
11111010110000000100000110110001110	91.97	1.63	89.32	1.59
11111011110000001100000100111101101	93.12	1.14	89.32	2.10
11101011110000000000000110110001110	93.03	1.22	89.06	2.12
11111010110000000000000100110010001	92.14	0.72	89.06	2.64
11111011111111100010000011111111111	92.31	2.16	88.89	3.03
11101110110000000000000100110011100	91.79	1.01	88.80	1.89
11111010110000010000000110110011111	92.61	1.87	88.80	2.43
11111111111111100010000011111111111	93.93	2.38	88.72	2.64
11111011111101000100000111111111111	93.29	2.33	88.63	2.45
11111011110100000100000110111001110	92.74	1.68	88.63	2.89
11111010111100010000000110111011111	93.16	2.07	88.55	2.92
11111111111100000100001111111111111	93.50	2.39	88.55	2.79
11111111111111000100000111111111111	93.85	1.90	88.46	2.14
11111011111111001100000111111111111	93.25	1.62	88.38	2.14
11111111111110010110001111111111111	93.21	1.49	88.38	3.25
11111011111000001100000101111011101	92.82	1.50	88.29	3.04